SOLA DOM API Parser Reference

Revision: August 2017

Table of Contents

The SOLA DOM API

The Document Object Model (DOM) is a specification designed by the World-Wide-Web Consortium (W3C) to provide an object oriented and vendor independent way to inquire on and modify XML documents. DOM works by loading the entire XML document into memory and then converting the document into a tree structure. Inquiring on, modifying and creating a new XML document is done using methods provided by the DOM specification.

SOLA provides the SOLA DOM API to inquire on, modify and create XML documents. The SOLA DOM API is a "DOM like" API that implements the functionality of DOM in a format that is easily used by procedural languages like COBOL. The SOLA DOM API can be used by CICS transactions and batch jobs. It provides functions to inquire on an XML document repeatedly in any direction and modify an existing document. The API also provides methods to create new XML documents. This document lists all the features of the SOLA DOM API and provides many samples for use in mainframe programs running in CICS and batch.

In most cases, the SOLA DOM API is identical for CICS and batch programs. In those instances where the APIs differ, the differences will be noted in the text.

Advantages

The SOLA DOM API provides an easy and efficient way for any mainframe program to process XML documents. Performance (CPU utilization and memory consumption) was a major focus in the design and development of the SOLA DOM API. To keep CPU consumption to a minimum, the SOLA DOM API is highly optimized and is written entirely in Assembler for z/OS. The SOLA DOM API has a relatively small memory footprint for the DOM tree due to the fact that it parses the XML document and normalizes it in place without requiring any additional memory to hold the node values. The DOM tree contains pointers to the parsed and normalized document, thereby holding the memory footprint to a minimum.

The SOLA DOM API makes it easy for the mainframe programmer to produce well-formed XML without the need to perform complex string manipulations. Without the SOLA DOM API, the programmer would have to follow the XML

specifications provided by the W3C which can be a time consuming and daunting task.

For example, the following table shows incorrect XML formations that will be rejected by an XML parser:

`<SSN#>212-99-9654</SSN#>`	`The element name can't contain "#" Character in XML version 1.0`
`<SecurityName>AT&T </SecurityName>`	`The special entity "&" should be represented as &`

Because the SOLA DOM API conforms to the W3C specifications, XML documents created using the SOLA DOM API will always be valid and well formed.

DEFINITION OF TERMS

Node Identifier	Each node in the DOM tree is associated with and can be referenced by a unique identifier (the node identifier) for that node. The identifier (a binary fullword) is a unique, non changing value that can be used to uniquely identify each node within an XML document. Please be aware that the identifier is only unique within a given XML document; there is no guarantee that an identifier is unique across documents. Also, the identifier is only maintained, and is only guaranteed not to change, during the time that a node exists in the DOM tree. Should you externalize an XML document and then subsequently parse it again, there is no assurance that a node's id will be preserved, nor should there be any expectation that it will be unique across other XML documents. Throughout this document, the node identifier is referred to as WS-DOM-NODE-ID.
Element Node	A document contains at least one element. Elements are delimited by start-tags and end-tags, or, for empty elements by an empty-element tag. Each element has a name and may have a set of attributes and a text node. An element that contains other elements and/or attributes is called a complexType element.
complexType Element	An element node that has children. The children can be elements or attributes. An element node containing only text (and no other child) is not a complexType element.
Attribute Node	An attribute node is a child of an element node. Each attribute has a name and a value.
Text Node	The text value of an element is known as a text node. The SOLA DOM API doesn't independently maintain text nodes within the DOM tree. Instead, a text node is managed using an element node.

Namespace Node	Namespace nodes are managed independently by the SOLA DOM API. A namespace node has a namespace alias and a namespace value.
XPath Expression	XPath uses path expressions to select a node in an XML document. The node is selected by following a path or steps. The XPath expression follows a simple grammar that describes a node to be searched for by means of its place within the logical structure of the XML document.
Namespace	Namespace nodes are managed independently by the SOLA DOM API. A namespace can be obtained by using the getNamespace function. Additionally, the SOLA DOM API also includes functions to set a namespace. An update namespace function is not available in the current release. Currenly,the only way to update a namespace is to remove it using the removeNode function and set it again using the set function. Further capabilities to manage namespaces are scheduled for future releases of the SOLA DOM API.
First	Throughout this document the term *first* is used to describe which node will be retrieved whenever more than one node satisfies a set of search criteria. In this context, *first* refers to the first (logically) node in the XML document that meets the search criteria. If a node satisfying the same search criteria is inserted before another node that also satisfies the criteria, the newly inserted node would then be considered to be *first*.
Null	The term "Null" is used throughout this document to refer to a value that causes a variable to be ignored. The SOLA DOM API considers a variable to be null if the first four bytes of the variable are set to binary zeros.
Sequence Number	A sequence number is the relative position of a node in the tree and can change when nodes are inserted or deleted before it. This differs from the unique node identifier which is a non-changing identifier that can be used to uniquely identify a node.

SOLA DOM API FUNCTIONS

The SOLA DOM API distinguishes between element, attribute and namespace node identifier. These three node identifiers are maintained separately. Text node identifier is not maintained separately.

The SOLA DOM API is an assembler program callable from any mainframe program. The API supports inquiry and modification of XML documents as well as the creation of new XML documents.

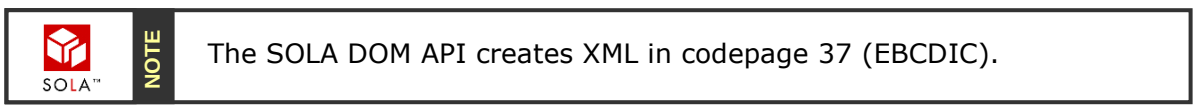
The API is case sensitive, so when using the API the case of the function names must be maintained. For this reason we recommend using the XMLDOMWS copybook 88 levels to set the WS-DOM-FUNCTION value.

XML Inquiry Functions

getAttribute	This function retrieves the value for a given attribute name in a name/value pair. This function can be used in **CICS** and **batch**.
getAttributeAddress	This function is identical to getAttribute, except that the attribute address is returned in place of the attribute value. This function can be used in **CICS** and **batch**.
getAttributeAddressById	The getAttributeAddressById function is identical to getAttributeById, except that the attribute value address is returned in place of the attribute value. This function can be used in **CICS** and **batch**.
getAttributeArray	This function retrieves an array containing all the attributes for a given element node. This function can be used in **CICS** and **batch**.

getAttributeById	This function retrieves an attribute and its value using a given node Id. This function can be used in **CICS** and **batch**.
getElementAddress	This function is identical to getElementByTagName, except that the element value address is returned in place of the element value. This function can be used in **CICS** and **batch**.
getElementAddressById	The getElementAddressById function is identical to getElementById, except that the element value address is returned in place of the element value. This function can be used in **CICS** and **batch**.
getElementById	This function retrieves an element and its child text node using a given node Id. This function can be used in **CICS** and **batch**.
getElementByTagName	The getElementByTagName retrieves an element node by its tag name. This function can be used in **CICS** and **batch**.
getElementByXPath	This function retrieves an element and its child text node for a given XPath expression. This function can be used in **CICS** and **batch**.
getNamespace	This function retrieves the value of a namespace associated with a namespace unique node identifier. This function can be used in **CICS** and **batch**.
Parse	This function parses, normalizes and creates the tree structure for a given input XML document. This function can be used in **CICS** and **batch**.

resetSelectAttrNodes	This function is used to reset the internal counters used by the selectAttrNodes function. The function should only be used in conjunction with the selectAttrNodes function when it is operating in an iterative mode. This function can be used in **CICS** and **batch**.
resetSelectNodes	This function is used to reset the internal counters used by the selectNodes function. The function should only be used in conjunction with the selectNodes function when it is operating in an iterative mode. This function can be used in **CICS** and **batch**.
selectAttrNodes	This function is used to inquire on either the total number of attribute nodes in an XML document, to get a particular attribute in the sequence or retrieve attributes iteratively. This function can be used in **CICS** and **batch**.
selectAttrNodesAddress	This function is identical to selectAttrNodes, except that the address of the attribute value is returned instead of the value of the attribute. This function can be used in **CICS** and **batch**.
selectNodes	This function is used to inquire on either the total number of element nodes in an XML document, to get a particular element in the sequence or retrieve elements iteratively. This function can be used in **CICS** and **batch**.
selectNodesAddress	This function is identical to selectNodes, except that the address of the element is returned instead of the value of the element. This function can be used in **CICS** and **batch**.



XML Creation and Modification Functions

appendChild	This function appends a new child element to a specified parent element. Optionally, a text node value can be attached to the element. This function can be used in **CICS** and **batch**.
appendChildBefore	This function is used to add an element before a particular element. The element node to be added will share the same parent as the element that it is to be appended before. This function can be used in **CICS** and **batch**.
appendChildNL	This function is identical to appendChild, except that it will also insert a newline character into the document before appending the child element node. This is a **batch** only function.
appendTextNode	This function is used to add a child text node to an element node. This function can be used in **CICS** and **batch**.
createDocument	This function creates the root element of an XML document. This function can be used in **CICS** and **batch**.

finalize	The finalize function is used to finalize and optionally retrieve a newly created XML document. Finalize will convert the DOM tree into an XML stream, as it does so all open tags will be closed and the document will be made ready to be externalized. The finalize function returns a pointer to the XML document. Once finalized, the document can be accessed with the address pointer. Alternatively, the document can be accessed using the retrieve function. This function can be used in **CICS** and **batch**.
getXMLLength	This function retrieves the length of the completed XML document. This is a **batch** only function.
removeNode	This function removes any type of node (element, attribute or namespace node) from an XML document. To remove text nodes, use the updateTextNode function. This function can be used in **CICS** and **batch**.
retrieve	This function finalizes the document and then gets the complete (if appendChildNL was not used) document and copies it to a data area that you provide. The SOLA DOM API will then release any storage space used by the DOM tree and internal control block. If appendChildNL was used then the retrieve function will fetch the document in pieces, each piece ending in a newline. This function can be used in **CICS** and **batch**.
setAttribute	This function is used to add and attach attributes (name/value pairs) to an element. This function can be used in **CICS** and **batch**.

setNamespace	This function is used to add and attach a namespace to an element. This function can be used in **CICS** and **batch**.
updateAttribute	This function is used to update the text of an existing attribute. Changing the attribute name is not allowed (to change the name of an attribute you must first remove and then set the attribute). This function can be used in **CICS** and **batch**.
updateTextNode	This function is used to change the value of or to remove an existing text Node under an element node. This function can be used in **CICS** and **batch**.

Miscellaneous Functions

freeStorage	This function frees any storage that has been acquired by the DOM parser and SOLA DOM API. This function can be used in **CICS** and **batch**.

COBOL COPYBOOKS

Copybook Name: XMLDOMWS

Programs that use the SOLA DOM API need to copy the copybook **XMLDOMWS** which is shipped with the product and can be found in the SAMPLIB directory. The following are the contents of this copy member:

```
01  WS-DOM-API-WORK-FIELDS.
    05  WS-DOM-API                  PIC  X(08) VALUE 'XMLPC112'.
    05  WS-DOM-RC                   PIC  S9(04) BINARY.
    05  WS-DOM-MSG                  PIC  X(80).
    05  WS-DOM-HANDLE               USAGE IS POINTER.
    05  WS-DOM-HANDLE-NUM           REDEFINES
        WS-DOM-HANDLE                PIC S9(8) COMP VALUE 0.
    05  WS-DOM-FUNCTION             PIC  S9(4) COMP-4.
*        Create Functions.
         88  WS-DOM-CREATE-DOC       VALUE +1.
         88  WS-DOM-APPEND-CHILD     VALUE +2.
         88  WS-DOM-APPEND-CHILD-BEFORE
                                     VALUE +3.
         88  WS-DOM-APPEND-CHILDNL   VALUE +4.
         88  WS-DOM-APPEND-TEXT-NODE VALUE +5.
         88  WS-DOM-SET-ATTRIBUTE    VALUE +6.
         88  WS-DOM-SET-NAMESPACE    VALUE +7.
         88  WS-DOM-UPDATE-ATTRIBUTE VALUE +8.
         88  WS-DOM-UPDATE-TEXT-NODE VALUE +9.
         88  WS-DOM-REMOVE-NODE      VALUE +10.
*        Value 11 and 12 are reserved for future use.
         88  WS-DOM-FINALIZE         VALUE +13.
         88  WS-DOM-FREE-STORAGE     VALUE +14.
*        Inquiry Functions.
         88  WS-DOM-PARSE            VALUE +15.
         88  WS-DOM-SELECT-NODES     VALUE +16.
         88  WS-DOM-GET-ATTRIBUTE    VALUE +17.
         88  WS-DOM-SEL-ATTR-NODES   VALUE +18.
         88  WS-DOM-GET-ATTRIBUTE-ID VALUE +19.
         88  WS-DOM-GET-ATTRIBUTE-ARRAY
                                     VALUE +20.
         88  WS-DOM-SEL-ATTR-NODES-ADR
                                     VALUE +21.
         88  WS-DOM-GET-ATTRIBUTE-ADR-ID
                                     VALUE +22.
         88  WS-DOM-GET-ATTRIBUTE-ADR
                                     VALUE +23.
         88  WS-DOM-RESET-SELECT-NODES
                                     VALUE +24.
         88  WS-DOM-RESET-SELECT-ATTR-NODES
                                     VALUE +25.
         88  WS-DOM-GET-NAMESPACE    VALUE +26.
         88  WS-DOM-GET-ELEMENT-BYTAG
                                     VALUE +27.
         88  WS-DOM-GET-ELEMENT-XPATH
```

```
                                   VALUE +28.
      88  WS-DOM-GET-ELEMENT-ID    VALUE +29.
      88  WS-DOM-GET-ELEMENT-ADR   VALUE +30.
      88  WS-DOM-GET-ELEMENT-ADR-ID
                                   VALUE +31.
      88  WS-DOM-SEL-NODE-ADR      VALUE +32.
      88  WS-DOM-RETRIEVE          VALUE +33.
      88  WS-DOM-GET-XML-LEN       VALUE +34.

05  WS-DOM-PARENT                PIC  X(256).
05  WS-DOM-SEQUENCE REDEFINES WS-DOM-PARENT.
    10  WS-DOM-SEQ               PIC S9(04) BINARY.
    10  WS-DOM-CONTROL REDEFINES    WS-DOM-SEQ
                                  PIC S9(04) BINARY.
    10  FILLER                   PIC  X(254).
05  WS-DOM-TAG-NAME              PIC  X(256).
05  WS-DOM-NMS-ALIAS REDEFINES WS-DOM-TAG-NAME PIC X(256).
05  WS-DOM-VALUE-LEN-X.
    10  WS-DOM-VALUE-LENGTH      PIC S9(08) BINARY.
05  WS-DOM-VALUE                 PIC  X(256).
05  WS-DOM-VALUE-PTR REDEFINES WS-DOM-VALUE.
    10  WS-DOM-VALUE-PTR USAGE IS POINTER.
    10  WS-DOM-ADDRESS REDEFINES
        WS-DOM-VALUE-PTR          PIC S9(08) BINARY.
    10  WS-DOM-VALUE-LITERAL     PIC X(252).
05  WS-DOM-NAMESPACE REDEFINES WS-DOM-VALUE PIC X(256).
05  WS-DOM-NODE-ID               PIC S9(08) BINARY.
05  WS-DOM-ELEMENT-ID REDEFINES WS-DOM-NODE-ID
                                  PIC S9(8) BINARY.
05  WS-DOM-ATTRIBUTE-ID REDEFINES WS-DOM-NODE-ID
                                  PIC S9(8) BINARY.
05  WS-DOM-XML-LENGTH   REDEFINES WS-DOM-NODE-ID
                                  PIC S9(08) BINARY.
05  WS-DOM-NMS-NODE-ID           PIC S9(8) BINARY.
05  WS-DOM-CHILD-ELEM-COUNT      PIC S9(08) BINARY.
05  WS-DOM-ATTR-NODE-COUNT       PIC S9(08) BINARY.
05  WS-DOM-PARENT-NODE-ID        PIC S9(08) BINARY.
05  WS-DOM-NODE-TYPE             PIC S9(4) COMP.
     88 WS-DOM-ELEMENT-NODE      VALUE 1.
     88 WS-DOM-ATTRIBUTE-NODE    VALUE 2.
     88 WS-DOM-NAMESPACE-NODE    VALUE 3.
05  WS-DOM-MAX-LENGTH            PIC S9(08) BINARY.
05  WS-DOM-PLACE-HOLDER          PIC S9(08) BINARY VALUE 0.
05  FILLER                       PIC  X(77).
```

RETURN CODES

All function calls include the WS-DOM-RC. The following are the possible return code values received from the API call:

RC Value	Meaning
0	Successful - No messages issued.
+1	A ComplexType element has ended or a Message Set has ended.
+2	Entire DOM Tree is traversed. SOLA DOM API will reset the iterator.
+3	Text node already exists. No action taken.
+4	Inquiry node Not found - Messages were issued in field WS-DOM-MSG
+5	Returned data is bigger than allowed. Data truncated to the specified Length.

RC Value	Meaning
+8	Failed - failed message returned in WS-DOM-MSG. The following are the possible values for WS-DOM-MSG field: • Attribute not found • The SOLA DOM API call doesn't conform to call syntax • Element tag name not specified • The function specified is not supported • Input error • The namespace value is not specified • Neither parent name or parent node is specified • Node identifier doesn't exist • Node type not recognized • Not enough memory to create the tree • Parent not found • Parent node identifier does not exist • Parse failed due to XML Syntax error • Reference node identifier doesn t exist • Root element tag name not specified • Sequence value not specified • Specified Length of input xml is zero • Specified parent node doesn't exist • Specified parent node identifier doesn't exist • Specified attribute node identifier doesn't exist • Specified attribute node doesn't exist • Specified namespace node identifier doesn't exist • Seq too high (The sequence values is higher than the total number element nodes in the document) • XPath starts with a "/" but root element name doesn't match
+12	SOLA DOM API abended. Abend code – XXXX (four character code returned at run time).
+16	WS-DOM-HANDLE doesn't contain a valid handle.

SOLA DOM API Reference

This chapter contains a description of syntax conventions and return codes for each of the SOLA DOM API calls, followed by a detailed description of each call.

Invocation of Services

The color **red** depicts optional variables while the color **blue,** underlined, depicts required variables. For example, this variable is required; **WS-DOM-VALUE-LENGTH**, and this variable is optional; **WS-DOM-NODE-ID**.

Each API is implemented as a CALL. For illustration purposes, each CALL is shown using COBOL syntax. The actual CALL can be implemented in any language that follows the register 1 parmlist convention.

Included in each API description is an example of its use in COBOL.

The basic SOLA DOM API call syntax is as follows:

```
CALL WS-DOM-API USING
       WS-DOM-RC,                 /* return code */
       WS-DOM-MSG,                /* error message */
       WS-DOM-HANDLE,             /* Pointer */
       WS-DOM-FUNCTION,           /* Function requested */
       WS-DOM-SEQ/WS-DOM-PARENT,  /* Sequence Number/Parent */
       WS-DOM-TAG-NAME,           /* XML Tag value /*
       WS-DOM-VALUE,              /* Data */
       WS-DOM-VALUE-LENGTH,       /* Data length */
       WS-DOM-PARENT-NODE-ID,     /* Parent node identifier */
       WS-DOM-NODE-ID,            /* Unique ID in the tree */
       WS-DOM-NMS-NODE-ID         /* Unique ID for the namespace node */
       WS-DOM-CHILD-ELEM-COUNT,   /* Number of children */
       WS-DOM-ATTR-NODE-COUNT,    /* Attr node count */
       WS-DOM-MAX-LENGTH          /* Max length to be returned */
```

This basic syntax is not followed by the createDocument, parse, getAttributeArray, and removeNode functions. Please refer to the description of those functions for the call syntax.

Variables

In each of the examples in this section, the API calls are shown using variables from the copybook XMLDOMWS. Use of this copybook is optional. Because the API is implemented as a CALL USING VAR1, VAR2 … VARn, each parameter in the call is position sensitive and is passed to the API as an address reference. You can substitute your own variables for the ones provided in XMLDOMWS. The API is sensitive to the lengths of the binary integer variables. There is no implied length of character string variables. For these variables the API relies on the length attribute (for data values) or space delimited values (tag name values). Consequently there is no limit imposed by the API for the length of a tag name or a data value.

If a variable is not used but there are variables following that variable that are used, then this document uses the notation WS-DOM-PLACE-HOLDER in place of the unused variables. You can substitute any value you wish for WS-DOM-PLACE-HOLDER, but the position of the variables within the positional variable list must be maintained.

If an optional variable is not used as input to the function, and is required to serve as a place holder for a subsequent optional variable, it must have a value of Null. A non-Null value could imply that the variable is populated with a significant value. In the examples in this document we use the variable WS-DOM-PLACE-HOLDER as a place holder for unused variables. This variable is pre-set to binary zero in the XMLDOMWS copybook (binary zero, or low-values, is considered to be a null value).

SOLA DOM API calls that don’t conform to the call syntax for the function being called will result in a return code +8 being returned by the SOLA DOM API. WS-DOM-MSG will contain a description of the error being returned.

Variable Definitions

Each of the variables used in the CALL format are described below. The descriptions use the variables from the copybook XMLDOMWS. Throughout this document, the XMLDOMWS variables are used in each function call description. You are free to substitute your own variables in place of XMLDOMWS variables.

WS-DOM-RC	A halfword binary variable. Return codes are described in the Return Codes section on page 14.
WS-DOM-MSG	A Char(80) variable. When WS-DOM-RC is non-zero, this variable will contain the error message issued by the SOLA DOM API.
WS-DOM-HANDLE	A pointer variable. This pointer is used by the SOLA DOM API to keep track of the DOM tree and associated control blocks. Before the first call to the SOLA DOM API to create or parse an XML document, the DOM Handle should be set to null. It is possible to manipulate multiple XML documents simultaneously by managing multiple DOM handles.
WS-DOM-FUNCTION	An S9(4) COMP-4 variable. This variable must be set to the relevant function call. XMLDOMWS Copybook contains a list of functions as 88 levels. We recommend that you set this variable by using the COBOL SET statement.
WS-DOM-PARENT	This is a space delimited variable. The name of the parent node for a node being added or updated.
WS-DOM-SEQ	The sequence number to retrieve a node. Can have multiple uses, as described in each function call. WS-DOM-SEQ redefines WS-DOM-PARENT in the copybook because they are mutually exclusive fields.

WS-DOM-TAG-NAME	This is a space delimited variable. When using the append functions, this variable contains the name of the tag to be added to the XML document. When using the get by name functions, this variable contains the name of the tag to be retrieved from the XML document. When using the get by XPath function, this variable contains the XPath expression used to locate the node in the XML document. The variable WS-DOM-NMS-ALIAS, which is used for getting and setting a namespace, redefines WS-DOM-TAG-NAME in the XMLDOMWS copybook. It is the same field as WS-DOM-TAG-NAME but is renamed for clarity.
WS-DOM-VALUE	For append and update functions, this variable contains the data string to be added to or modified in the XML document. For get functions this variable is used to receive the data string to be retrieved from the XML document. The variable WS-DOM-NAMESPACE, which is used for getting and setting a namespace, redefines WS-DOM-VALUE in the XMLDOMWS copybook. It is the same field as WS-DOM-VALUE but is renamed for clarity. WS-DOM-PTR also redefines WS-DOM-VALUE and is used during *Address functions.
WS-DOM-VALUE-LENGTH	A binary fullword variable. For append and update functions this variable contains the length of the data string to be added to or modified in the XML document. For get functions this variable receives the length of the data string retrieved from the XML document.
WS-DOM-PARENT-NODE-ID	The unique identifier for the parent node. When used, overrides WS-DOM-PARENT. Can have multiple uses, as described in each function call.
WS-DOM-NODE-ID	This is the unique ID of a node. It can be used to retrieve a node with the getElementById and getAttributeById functions. It will be returned by the SOLA DOM API if the variable is present on the getElementBy*, getAttributeBy* and select* function calls.

WS-DOM-ELEMENT-ID	Redefines WS-DOM-NODE-ID. The unique ID of an element.
WS-DOM-ATTRIBUTE-ID	Redefines WS-DOM-NODE-ID. The unique ID of an attribute.
WS-DOM-NMS-ALIAS	Redefines WS-DOM-TAG-NAME. This variable is used to define the namespace alias, which can be used in place of the namespace.
WS-DOM-NAMESPACE	Redefines WS-DOM-VALUE. This variable is used for getting and setting a namespace.
WS-DOM-VALUE-PTR	Redefines WS-DOM-VALUE. This variable contains an address pointer to a value.
WS-DOM-NODE-TYPE	This represents the node type. Options are element node, attribute node or namespace node.
WS-DOM-NMS-NODE-ID	This is the unique ID for a namespace node. It can be used to retrieve a specific namespace. It is returned, if the element node has a namespace, by the getElement* and selectNodes functions.
WS-DOM-CHILD-ELEM-COUNT	A binary fullword variable. If specified, the number of child elements for a complexType element will be returned in this variable.
WS-DOM-ATTR-NODE-COUNT	A binary fullword variable. If specified for the getElementBy function calls the number of attributes for an element will be returned in this variable.
WS-DOM-MAX-LENGTH	A binary fullword variable. If specified, data in WS-DOM-VALUE will be truncated to this length. WS-DOM-VALUE-LENGTH will contain the actual (un-truncated) length of the data.
WS-DOM-CONTROL	This variable controls how the XML document is created or parsed. Please see the CreateDocument and Parse functions for a desctiption of how this variable is used.
WS-DOM-ARRAY	A structured variable. WS-DOM-ARRAY is a variable length table of lengths and addresses. This structure is used to receive a variable list of pointers and lengths from the getAttributeArray function.

WS-DOM-PLACE-HOLDER	A null fullword variable. Because function calls to the SOLA DOM API use a CALL USING syntax, the position of the variables in the (USING) list is important. WS-DOM-PLACE-HOLDER can be used in the call list in place of an unused variable when a variable position in the list is not used but variables after that position are used. The value of WS-DOM-PLACE-HOLDER is set to null in the copybook although the SOLA DOM API may not look at the value of the field for functions where WS-DOM-PLACE-HOLDER is used.

API DESCRIPTIONS – XML INQUIRY FUNCTIONS

getAttribute

CICS BATCH

The getAttribute function retrieves the value for a given attribute name in a name/value pair. The default scope of the attribute name search is the entire XML document. If multiple instances of the attribute exist in the XML document, the instance of the attribute corresponding to a sequence number will be returned.

The default search scope can be overridden in two ways:

- First, by specifying a parent node identifier on input to the request. The search will then be limited to finding an attribute with a matching name for the given parent node identifier.
- Second, by specifying a parent tag name instead of a sequence number or a parent node identifier. This second search scope will find the first instance of an attribute with a matching name and parent tag name combination.

The precedence of search criteria is as follows: parent node identifier has precedence over parent tag name or attribute sequence number. Parent tag name and attribute sequence number are mutually exclusive.

After the call the WS-DOM-VALUE will contain the value of the attribute and WS-DOM-VALUE-LENGTH will be set to the length of the attribute value.

The getAttribute function can be used in CICS and batch.

```
CALL WS-DOM-API USING
      WS-DOM-RC,        /* Halfword. Return Code */
      WS-DOM-MSG,       /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,    /* Pointer.  For internal use */
      WS-DOM-FUNCTION, /* Char(25).  The name of the API */
      WS-DOM-SEQ/WS-DOM-PARENT, /* Sequence Number/Parent */
      WS-DOM-TAG-NAME, /* Name of the tag to search for */
      WS-DOM-VALUE,       /* The text node is returned here */
      WS-DOM-VALUE-LENGTH, /* Length of the returned text node */
      WS-DOM-PARENT-NODE-ID, /* The ID of the parent of the node */
      WS-DOM-ATTRIBUTE-ID,    /* The node identifier will be returned
      here */
      WS-DOM-PLACE-HOLDER, /* placeholder */
```

```
WS-DOM-PLACE-HOLDER, /* Placeholder */
WS-DOM-PLACE-HOLDER, /* Placeholder */
WS-DOM-MAX-LENGTH /* Can specify max length of data here */
```

Variables:

WS-DOM-RC **Output** **Mandatory**

Returns status.

WS-DOM-MSG **Output** **Mandatory**

When the return code is not zero, this field will contain messages.

WS-DOM-HANDLE **Input** **Mandatory**

For internal use. The DOM handle field uniquely identifies the XML DOM tree from which the attribute will be retrieved.

WS-DOM-FUNCTION **Input** **Mandatory**

Must be set to **WS-DOM-GET-ATTRIBUTE**.

WS-DOM-SEQ/WS-DOM-PARENT **Input** **Mandatory**

This field instructs the SOLA DOM API to retrieve a particular instance of an attribute where WS-DOM-TAG-NAME matches a tag name in the tree. WS-DOM-PARENT can be used instead of WS-DOM-SEQ. When WS-DOM-PARENT is used, the DOM-API tries to retrieve an attribute for which the parent and attribute tag name combination match. If WS-DOM-PARENT-NODE-ID is specified as input (value other than null), then WS-DOM-PARENT is ignored.

WS-DOM-TAG-NAME **Input** **Mandatory**

This field instructs the SOLA DOM API to search for an attribute with a matching tag name.

WS-DOM-VALUE **Output** **Mandatory**

This field will contain the normalized text value of the attribute.

WS-DOM-VALUE-LENGTH **Output** **Mandatory**

This field will contain the length of the text value of the attribute.

WS-DOM-PARENT-NODE-ID **Input/Output** **Optional**

This field is optional. When specified in the call, the parent node identifier limits the scope of the search (for an attribute name that matches the WS-DOM-TAG-NAME) to a specific parent node. Please note that parent node is always an element node. If this field is specified on the call, then after a successful call, the SOLA DOM API will populate this variable with the node identifier of the parent element.

WS-DOM-ATTRIBUTE-ID **Output** **Optional**

After a successful call, the SOLA DOM API will populate this variable with the node identifier of the attribute. This variable is a redefinition of WS-DOM-NODE-ID.

WS-DOM-PLACE-HOLDER	Placeholder	Optional
WS-DOM-PLACE-HOLDER	Placeholder	Optional
WS-DOM-PLACE-HOLDER	Placeholder	Optional
WS-DOM-MAX-LENGTH	Input	Optional

This optional field, when specified, tells the SOLA DOM API to truncate the data in WS-DOM-VALUE field if WS-DOM-VALUE-LENGTH exceeds WS-DOM-MAX-LENGTH. A return code of +5 is returned if the returned data is truncated. In this case, WS-DOM-VALUE-LENGTH contains the actual data length but the data in WS-DOM-VALUE is truncated to WS-DOM-MAX-LENGTH.

The following return codes are possible:

RC Value	Meaning
0	Successful - No messages issued.
+4	Attribute Not found - Messages were issued in field WS-DOM-MSG
+5	Returned data is bigger than allowed. Data truncated to the specified Length.
+8	Failed. The following are the possible values for WS-DOM-MSG field: • Seq too high (one of the sequence values is higher than the total number element nodes in the document) • Specified parent node identifier doesn't exist • Input error Possible input errors could be: • The function specified is not supported. • Attribute name not specified. • Neither parent name or parent node is specified.
+12	SOLA DOM API abended. Abend code - XXXX (four character code returned at run time).
+16	WS-DOM-HANDLE doesn't contain a valid handle.

Example

In this example an attribute named 'Type' is being searched for. The scope of the search is the entire XML document. The SOLA DOM API will search for the first attribute that meets this search criterion.

```
SET  WS-DOM-GET-ATTRIBUTE TO TRUE
MOVE 1                    TO WS-DOM-SEQ
MOVE 'Type'               TO WS-DOM-TAG-NAME

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-SEQ,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE,
                      WS-DOM-VALUE-LENGTH
```

In this next example, an attribute named 'Type' is being searched for. We are limiting the search scope to within the parent 'Transaction'. If there are multiple instances of the parent element 'Transaction', the *first* such element is selected.

```
SET  WS-DOM-GET-ATTRIBUTE TO TRUE
MOVE 'Type'               TO WS-DOM-TAG-NAME
MOVE 'Transaction'        TO WS-DOM-PARENT

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PARENT,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE,
                      WS-DOM-VALUE-LENGTH
```

In this next example, we are searching for an attribute named 'Type'. We are limiting the search scope to within the parent that has a unique node identifier that was saved in SAVED-PARENT-ID.

```
SET  WS-DOM-GET-ATTRIBUTE TO TRUE
MOVE 'Type'               TO WS-DOM-TAG-NAME
MOVE SAVED-PARENT-ID      TO WS-DOM-PARENT-NODE-ID

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE,
                      WS-DOM-VALUE-LENGTH,
                      WS-DOM-PARENT-NODE-ID
```

In this next example, we are searching for an attribute named 'Type'. We are limiting the search scope to within the parent that has a unique node identifier that was saved in SAVED-PARENT-ID. We are requesting that the unique node identifier of the attribute be returned.

```
SET  WS-DOM-GET-ATTRIBUTE TO TRUE
MOVE 'Type'               TO WS-DOM-TAG-NAME
MOVE SAVED-PARENT-ID      TO WS-DOM-PARENT-NODE-ID

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE,
                      WS-DOM-VALUE-LENGTH,
                      WS-DOM-PARENT-NODE-ID,
                      WS-DOM-ATTRIBUTE-ID
```

getAttributeAddress

CICS BATCH

The getAttributeAddress function is identical to getAttribute, except that the address of attribute value is returned in place of the attribute value. This can be useful if you don't want to provide storage for the attribute value, or where you want to use the XMLDOMWS copybook variables and the attribute value is greater than 256 bytes.

The getAttributeAddress function retrieves the attribute value address for a given attribute name in a name/value pair. The default scope of the attribute search is the entire XML document. If multiple instances of the attribute exist in the XML document, the instance of the attribute corresponding to a sequence number will be returned.

The default search scope can be overridden in two ways:

- First, by specifying a parent node identifier on input to the request. The search will then be limited to finding an attribute with a matching name for the given parent node.
- Second, by specifying a parent tag name instead of a sequence number or a parent node identifier. This second search scope will find the first instance of an attribute with a matching name and parent tag name combination.

The precedence of search criteria is as follows: parent node identifier has precedence over parent tag name or attribute sequence number. Parent tag name and attribute sequence number are mutually exclusive.

After the call, the WS-DOM-VALUE-PTR will contain the address of the attribute and WS-DOM-VALUE-LENGTH will be set to the length of the attribute value.

The getAttributeAddress function can be used in CICS and batch.

```
CALL WS-DOM-API USING
      WS-DOM-RC,        /* Halfword. Return Code */
      WS-DOM-MSG,       /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,    /* Pointer.  For internal use */
      WS-DOM-FUNCTION, /* Char(25).  The name of the API */
      WS-DOM-SEQ/WS-DOM-PARENT, /* Sequence Number/Parent */
      WS-DOM-TAG-NAME, /* Name of the tag to search for */
      WS-DOM-VALUE-PTR,       /* The address is returned here */
      WS-DOM-VALUE-LENGTH, /* Length of the returned text node */
      WS-DOM-PARENT-NODE-ID, /* The ID of the parent of the node */
      WS-DOM-ATTRIBUTE-ID    /* The node id will be returned here */
```

Variables:

WS-DOM-RC **Output** **Mandatory**
Returns status.

WS-DOM-MSG **Output** **Mandatory**
When the return code is not zero, this field will contain messages.

WS-DOM-HANDLE **Input** **Mandatory**
For internal use. The DOM handle field uniquely identifies the XML DOM tree from which the attribute will be retrieved.

WS-DOM-FUNCTION **Input** **Mandatory**
Must be set to **WS-DOM-GET-ATTRIBUTE-ADR**.

WS-DOM-SEQ/WS-DOM-PARENT **Input** **Mandatory**
This field instructs the SOLA DOM API to retrieve the address of a particular instance of an attribute where WS-DOM-TAG-NAME matches a tag name in the tree. WS-DOM-PARENT can be used instead of WS-DOM-SEQ. When WS-DOM-PARENT is used, the DOM-API tries to retrieve the address of an attribute for which the parent and attribute tag name combination match. If WS-DOM-PARENT-NODE-ID is specified as input (value other than null), then WS-DOM-PARENT is ignored.

WS-DOM-TAG-NAME **Input** **Mandatory**
This field instructs the SOLA DOM API to search for an attribute with a matching tag name.

WS-DOM-VALUE-PTR **Output** **Mandatory**
This field will contain the attribute's address.

WS-DOM-VALUE-LENGTH **Output** **Mandatory**
This field will contain the length of the text value of the attribute.

WS-DOM-PARENT-NODE-ID **Input/Output** **Optional**
This field is optional. When specified in the call, the parent node identifier limits the scope of the search (for an attribute name that matches the WS-DOM-TAG-NAME) to a specific parent node. Please note that parent node is always an element node. After a successful call, the SOLA DOM API will populate this variable with the node identifier of the parent element.

WS-DOM-ATTRIBUTE-ID **Output** **Optional**
After a successful call, the SOLA DOM API will populate this variable with the node identifier of the attribute. This variable is a redefinition of WS-DOM-NODE-ID.

The following return codes are possible:

RC Value	Meaning

0	Successful - No messages issued.
+4	Attribute Not found - Messages were issued in field WS-DOM-MSG
+8	Failed. The following are the possible values for WS-DOM-MSG field: • Seq too high (one of the sequence values is higher than the total number element nodes in the document) • Specified parent node identifier doesn't exist • Input error Possible input errors could be: • The function specified is not supported. • Attribute name not specified. • Neither parent name or parent node is specified.
+12	SOLA DOM API abended. Abend code - XXXX (four character code returned at run time).
+16	WS-DOM-HANDLE doesn't contain a valid handle.

Example

In this example, an attribute named 'Type' is being searched for. The SOLA DOM API will search for the first attribute that meets this search criteria.

```
SET  WS-DOM-GET-ATTRIBUTE-ADR TO TRUE
MOVE 1                        TO WS-DOM-SEQ
MOVE 'Type'                   TO WS-DOM-TAG-NAME

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-SEQ,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE-PTR,
                      WS-DOM-VALUE-LENGTH
```

In this next example, an attribute named 'Type' is being searched for. We are limiting the search scope to within the parent 'Transaction'. If there are multiple instances of the parent element 'Transaction', the *first* such element is selected.

```
SET  WS-DOM-GET-ATTRIBUTE-ADR TO TRUE
```

```
MOVE 'Type'                    TO WS-DOM-TAG-NAME
MOVE 'Transaction'             TO WS-DOM-PARENT

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PARENT,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE-PTR,
                      WS-DOM-VALUE-LENGTH
```

In this next example, we are searching for an attribute named 'Type'. We are limiting the search scope to within the parent that has a unique node identifier that was saved in SAVED-PARENT-ID.

```
SET  WS-DOM-GET-ATTRIBUTE-ADR TO TRUE
MOVE 'Type'                   TO WS-DOM-TAG-NAME
MOVE SAVED-PARENT-ID          TO WS-DOM-PARENT-NODE-ID

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE-PTR,
                      WS-DOM-VALUE-LENGTH,
                      WS-DOM-PARENT-NODE-ID
```

In this next example we are searching for an attribute named 'Type'. We are limiting the search scope to the parent that has a unique node identifier that was saved in SAVED-PARENT-ID. We are requesting that the unique node identifier of the attribute to be returned, along with the pointer to the text value of the attribute found.

```
SET  WS-DOM-GET-ATTRIBUTE-ADR TO TRUE
MOVE 'Type'                   TO WS-DOM-TAG-NAME
MOVE SAVED-PARENT-ID          TO WS-DOM-PARENT-NODE-ID
CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE-PTR,
                      WS-DOM-VALUE-LENGTH,
                      WS-DOM-PARENT-NODE-ID,
                      WS-DOM-ATTRIBUTE-ID
```

getAttributeAddressById

CICS BATCH

The getAttributeAddressById function is identical to getAttributeById, except that the attribute value address is returned in place of the attribute value. This can be useful if you don't want to provide storage for the attribute value, or where you want to use the XMLDOMWS copybook variables and the attribute value is greater than 256 bytes.

The getAttributeAddressById function retrieves an attribute address and its value using a given node Id. The node identifier (shown below as WS-DOM-NODE-ID) is the unique node identifier for the attribute node. It is used to identify and locate the node in the tree.

The getAttributeAddressById function can be used in CICS and batch.

```
CALL WS-DOM-API USING
      WS-DOM-RC,         /* Halfword. Return Code */
      WS-DOM-MSG,        /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,     /* Pointer.  For internal use */
      WS-DOM-FUNCTION,   /* Char(25).  The name of the API */
      WS-DOM-PLACE-HOLDER,             /* placeholder */
      WS-DOM-TAG-NAME,    /* The tag name is returned here */
      WS-DOM-VALUE-PTR,       /* The text node is returned here */
      WS-DOM-VALUE-LENGTH,  /* Length of the returned text node */
      WS-DOM-PARENT-NODE-ID,  /* The ID of the parent of the node */
      WS-DOM-ATTRIBUTE-ID     /* The ID of the node to be retrieved */
```

Variables:

WS-DOM-RC **Output** **Mandatory**

Returns status.

WS-DOM-MSG **Output** **Mandatory**

When the return code is not zero, this field will contain messages.

WS-DOM-HANDLE **Input** **Mandatory**

For internal use. The DOM handle field uniquely identifies the XML DOM tree from which the attribute will be retrieved.

WS-DOM-FUNCTION **Input** **Mandatory**

Must be set to **WS-DOM-GET-ATTRIBUTE-ADR-ID**.

WS-DOM-PLACE-HOLDER **Place holder** **Mandatory**

WS-DOM-TAG-NAME **Output** **Mandatory**

The tag name of the attribute will be returned here.

WS-DOM-VALUE-PTR **Output** **Mandatory**

This field will contain the attribute's address.

WS-DOM-VALUE-LENGTH **Output** **Mandatory**

This field will contain the length of the text value of the attribute.

WS-DOM-PARENT-NODE-ID **Output** **Mandatory**

After a successful call, the SOLA DOM API will populate this variable with the parent element node identifier.

WS-DOM-ATTRIBUTE-ID **Input** **Mandatory**

This field contains the unique node identifier of an attribute. The SOLA DOM API uses this value to retrieve the attribute details. This variable is a redefinition of WS-DOM-NODE-ID.

The following return codes are possible:

RC Value	Meaning
0	Successful - No messages issued.
+8	Failed. The following are the possible values for WS-DOM-MSG field: Specified attribute node identifier doesn't exist Input error Possible input errors could be: The function specified is not supported
+12	SOLA DOM API abended. Abend code - XXXX (four character code returned at run time).
+16	WS-DOM-HANDLE doesn't contain a valid handle.

Example

In this example, an attribute that had its unique node identifier saved in SAVED-NODE-ID is being retrieved.

```
SET  WS-DOM-GET-ATTRIBUTE-ADR-ID TO TRUE
MOVE SAVED-NODE-ID            TO WS-DOM-NODE-ID

   CALL WS-DOM-API USING WS-DOM-RC,
             WS-DOM-MSG,
             WS-DOM-HANDLE,
             WS-DOM-FUNCTION,
             WS-DOM-PLACE-HOLDER,
             WS-DOM-TAG-NAME,
             WS-DOM-VALUE-PTR,
             WS-DOM-VALUE-LENGTH,
             WS-DOM-PARENT-NODE-ID,
             WS-DOM-ATTRIBUTE-ID
```

getAttributeArray

CICS BATCH

The getAttributeArray function is used to retrieve an array containing all the attributes for a given element node. To retrieve the value of all the attributes for a given element node, you retrieve the array by specifying the identifier (specified here as WS-DOM-PARENT-NODE-ID) for the parent element node.

```
CALL WS-DOM-API USING
      WS-DOM-RC,         /* Halfword. Return Code */
      WS-DOM-MSG,        /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,     /* Pointer.  For internal use */
      WS-DOM-FUNCTION,   /* Char(25).  The name of the API */
      WS-DOM-PARENT-NODE-ID, /* ID of the parent element */
      WS-DOM-ARRAY            /* Structure to receive the results */
```

The SOLA DOM API expects the format of the variable WS-DOM-ARRAY to be a variable length table of lengths and addresses. The following code fragment defines the format of WS-DOM-ARRAY. The number of occurrences (shown here as 100) should be set to a value that is sufficient to contain the maximum number of attributes you expect to receive back from the function.

```
01  WS-DOM-ARRAY
    05 WS-MAX-COUNT-ALLOWED       PIC S9(4) COMP.
    05 WS-ACTUAL-NODE-COUNT       PIC S9(4) COMP.
    05 WS-NODE-ARRAY OCCURS       1 TO 100 TIMES
               DEPENDING ON WS-MAX-COUNT-ALLOWED.
               10  WS-NODE-NAME-LEN     PIC S9(4) COMP.
               10  WS-NODE-NAME-ADR     USAGE IS POINTER.
               10  WS-NODE-VALUE-LEN    PIC S9(4) COMP.
               10  WS-NODE-VALUE-ADR    USAGE IS POINTER.
```

If the number of attributes exceeds the WS-MAX-COUNT-ALLOWED value, then the number of attributes returned will be limited to WS-MAX-COUNT-ALLOWED. WS-ACTUAL-NODE-COUNT will contain the actual count of the attributes available to be returned. If WS-MAX-COUNT-ALLOWED is set to a larger size than the number of occurrences of the table then corruption of storage beyond the table may occur.

The getAttributeArray function can be used in CICS and batch.

Variables:

WS-DOM-RC **Output** **Mandatory**

Returns status.

WS-DOM-MSG **Output** **Mandatory**

When the return code is not zero, this field will contain messages.

WS-DOM-HANDLE **Input** **Mandatory**

For internal use. The DOM handle field uniquely identifies the XML DOM tree from which the attribute array will be retrieved.

WS-DOM-FUNCTION **Input** **Mandatory**

Must be set to **WS-DOM-GET-ATTRIBUTE-ARRAY**.

WS-DOM-PARENT-NODE-ID **Input** **Mandatory**

The SOLA DOM API will retrieve all the attributes for given parent element node identifier. Please note that a parent node is always an element node.

WS-DOM-ARRAY **Output** **Mandatory**

The structure of this variable is described in the examples section.

The following return codes are possible:

RC Value	Meaning
0	Successful - No messages issued.
+4	No attribute exists for the element - Messages were issued in field WS-DOM-MSG
+8	Failed. The following are the possible values for WS-DOM-MSG field: • Specified parent node identifier doesn't exist • Input error Possible input errors could be: • The function specified is not supported.
+12	SOLA DOM API abended. Abend code - XXXX (four character code returned at run time).
+16	WS-DOM-HANDLE doesn't contain a valid handle.

Example

In this example we are retrieving all of the attributes of an element that had its unique node identifier saved in SAVED-PARENT-NODE-ID. We are specifying that a maximum of 100 attributes be returned.

```
01  WS-DOM-ARRAY
     05 WS-MAX-COUNT-ALLOWED       PIC S9(4) COMP.
     05 WS-ACTUAL-NODE-COUNT     PIC S9(4) COMP.
     05 WS-NODE-ARRAY OCCURS  1 TO 100 TIMES
                DEPENDING ON WS-MAX-COUNT-ALLOWED.
                10  WS-NODE-NAME-LEN     PIC S9(4) COMP.
                10  WS-NODE-NAME-ADR     USAGE IS POINTER.
```

```
                10  WS-NODE-VALUE-LEN    PIC S9(4) COMP.
                10  WS-NODE-VALUE-ADR    USAGE IS POINTER.

SET  WS-DOM-GET-ATTRIBUTE-ARRAY TO TRUE
MOVE SAVED-PARENT-NODE-ID       TO WS-DOM-PARENT-NODE-ID
MOVE 100                        TO WS-MAX-COUNT-ALLOWED

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PARENT-NODE-ID,
                      WS-DOM-ARRAY
```

getAttributeById

CICS BATCH

The getAttributeById function retrieves an attribute and its value using a given node Id. The node identifier (shown below as WS-DOM-NODE-ID) is the unique node identifier for the attribute node. It is used to identify and locate the node in the tree.

The getAttributeById function can be used in CICS and batch.

```
CALL WS-DOM-API USING
      WS-DOM-RC,         /* Halfword. Return Code */
      WS-DOM-MSG,        /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,     /* Pointer.  For internal use */
      WS-DOM-FUNCTION, /* Char(25).  The name of the API */
      WS-DOM-PLACE-HOLDER,            /* placeholder */
      WS-DOM-TAG-NAME,    /* The tag name is returned here */
      WS-DOM-VALUE,       /* The text node is returned here */
      WS-DOM-VALUE-LENGTH, /* Length of the returned text node */
      WS-DOM-PARENT-NODE-ID, /* The ID of the parent of the node */
      WS-DOM-ATTRIBUTE-ID,   /* The ID of the node to be retrieved */
      WS-DOM-PLACE-HOLDER,             /* Placeholder */
      WS-DOM-PLACE-HOLDER,             /* Placeholder */
      WS-DOM-PLACE-HOLDER,             /* Placeholder */
      WS-DOM-MAX-LENGTH /* Can specify max length of data here */
```

Variables:

WS-DOM-RC **Output** **Mandatory**

Returns status.

WS-DOM-MSG **Output** **Mandatory**

When the return code is not zero, this field will contain messages.

WS-DOM-HANDLE **Input** **Mandatory**

For internal use. The DOM handle field uniquely identifies the XML DOM tree from which the attribute will be retrieved.

WS-DOM-FUNCTION **Input** **Mandatory**

Must be set to **WS-DOM-GET-ATTRIBUTE-ID**.

WS-DOM-PLACE-HOLDER **Place holder** **Mandatory**

WS-DOM-TAG-NAME **Output** **Mandatory**

The tag name of the attribute will be returned here.

WS-DOM-VALUE **Output** **Mandatory**

This field will contain the normalized text value of the attribute.

WS-DOM-VALUE-LENGTH **Output** **Mandatory**

This field will contain the length of the text value of the attribute.

WS-DOM-PARENT-NODE-ID **Output** **Mandatory**

After a successful call, the SOLA DOM API will populate this variable with the parent element node identifier.

WS-DOM-ATTRIBUTE-ID **Input** **Mandatory**

This field contains the unique node identifier of an attribute. The SOLA DOM API uses this value to retrieve the attribute details. This variable is a redefinition of WS-DOM-NODE-ID.

WS-DOM-PLACE-HOLDER **Place Holder** **Optional**

WS-DOM-PLACE-HOLDER **Place Holder** **Optional**

WS-DOM-PLACE-HOLDER **Place Holder** **Optional**

WS-DOM-MAX-LENGTH **Input** **Optional**

This optional field, when specified, tells the SOLA DOM API to truncate the data in WS-DOM-VALUE field if WS-DOM-VALUE-LENGTH exceeds WS-DOM-MAX-LENGTH. A return code of +5 is returned if the returned data is truncated. In this case, WS-DOM-VALUE-LENGTH contains the full length, but data in WS-DOM-VALUE is truncated to WS-DOM-MAX-LENGTH.

The following return codes are possible:

RC Value	Meaning
0	Successful - No messages issued.
+5	Returned data is bigger than allowed. Data truncated to the specified Length.
+8	Failed. The following are the possible values for WS-DOM-MSG field: • Specified attribute node identifier doesn't exist • Input error Possible input errors could be: • The function specified is not supported
+12	SOLA DOM API abended. Abend code - XXXX (four character code returned at run time).
+16	WS-DOM-HANDLE doesn't contain a valid handle.

Example

In this example, an attribute that had its unique node identifier saved in SAVED-NODE-ID is being retrieved.

```
SET  WS-DOM-GET-ATTRIBUTE-ID TO TRUE
MOVE SAVED-NODE-ID           TO WS-DOM-NODE-ID
```

```
CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE,
                      WS-DOM-VALUE-LENGTH,
                      WS-DOM-PARENT-NODE-ID,
                      WS-DOM-ATTRIBUTE-ID
```

In this next example, an attribute that had its unique node identifier saved in SAVED-NODE-ID is being retrieved. We are also telling the SOLA DOM API to truncate the attribute's text node value should the length of the value exceed 255 bytes.

```
SET  WS-DOM-GET-ATTRIBUTE-ID TO TRUE
MOVE SAVED-NODE-ID           TO WS-DOM-NODE-ID
MOVE +255                    TO WS-DOM-MAX-LENGTH

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE,
                      WS-DOM-VALUE-LENGTH,
                      WS-DOM-PARENT-NODE-ID,
                      WS-DOM-NODE-ID,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-PLACE-HOLDER,
                     WS-DOM-PLACE-HOLDER,
                      WS-DOM-MAX-LENGTH
```

getElementAddress

CICS BATCH

The getElementAddress function is identical to getElementByTagName, except that the element value's address is returned in place of the element value. This can be useful if you don't want to provide storage for the element value, or where you want to use the XMLDOMWS copybook variables and the element value is greater than 256 bytes.

The getElementAddress function retrieves the address of an element node by its tag name. If multiple instances of the element exist in the XML document, the instance of the element corresponding to a sequence number will be returned.

The default scope of the element tag name search is the entire XML document. This default search scope can be overridden in two ways:

- First, by specifying a parent node identifier on input to the request. The search will then be limited to finding a child element with a matching tag name for the uniquely specified parent node.
- Second, by specifying a parent tag name instead of a sequence number or a parent node identifier. This second search scope will find the first instance of an element node with a matching tag name and parent tag name combination.

The precedence of search criteria is as follows: parent node identifier has precedence over parent tag name. The parent tag name and element sequence number are mutually exclusive.

The getElementAddress function can be used in CICS and batch.

```
CALL WS-DOM-API USING
      WS-DOM-RC,         /* Halfword. Return Code */
      WS-DOM-MSG,        /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,     /* Pointer.  For internal use */
      WS-DOM-FUNCTION,   /* Char(25).  The name of the API */
      WS-DOM-SEQ/WS-DOM-PARENT,/* Sequence Number/Parent */
      WS-DOM-TAG-NAME,   /* Name of the tag to search for */
      WS-DOM-VALUE-PTR,     /* The address will be returned here */
      WS-DOM-VALUE-LENGTH,  /* Length of the returned text node */
      WS-DOM-PARENT-NODE-ID, /* The ID of the parent of the node */
      WS-DOM-ELEMENT-ID,/* The ID of the node will be returned here */
      WS-DOM-NMS-NODE-ID,    /* Unique id of the namespace node */
      WS-DOM-CHILD-ELEM-COUNT, /* Num of children returned here */
      WS-DOM-ATTR-NODE-COUNT   /* Num of attributes returned here */
```

Variables:

WS-DOM-RC **Output** **Mandatory**
Returns status.

WS-DOM-MSG **Output** **Mandatory**
When the return code is not zero, this field will contain messages.

WS-DOM-HANDLE **Input** **Mandatory**
For internal use. The DOM handle field uniquely identifies the XML DOM tree from which the element will be retrieved.

WS-DOM-FUNCTION **Input** **Mandatory**
Must be set to **WS-DOM-GET-ELEMENT-ADR**.

WS-DOM-SEQ/WS-DOM-PARENT **Input** **Mandatory**
This field can contain either a tag name or a sequence number. WS-DOM-SEQ or WS-DOM-PARENT can be used. When it contains a sequence number it instructs the SOLA DOM API to retrieve the address of a particular instance of an element where WS-DOM-TAG-NAME matches a tag name in the tree. When it contains a tag name the DOM-API tries to retrieve the address of an element for which the parent name and tag names match.

WS-DOM-TAG-NAME **Input** **Mandatory**
This field instructs the SOLA DOM API to search for an element with a matching tag name.

WS-DOM-VALUE-PTR **Output** **Mandatory**
The address of the text value for the element will be returned here.

WS-DOM-VALUE-LENGTH **Output** **Mandatory**
When an element has a text node attached, this field will contain the length of the text node.

WS-DOM-PARENT-NODE-ID **Input/Output** **Optional**
This field is optional on input. When specified in the CALL with a value other than null, the SOLA DOM API will search for a child element with a matching tag name for this given parent node identifier. On output, after the call, this field will contain the node identifier of the parent.

WS-DOM-ELEMENT-ID **Output** **Optional**
The SOLA DOM API populates this variable with the unique node identifier of the element found. This variable is a redefinition of WS-DOM-NODE-ID.

WS-DOM-NMS-NODE-ID **Output** **Optional**
This is the unique node identifier for a namespace. If there is a namespace for the node, the unique node idenifier of the namespace will be returned in this variable. A null value returned in this variable indicates no namespace.

WS-DOM-CHILD-ELEM-COUNT **Output** **Optional**

If the element node has child elements (a Complex Type), the SOLA DOM API populates this field with the child element node count.

WS-DOM-ATTR-NODE-COUNT **Output** **Optional**

If the element node has attribute nodes, the SOLA DOM API populates this field with the attribute node count.

The following return codes are possible:

RC Value	Meaning
0	Successful - No messages issued.
+4	Element Not Found Messages were issued in field WS-DOM-MSG
+8	Failed. The following are the possible values for WS-DOM-MSG field: • Seq too high (higher than total number of element nodes in the document) • Specified parent node identifier doesn't exist • Input error Possible input errors could be: • The function specified is not supported. • Element tag name not specified.
+12	SOLA DOM API abended. Abend code - XXXX (four character code returned at run time).
+16	WS-DOM-HANDLE doesn't contain a valid handle.

Example

In this example an element named 'Bossid' is being searched for. WS-DOM-SEQ is set to +3 because we are want to get the third occurrence of the Bossid element.

```
SET  WS-DOM-GET-ELEMENT-ADR TO TRUE
MOVE 'BossId'               TO WS-DOM-TAG-NAME
MOVE +3                     TO WS-DOM-SEQ

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-SEQ,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE-PTR,
                      WS-DOM-VALUE-LENGTH
```

In this next example, we are searching for an element named 'Bossid'. We are limiting the search scope to within the parent 'NameSearch'. If there are multiple instances of the parent 'NameSearch', the *first* such instance will be selected.

```
SET  WS-DOM-GET-ELEMENT-ADR TO TRUE
MOVE 'BossId'            TO WS-DOM-TAG-NAME
MOVE 'NameSearch'        TO WS-DOM-PARENT

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PARENT,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE-PTR,
                      WS-DOM-VALUE-LENGTH
```

In this next example, we are searching for an element named 'Bossid'. We are limiting the search scope to within the parent that has a parent unique node identifier that was saved in SAVED-PARENT-ID.

```
SET  WS-DOM-GET-ELEMENT-ADR TO TRUE
MOVE 'BossId'               TO WS-DOM-TAG-NAME
MOVE SAVED-PARENT-ID        TO WS-DOM-PARENT-NODE-ID

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE-PTR,
                      WS-DOM-VALUE-LENGTH,
                      WS-DOM-PARENT-NODE-ID
```

getElementAddressById

CICS BATCH

The getElementAddressById function is identical to getElementById, except that the element value address is returned in place of the element value. This can be useful if you don't want to provide storage for the element value, or where you want to use the XMLDOMWS copybook variables and the element value is greater than 256 bytes.

The getElementAddressById function retrieves the address of an element node using a given node Id. The node identifier (shown below as WS-DOM-NODE-ID) is the unique node identifier for the node. It is used to identify and locate the node in the tree.

The getElementAddressById function can be used in CICS and batch.

```
CALL WS-DOM-API USING
      WS-DOM-RC,         /* Halfword. Return Code */
      WS-DOM-MSG,        /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,     /* Pointer.  For internal use */
      WS-DOM-FUNCTION,   /* Char(25).  The name of the API */
      WS-DOM-PLACE-HOLDER, /* placeholder */
      WS-DOM-TAG-NAME,   /* Element tag name will be returned here */
      WS-DOM-VALUE-PTR,     /* The text node will be returned here */
      WS-DOM-VALUE-LENGTH,  /* Length of the returned text node */
      WS-DOM-PARENT-NODE-ID,  /* Parent ID returned here */
      WS-DOM-ELEMENT-ID,    /* The ID of the node to be searched for */
      WS-DOM-NMS-NODE-ID,    /* Unique id of the namespace node */
      WS-DOM-CHILD-ELEM-COUNT,  /* Num of children returned here */
      WS-DOM-ATTR-NODE-COUNT   /* Num of attributes returned here */
```

Variables:

WS-DOM-RC **Output** **Mandatory**

Returns status.

WS-DOM-MSG **Output** **Mandatory**

When the return code is not zero, this field will contain messages.

WS-DOM-HANDLE **Input** **Mandatory**

For internal use. The DOM handle field uniquely identifies the XML DOM tree from which the element will be retrieved.

WS-DOM-FUNCTION **Input** **Mandatory**

Must be set to **WS-DOM-GET-ELEMENT-ADR-ID**.

WS-DOM-PLACE-HOLDER **Place holder** **Mandatory**

WS-DOM-TAG-NAME **Output** **Mandatory**

The element tag name will be returned in this field.

WS-DOM-VALUE-PTR **Output** **Mandatory**

This field will contain the element's address.

WS-DOM-VALUE-LENGTH **Output** **Mandatory**

When an element has a text node attached, this field will contain the length of the text node.

WS-DOM-PARENT-NODE-ID **Output** **Mandatory**

After the call, this field will contain the node identifier of the parent.

WS-DOM-ELEMENT-ID **Input** **Mandatory**

This field contains the unique node identifier of the element that the SOLA DOM API is going to retrieve. This variable is a redefinition of WS-DOM-NODE-ID.

WS-DOM-NMS-NODE-ID **Output** **Optional**

This is the unique node identifier for a namespace. If there is a namespace for the node, the unique node identifier of the namespace will be returned in this variable. A null value returned in this variable indicates no namespace or default namespace.

WS-DOM-CHILD-ELEM-COUNT **Output** **Optional**

If the element node has child elements (a Complex Type), the SOLA DOM API populates this field with the child element node count.

WS-DOM-ATTR-NODE-COUNT **Output** **Optional**

If the element node has attribute nodes, the SOLA DOM API populates this field with the attribute node count.

The following return codes are possible:

RC Value	Meaning
0	Successful - No messages issued.
+8	Failed. The following are the possible values for WS-DOM-MSG field: • Input error • Node identifier doesn't exist Possible input errors could be: • The function specified is not supported.
+12	SOLA DOM API abended. Abend code - XXXX (four character code returned at run time).
+16	WS-DOM-HANDLE doesn't contain a valid handle.

Example

In this example, the address of an element which had its node identifier saved in SAVED-NODE-ID is being retrieved. At successful completion of the call the node identifier of the parent node will be returned in WS-DOM-PARENT-NODE-ID.

If the element has no value, the WS-DOM-VALUE-LENGTH will be zero and WS-DOM-VALUE-PTR will be null. It is recommended that WS-DOM-VALUE-LENGTH be verified before using the value pointer.

```
SET  WS-DOM-GET-ELEMENT-ADR-ID TO TRUE
MOVE SAVED-NODE-ID             TO WS-DOM-NODE-ID

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE-PTR,
                      WS-DOM-VALUE-LENGTH,
                      WS-DOM-PARENT-NODE-ID,
                      WS-DOM-NODE-ID
```

In this example the address of an element which had its node identifier saved in SAVED-NODE-ID is being retrieved. At successful completion of the call the node identifier of the parent node will be returned in WS-DOM-PARENT-NODE-ID along with the node identifier of the element's namespace (if present), the number of child elements, the unique node identifier of the element's namespace (if any) and number of attributes belonging to the element.

```
SET  WS-DOM-GET-ELEMENT-ADR-ID TO TRUE
MOVE SAVED-NODE-ID             TO WS-DOM-NODE-ID

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE-PTR,
                      WS-DOM-VALUE-LENGTH,
                      WS-DOM-PARENT-NODE-ID,
                      WS-DOM-NODE-ID,
                      WS-DOM-NMS-NODE-ID,
                      WS-DOM-CHILD-ELEM-COUNT,
                      WS-DOM-ATTR-NODE-COUNT
```

getElementById

CICS BATCH

The getElementById function retrieves an element and its child text node using a given node Id. The node identifier (shown below as WS-DOM-NODE-ID) is the unique node identifier for the node. It is used to identify and locate the node in the tree.

The getElementById function can be used in CICS and batch.

```
CALL WS-DOM-API USING
      WS-DOM-RC,         /* Halfword. Return Code */
      WS-DOM-MSG,        /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,     /* Pointer.  For internal use */
      WS-DOM-FUNCTION,   /* Char(25).  The name of the API */
      WS-DOM-PLACE-HOLDER, /* placeholder */
      WS-DOM-TAG-NAME,   /* Element tag name will be returned here */
      WS-DOM-VALUE,      /* The text node will be returned here */
      WS-DOM-VALUE-LENGTH, /* Length of the returned text node */
      WS-DOM-PARENT-NODE-ID, /* Parent ID returned here */
      WS-DOM-ELEMENT-ID,    /* The ID of the node to be searched for */
      WS-DOM-NMS-NODE-ID /* Unique ID for the namespace node */
      WS-DOM-CHILD-ELEM-COUNT, /* Num of children returned here */
      WS-DOM-ATTR-NODE-COUNT, /* Num of attributes returned here */
      WS-DOM-MAX-LENGTH /* Can specify max length of data here */
```

Variables:

WS-DOM-RC **Output** **Mandatory**

Returns status.

WS-DOM-MSG **Output** **Mandatory**

When the return code is not zero, this field will contain messages.

WS-DOM-HANDLE **Input** **Mandatory**

For internal use. The DOM handle field uniquely identifies the XML DOM tree from which the element will be retrieved.

WS-DOM-FUNCTION **Input** **Mandatory**

Must be set to **WS-DOM-GET-ELEMENT-ID**.

WS-DOM-PLACE-HOLDER **Place holder** **Mandatory**

WS-DOM-TAG-NAME **Output** **Mandatory**

The element tag name will be returned in this field.

WS-DOM-VALUE **Output** **Mandatory**

When an element has a text node attached, this field will have a normalized text value.

WS-DOM-VALUE-LENGTH **Output** **Mandatory**

When an element has a text node attached, this field will contain the length of the text node.

WS-DOM-PARENT-NODE-ID **Output** **Mandatory**

After the call, this field will contain the node identifier of the parent.

WS-DOM-ELEMENT-ID **Input** **Mandatory**

This field contains the unique node identifier of the element that the SOLA DOM API is going to retrieve. This variable is a redefinition of WS-DOM-NODE-ID.

WS-DOM-NMS-NODE-ID **Output** **Optional**

This is the unique node identifier for a namespace. If there is a namespace for the node, the unique node identifier of the namespace will be returned in this variable. A null value returned in this variable indicates no namespace.

WS-DOM-CHILD-ELEM-COUNT **Output** **Optional**

If the element node has child elements (a Complex Type), the SOLA DOM API populates this field with the child element node count.

WS-DOM-ATTR-NODE-COUNT **Output** **Optional**

If the element node has attribute nodes, the SOLA DOM API populates this field with the attribute node count.

WS-DOM-MAX-LENGTH **Input** **Optional**

This optional field, when specified, tells SOLA DOM API to truncate data in WS-DOM-VALUE field if WS-DOM-VALUE-LENGTH exceeds WS-DOM-MAX-LENGTH. A return code of +5 is returned if the returned data is truncated. In such case, WS-DOM-VALUE-LENGTH still contains the full length but data in WS-DOM-VALUE is truncated to WS-DOM-MAX-LENGTH.

The following return codes are possible:

RC Value	Meaning
0	Successful - No messages issued.
+5	Returned data is bigger than allowed. Data truncated to the specified Length.
+8	Failed. The following are the possible values for WS-DOM-MSG field: • Input error • Node identifier doesn't exist Possible input errors could be: • The function specified is not supported.

RC Value	Meaning
+12	SOLA DOM API abended. Abend code - XXXX (four character code returned at run time).
+16	WS-DOM-HANDLE doesn't contain a valid handle.

Example

In this example, an element which had its node identifier saved in SAVED-NODE-ID is being retrieved. At successful completion of the call, the node identifier of the parent node will be returned in WS-DOM-PARENT-NODE-ID. If the element has a text node, the WS-DOM-VALUE will be populated with the value of the text node, and WS-DOM-VALUE-LENGTH will be populated with the length of the text. If the element has no text, the WS-DOM-VALUE-LENGTH will be null and the WS-DOM-VALUE will contain spaces.

```
SET  WS-DOM-GET-ELEMENT-ID TO TRUE
MOVE SAVED-NODE-ID         TO WS-DOM-NODE-ID

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE,
                      WS-DOM-VALUE-LENGTH,
                      WS-DOM-PARENT-NODE-ID,
                      WS-DOM-ELEMENT-ID
```

In this example an element which had its node identifier saved in SAVED-NODE-ID is being retrieved. At successful completion of the call the node identifier of the parent node will be returned in WS-DOM-PARENT-NODE-ID along with the node identifier of the element's namespace (if present), the number of child elements, the unique node identifier of the element's namespace (if any) and number of attributes belonging to the element. We are also telling the SOLA DOM API to truncate the element's text node value should the length of the value exceed 255 bytes.

```
SET  WS-DOM-GET-ELEMENT-ID TO TRUE
MOVE SAVED-NODE-ID         TO WS-DOM-NODE-ID
MOVE +255                  TO WS-DOM-MAX-LENGTH

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE,
```

```
WS-DOM-VALUE-LENGTH,
WS-DOM-PARENT-NODE-ID,
WS-DOM-NODE-ID
WS-DOM-NMS-NODE-ID,
WS-DOM-CHILD-ELEM-COUNT,
WS-DOM-ATTR-NODE-COUNT,
WS-DOM-MAX-LENGTH
```

getElementByTagName

CICS BATCH

The getElementByTagName function retrieves an element node by its tag name. If multiple instances of the element exist in the XML document, the instance of the element corresponding to a sequence number will be returned.

The default scope of the element tag name search is the entire XML document. This default search scope can be overridden in two ways:

- First, by specifying a parent node identifier and a sequence number on input to the request. The search will then be limited to finding a child element with a matching tag name for the uniquely specified parent node. The sequence number further limits the search by specifying which occurrence of the element with a matching tag name is returned. If the sequence number specified is greater than the number of matching elements, an error will be returned.

- Second, by specifying a parent tag name instead of a sequence number or a parent node identifier. This second search scope will find the first instance of an element node with a matching tag name and parent tag name combination.

The precedence of search criteria is as follows: parent node identifier has precedence over parent tag name. The parent tag name and element sequence number are mutually exclusive.

The getElementByTagName function can be used in CICS and batch.

```
CALL WS-DOM-API USING
      WS-DOM-RC,         /* Halfword. Return Code */
      WS-DOM-MSG,        /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,     /* Pointer.  For internal use */
      WS-DOM-FUNCTION, /* Char(25).  The name of the API */
      WS-DOM-SEQ/WS-DOM-PARENT,/* Sequence Number/Parent */
      WS-DOM-TAG-NAME, /* Name of the tag to search for */
      WS-DOM-VALUE,      /* The text node will be returned here */
      WS-DOM-VALUE-LENGTH, /* Length of the returned text node */
      WS-DOM-PARENT-NODE-ID, /* The ID of the parent of the node */
      WS-DOM-ELEMENT-ID,/* The ID of the node will be returned here */
      WS-DOM-NMS-NODE-ID,    /* Unique id of the namespace node */
      WS-DOM-CHILD-ELEM-COUNT, /*Num of child elements returned here*/
      WS-DOM-ATTR-NODE-COUNT, /* Num of attributes returned here */
      WS-DOM-MAX-LENGTH /* Can specify max length of data here */
```

Variables:

WS-DOM-RC	Output	Mandatory

Returns status.

WS-DOM-MSG **Output** **Mandatory**
When the return code is not zero, this field will contain messages.

WS-DOM-HANDLE **Input** **Mandatory**
For internal use. The DOM handle field uniquely identifies the XML DOM tree from which the element will be retrieved.

WS-DOM-FUNCTION **Input** **Mandatory**
Must be set to **WS-DOM-GET-ELEMENT-BYTAG**.

WS-DOM-SEQ/WS-DOM-PARENT **Input** **Mandatory**
This field can contain either a tag name or a sequence number. WS-DOM-SEQ or WS-DOM-PARENT can be used. When it contains a sequence number it instructs the SOLA DOM API to retrieve a particular instance of an element where WS-DOM-TAG-NAME matches a tag name in the tree. When it contains a parent tag name, the DOM-API tries to retrieve an element for which the parent name and tag names match.

WS-DOM-TAG-NAME **Input** **Mandatory**
This field instructs the SOLA DOM API to search for an element with a matching tag name.

WS-DOM-VALUE **Output** **Mandatory**
When an element has a text node attached, this field will contain the normalized text value.

WS-DOM-VALUE-LENGTH **Output** **Mandatory**
When an element has a text node attached, this field will contain the length of the text node.

WS-DOM-PARENT-NODE-ID **Input/Output** **Optional**
This field is optional on input. When specified in the CALL with a value other than zero, the SOLA DOM API will search for a child element with a matching tag name for this given parent node identifier. On output, after the call, this field will contain the node identifier of the parent.

WS-DOM-ELEMENT-ID **Output** **Optional**
The SOLA DOM API populates this variable with the unique node identifier of the element found. This variable is a redefinition of WS-DOM-NODE-ID.

WS-DOM-NMS-NODE-ID **Output** **Optional**
This is the unique node identifier for a namespace. If there is a namespace for the node, the unique node identifier of the namespace will be returned in this variable. A null value returned in this variable indicates no namespace.

WS-DOM-CHILD-ELEM-COUNT **Output** **Optional**

If the element node has child elements (a Complex Type), the SOLA DOM API populates this field with the child element node count.

WS-DOM-ATTR-NODE-COUNT **Output** **Optional**

If the element node has attribute nodes, the SOLA DOM API populates this field with the attribute node count.

WS-DOM-MAX-LENGTH **Input** **Optional**

This optional field, when specified, tells SOLA DOM API to truncate data in WS-DOM-VALUE field if WS-DOM-VALUE-LENGTH exceeds WS-DOM-MAX-LENGTH. A return code of +5 is returned if the returned data is truncated. In such case, WS-DOM-VALUE-LENGTH still contains the full length but data in WS-DOM-VALUE is truncated to WS-DOM-MAX-LENGTH.

The following return codes are possible:

RC Value	Meaning
0	Successful - No messages issued.
+4	Element Not Found Messages were issued in field WS-DOM-MSG
+5	Returned data is bigger than allowed. Data truncated to the specified Length.
+8	Failed. The following are the possible values for WS-DOM-MSG field: • Seq too high (higher than total number of element nodes in the document) • Specified parent node identifier doesn't exist • Input error Possible input errors could be: • The function specified is not supported. • Element tag name not specified.
+12	SOLA DOM API abended. Abend code - XXXX (four character code returned at run time).
+16	WS-DOM-HANDLE doesn't contain a valid handle.

Example

In this example an element named 'Bossid' is being searched for. WS-DOM-SEQ is set to +3 because we want to get the third occurrence of the Bossid element.

```
SET  WS-DOM-GET-ELEMENT-BYTAG TO TRUE
MOVE 'BossId'             TO WS-DOM-TAG-NAME
MOVE +3                   TO WS-DOM-SEQ
```

```
CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-SEQ,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE,
                      WS-DOM-VALUE-LENGTH
```

In this next example, we are searching for an element named 'Bossid'. We are limiting the search scope to within the parent 'NameSearch'. If there are multiple instances of the parent 'NameSearch', the *first* such instance is selected.

```
SET  WS-DOM-GET-ELEMENT-BYTAG
                        TO TRUE
MOVE 'BossId'           TO WS-DOM-TAG-NAME
MOVE 'NameSearch'       TO WS-DOM-PARENT

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PARENT,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE,
                      WS-DOM-VALUE-LENGTH
```

In this next example we are searching for an element named 'Bossid'. We are limiting the search scope to within the parent that has a parent unique node identifier that was saved in SAVED-PARENT-ID. When searching for an element under a specific parent node, a sequence number must be specified using WS-DOM-SEQ.

```
SET  WS-DOM-GET-ELEMENT-BYTAG TO TRUE
MOVE 'BossId'           TO WS-DOM-TAG-NAME
MOVE SAVED-PARENT-ID    TO WS-DOM-PARENT-NODE-ID
MOVE +1                 TO WS-DOM-SEQ


CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-SEQ,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE,
                      WS-DOM-VALUE-LENGTH,
                      WS-DOM-PARENT-NODE-ID
```

In this next example we are searching for an element named 'Bossid'. We are limiting the search scope to within the parent that has a parent unique node identifier that was saved in SAVED-PARENT-ID. We are requesting

that the unique node identifier of the element be returned, along with the number of child elements, the unique node identifier of the element's namespace (if any), and number of attributes belonging to the element. We are also telling the SOLA DOM API to truncate the element's text node value should the length of the value exceed 255 bytes.

```
SET  WS-DOM-GET-ELEMENT-BYTAG TO TRUE
MOVE 'BossId'             TO WS-DOM-TAG-NAME
MOVE SAVED-PARENT-ID      TO WS-DOM-PARENT-NODE-ID
MOVE +255                 TO WS-DOM-MAX-LENGTH

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE,
                      WS-DOM-VALUE-LENGTH,
                      WS-DOM-PARENT-NODE-ID,
                      WS-DOM-ELEMENT-ID,
                      WS-DOM-NMS-NODE-ID,
                      WS-DOM-CHILD-ELEM-COUNT,
                     WS-DOM-ATTR-NODE-COUNT,
                      WS-DOM-MAX-LENGTH
```

getElementByXPath

CICS BATCH

The getElementByXPath function retrieves an element and its child text node for a given XPath expression. Additionally, a sequence can be specified to get a particular occurrence of the element node.

Simple and limited XPath expressions are allowed in the WS-DOM-TAG-NAME when retrieving elements using getElementByXPath.

Below are two examples of an element search using an allowed XPath expression.

```
MOVE +1                          TO WS-DOM-SEQ
MOVE 'list/member'               TO WS-DOM-TAG-NAME
```

The above expression would retrieve the first element whose name is "member" and has a parent named "list".

When specifying an XPath expression the SOLA DOM API will look for the presence of "/" and "//" as the beginning characters of the search string. A double forward slash "//" as the leading character indicates that you want to search for the XPath expression anywhere within the document hierarchy. A single forward slash "/" as the leading character indicates that you are specifying an XPath expression that includes the root. If you don't specify "/" as the starting character in the XPath expression then SOLA DOM API will assume the specification of "//" as the starting characters.

```
MOVE '/book/chapter[5]/section[2]'          TO WS-DOM-TAG-NAME
```

The above expression would retrieve the second "section" element (the sequence is specified within braces []) whose parent is the fifth "chapter" element and the grand parent is an element named "book" (book is a root element and therefore "/" is specified).

Only local element names are allowed in the expression. Namespace aliases cannot be included during XPath searches in the current release.

A non-zero WS-DOM-SEQ value indicates that the specified occurrence of the XPath expression should be the one located. Alternatively, a bracketed occurrence inside the XPath will override any WS-DOM-SEQ specification. If neither a bracketed specification nor a sequence number (null value in WS-DOM-SEQ) is specified, the first matching element in the tree will be the one located. A leading slash indicates that a root element is specified. If the element following the leading slash is not a root element, a +8 error code will be returned

The getElementByXPath function can be used in CICS and batch.

```
CALL WS-DOM-API USING
      WS-DOM-RC,        /* Halfword. Return Code */
      WS-DOM-MSG,       /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,    /* Pointer.  For internal use */
      WS-DOM-FUNCTION,  /* Char(25).  The name of the API */
      WS-DOM-SEQ,       /* Occurrence of the tag within the document */
      WS-DOM-TAG-NAME,  /* XPath search expression */
      WS-DOM-VALUE,       /* Text node returned here */
      WS-DOM-VALUE-LENGTH, /* Length of the returned text node */
      WS-DOM-PARENT-NODE-ID, /* Unique id of the parent node*/
      WS-DOM-ELEMENT-ID,/* The ID of the node will be returned here */
      WS-DOM-NMS-NODE-ID /* Unique ID for the namespace node */
      WS-DOM-CHILD-ELEM-COUNT, /* Num of children returned here */
      WS-DOM-ATTR-NODE-COUNT, /* Num of attributes returned here */
      WS-DOM-MAX-LENGTH /* Can specify max length of data here */
```

Variables:

WS-DOM-RC **Output** **Mandatory**

Returns status.

WS-DOM-MSG **Output** **Mandatory**

When the return code is not zero, this field will contain messages.

WS-DOM-HANDLE **Input** **Mandatory**

For internal use. The DOM handle field uniquely identifies the XML DOM tree from which the element will be retrieved.

WS-DOM-FUNCTION **Input** **Mandatory**

Must be set to **WS-DOM-GET-ELEMENT-XPATH**.

WS-DOM-SEQ **Input** **Mandatory**

When non-zero, this field instructs the SOLA DOM API to retrieve a particular instance of an element that matches the XPath expression specified in the WS-DOM-TAG-NAME parameter. A sequence specified in the XPath expression using brackets ([and]) takes precedence over WS-DOM-SEQ.

WS-DOM-TAG-NAME **Input** **Mandatory**

The XPath expression describing the search criteria for the element to be retrieved is specified in the WS-DOM-TAG-NAME input parameter. The XPath expression must end with a space for delimiting purposes. Refer to the function's description for information regarding valid XPath expressions.

WS-DOM-VALUE **Output** **Mandatory**

When an element has a text node attached, this field will have a normalized text value.

WS-DOM-VALUE-LENGTH **Output** **Mandatory**

When an element has a text node attached, this field will contain the length of the text node.

WS-DOM-PARENT-NODE-ID **Output** **Optional**

After the call, this field will contain the node identifier of the parent.

WS-DOM-ELEMENT-ID **Output** **Optional**

The SOLA DOM API populates this variable with the unique node identifier of the element found. This variable is a redefinition of WS-DOM-NODE-ID.

WS-DOM-NMS-NODE-ID **Output** **Optional**

This is the unique node identifier for a namespace. If there is a namespace for the node, the unique node identifier of the namespace will be returned in this variable.

WS-DOM-CHILD-ELEM-COUNT **Output** **Optional**

If the element node has child elements (a ComplextType), the SOLA DOM API populates this field with the child element node count.

WS-DOM-ATTR-NODE-COUNT **Output** **Optional**

If the element node has attribute nodes, the SOLA DOM API populates this field with the attribute node count.

WS-DOM-MAX-LENGTH **Input** **Optional**

This optional field, when specified, tells SOLA DOM API to truncate data in WS-DOM-VALUE field if WS-DOM-VALUE-LENGTH exceeds WS-DOM-MAX-LENGTH. A return code of +5 is returned if the returned data is truncated. In such case, WS-DOM-VALUE-LENGTH still contains the full length but data in WS-DOM-VALUE is truncated to WS-DOM-MAX-LENGTH.

The following return codes are possible:

RC Value	Meaning
0	Successful - No messages issued.
+4	Element Not found - Messages were issued in field WS-DOM-MSG
+5	Returned data is bigger than allowed. Data truncated to the specified Length.
+8	Failed. The following are the possible values for WS-DOM-MSG field: • XPath starts with a "/" but root element name doesn't match • Seq too high (one of the sequence values is higher than the total number element nodes in the document) • Input error

	Possible input errors could be: • The function specified is not supported.
+12	SOLA DOM API abended. Abend code - XXXX (four character code returned at run time).
+16	WS-DOM-HANDLE doesn't contain a valid handle.

Example

In this example an element named "member" is being searched for. The element's parent is named "list". The first occurrence of the element is requested.

```
SET  WS-DOM-GET-ELEMENT-XPATH TO TRUE
MOVE +1                    TO WS-DOM-SEQ
MOVE 'list/member'         TO WS-DOM-TAG-NAME

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-SEQ,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE,
                      WS-DOM-VALUE-LENGTH
```

In this next example an element named "section" is being searched for. The element's parent is named "chapter". The parent of the element "chapter" is named "book". The element "book" is the root element of the document.

The fifth "chapter" within the root element "book" is sought, and within that instance the second "section" is required.

```
SET  WS-DOM-GET-ELEMENT-XPATH      TO TRUE
MOVE '/book/chapter[5]/section[2]' TO WS-DOM-TAG-NAME

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE,
                      WS-DOM-VALUE-LENGTH
```

In this next example the same element is being searched for, but this time, at successful completion of the call the node identifier of the parent node will be returned in WS-DOM-PARENT-NODE-ID along with the node identifier of the element's namespace (if present), the number of child elements, the unique node identifier of the element's namespace (if any)

and number of attributes belonging to the element. We are also telling the SOLA DOM API to truncate the element's text node value should the length of the value exceed 255 bytes.

```
SET  WS-DOM-GET-ELEMENT-XPATH       TO TRUE
MOVE '/book/chapter[5]/section[2]' TO WS-DOM-TAG-NAME
MOVE +255                          TO WS-DOM-MAX-LENGTH

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE,
                      WS-DOM-VALUE-LENGTH,
                      WS-DOM-PARENT-NODE-ID,
                      WS-DOM-ELEMENT-ID,
                      WS-DOM-NMS-NODE-ID,
                      WS-DOM-CHILD-ELEM-COUNT,
                      WS-DOM-ATTR-NODE-COUNT,
                      WS-DOM-MAX-LENGTH
```

getNamespace

CICS BATCH

The getNamespace function retrieves the value of a namespace associated with a namespace unique node identifier (WS-DOM-NMS-NODE-ID).

The getNamespace function can be used in CICS and batch.

```
CALL WS-DOM-API USING
      WS-DOM-RC,         /* Halfword. Return Code */
      WS-DOM-MSG,        /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,     /* Pointer.  For internal use */
      WS-DOM-FUNCTION,   /* Char(25).  The name of the API */
      WS-DOM-PLACE-HOLDER, /* placeholder */
      WS-DOM-NMS-ALIAS,    /* Namespace alias returned here */
      WS-DOM-NAMESPACE,    /* The namespace value returned here */
      WS-DOM-VALUE-LENGTH, /* Length of the returned value  */
      WS-DOM-PLACE-HOLDER  /* placeholder */
      WS-DOM-PLACE-HOLDER  /* placeholder */
      WS-DOM-NMS-NODE-ID  /* Unique ID for the namespace node */
```

Variables:

WS-DOM-RC **Output** **Mandatory**

Returns status.

WS-DOM-MSG **Output** **Mandatory**

When the return code is not zero, this field will contain messages.

WS-DOM-HANDLE **Input** **Mandatory**

For internal use. The DOM handle field uniquely identifies the XML DOM tree from which the namespace will be retrieved.

WS-DOM-FUNCTION **Input** **Mandatory**

Must be set to **WS-DOM-GET-NAMESPACE**.

WS-DOM-PLACE-HOLDER **Place Holder** **Mandatory**

WS-DOM-NMS-ALIAS **Output** **Mandatory**

The value of the namespace alias will be returned in this variable.

WS-DOM-NAMESPACE **Output** **Mandatory**

This field will contain the value of the namespace associated with the element node.

WS-DOM-VALUE-LENGTH **Output** **Mandatory**

This field will contain the length of the namespace associated with the element node.

WS-DOM-PLACE-HOLDER **Place Holder** **Mandatory**

WS-DOM-PLACE-HOLDER **Place Holder** **Mandatory**

WS-DOM-NMS-NODE-ID **Input** **Mandatory**

This is the unique node identifier for a namespace. Specify the unique node identifier of the namespace to be retrieved.

The following return codes are possible:

RC Value	Meaning
0	Successful - No messages issued.
+8	Failed. The following are the possible values for WS-DOM-MSG field: • Specified namespace node identifier doesn't exist • Input error Possible input errors could be: • The function specified is not supported.
+12	SOLA DOM API abended. Abend code - XXXX (four character code returned at run time).
+16	WS-DOM-HANDLE doesn't contain a valid handle.

Example

In this example we are retrieving the value of the namespace that had its unique node identifier saved in SAVED-NMS-NODE-ID.

```
SET  WS-DOM-GET-NAMESPACE     TO TRUE
MOVE SAVED-NMS-NODE-ID        TO WS-DOM-NMS-NODE-ID

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-NMS-ALIAS,
                      WS-DOM-NAMESPACE,
                      WS-DOM-VALUE-LENGTH,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-NMS-NODE-ID
```

parse

CICS BATCH

The parse function parses, normalizes and creates the tree structure for a given input XML document. The input XML document needs to be parsed before it can be processed with the SOLA DOM API.

The parse function can be used in CICS and batch.

```
CALL WS-DOM-API USING
      WS-DOM-RC,         /* Halfword. Return Code */
      WS-DOM-MSG,        /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,     /* Pointer.  For internal use */
      WS-DOM-FUNCTION,   /* Char(25).  The name of the API */
      WS-DOM-CONTROL,    /* Determines what happens to the input XML */
      WS-DOM-PLACE-HOLDER, /* placeholder */
      INPUT-XML,        /* The XML doc to be parsed */
      WS-DOM-VALUE-LENGTH /* Length of the XML doc */
```

Variables:

WS-DOM-RC **Output** **Mandatory**

Returns status.

WS-DOM-MSG **Output** **Mandatory**

When the return code is not zero, this field will contain messages.

WS-DOM-HANDLE **Input/Output** **Mandatory**

For internal use. The SOLA DOM API populates this variable with the unique node identifier for the newly parsed XML DOM tree that will be used in subsequent calls. The handles can be reused for multiple XML document parsing.

WS-DOM-FUNCTION **Input** **Mandatory**

Must be set to **WS-DOM-PARSE**.

WS-DOM-CONTROL **Input** **Mandatory**

WS-DOM-CONTROL determines what happens to the original input xml when the parse function is called. If set to 0 (default behavior), the original input XML will be destroyed to conserve memory. If set to 1, the original input XML will be preserved. Only 0 and 1 are valid values for WS-DOM-CONTROL when used with the parse function. A value of 0 does not add additional xml processing instructions.

WS-DOM-PLACE-HOLDER **Place Holder** **Mandatory**

INPUT-XML **Input** **Mandatory**

Place your input XML here.

WS-DOM-VALUE-LENGTH **Input** **Mandatory**

This field contains the length of the input XML.

The following return codes are possible:

RC Value	Meaning
0	Successful - No messages issued.
+8	Failed. The following are the possible values for WS-DOM-MSG field: • Invalid DOM Control Value • Input error Possible input errors could be: • The function specified is not supported. • Specified Length of input xml is zero • Parse failed due to XML Syntax error (specific detail appended). Detail example: Start/End tag mismatch at 00027, START TAG:- getPrice. (The number (00027) is the character position where the SOLA DOM API believes the start of the bad syntax is located.)
+12	SOLA DOM API abended. Abend code - XXXX (four character code returned at run time).
+16	WS-DOM-HANDLE doesn't contain a valid handle.

Example

In this example we are parsing the XML document contained in INPUT-XML. INPUT-XML will be destroyed by the PARSE function because we are setting WS-DOM-CONTROL to zero.

```
SET  WS-DOM-PARSE         TO TRUE
MOVE ZERO                 TO WS-DOM-CONTROL
MOVE LENGTH OF INPUT-XML TO WS-DOM-VALUE-LENGTH

CALL WS-DOM-API USING WS-DOM-RC,
          WS-DOM-MSG,
          WS-DOM-HANDLE,
          WS-DOM-FUNCTION,
          WS-DOM-CONTROL,
          WS-DOM-PLACE-HOLDER,
          INPUT-XML,
          WS-DOM-VALUE-LENGTH
```

resetSelectAttrNodes

CICS BATCH

This resetSelectAttrNodes function is used to reset the internal counters used by the selectAttrNodes function. The function should only be used in conjunction with the selectAttrNodes function when it is operating in an iterative mode. When the internal counter is reset and a parent node is specified using selectAttrNodes in iterative mode, the first child attribute node (belonging to the specified parent) will be selected. If a parent node is not specified, then the first attribute in the XML document will be selected.

The SOLA DOM API maintains two internal counters, one for selectNodes (page 82) and one for selectAttrNodes (page 68). This function affects only the selectAttrNodes counter.

If this function is used when selectAttrNodes is used in non iterative mode, this function still resets the internal counter, but doing so will have no effect until the next time selectAttrNodes is used in iterative mode.

The resetSelectAttrNodes function can be used in CICS and batch.

```
CALL WS-DOM-API USING
      WS-DOM-RC,        /* Halfword. Return Code */
      WS-DOM-MSG,       /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,    /* Pointer.  For internal use */
      WS-DOM-FUNCTION /* Char(25).  The name of the API */
```

Variables:

WS-DOM-RC **Output** **Mandatory**

Returns status.

WS-DOM-MSG **Output** **Mandatory**

When the return code is not zero, this field will contain messages.

WS-DOM-HANDLE **Input** **Mandatory**

For internal use. The DOM handle field uniquely identifies the XML DOM tree in which the attribute node iterator will be reset.

WS-DOM-FUNCTION **Input** **Mandatory**

Must be set to **WS-DOM-RESET-SELECT-ATTR-NODES**.

The following return codes are possible:

RC Value	Meaning
0	Successful - No messages issued.
+8	Failed. The following are the possible values for WS-DOM-MSG field: • Input error Possible input errors could be: • The function specified is not supported.
+12	SOLA DOM API abended. Abend code - XXXX (four character code returned at run time).
+16	WS-DOM-HANDLE doesn't contain a valid handle.

Example

In this example we are resetting the internal counters used by format 3 of selectAttrNodes (page 68).

```
SET WS-DOM-RESET-SELECT-ATTR-NODES TO TRUE
CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION
```

resetSelectNodes

CICS BATCH

This resetSelectNodes function is used to reset the internal counters used by the selectNodes function. The function should only be used in conjunction with the selectNodes function when it is operating in an iterative mode. When the internal counter is reset and a parent node is specified using selectNodes in iterative mode, the first child node (belonging to the specified parent) will be selected. If a parent node is not specified, then the root node of the XML document will be selected.

The SOLA DOM API maintains two internal counters, one for selectNodes (see description of selectNodes on page 82) and one for selectAttrNodes (see description of selectAttrNodes on page 68). This function affects only the selectNodes counter.

If this function is used when selectNodes is used in non iterative mode, this function still resets the internal counter, but doing so will have no effect until the next time selectNodes is used in iterative mode.

The resetSelectNodes function can be used in CICS and batch.

```
CALL WS-DOM-API USING
      WS-DOM-RC,        /* Halfword. Return Code */
      WS-DOM-MSG,       /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,    /* Pointer.  For internal use */
      WS-DOM-FUNCTION /* Char(25).  The name of the API */
```

Variables:

WS-DOM-RC **Output** **Mandatory**

Returns status.

WS-DOM-MSG **Output** **Mandatory**

When the return code is not zero, this field will contain messages.

WS-DOM-HANDLE **Input** **Mandatory**

For internal use. The DOM handle field uniquely identifies the XML DOM tree in which the node iterator will be reset.

WS-DOM-FUNCTION **Input** **Mandatory**

Must be set to **WS-DOM-RESET-SELECT-NODES**.

The following return codes are possible:

RC Value	Meaning
0	Successful - No messages issued.
+8	Failed. The following are the possible values for WS-DOM-MSG field: • Input error Possible input errors could be: • The function specified is not supported.
+12	SOLA DOM API abended. Abend code - XXXX (four character code returned at run time).
+16	WS-DOM-HANDLE doesn't contain a valid handle.

Example

In this example we are resetting the internal counters used by format 3 of selectNodes (page 82).

```
SET WS-DOM-RESET-SELECT-NODES TO TRUE
CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION
```

selectAttrNodes

CICS BATCH

The SOLA DOM API distinguishes between attribute node identifier and element node identifier.

The selectAttrNodes function is used to inquire on either the total number of attribute nodes in an XML document, to get a particular attribute in the sequence or retrieve attributes iteratively.

The selectAttrNodes function can be used in CICS and batch.

The selectAttrNodes function has three different modes of operation, each invoked using a different format of the selectAttrNodes function. In the first, it can be used to return a count of the total number of attribute nodes in the XML document. In the second, it can be used to return a particular attribute in the DOM tree by specifying the relative position of the node in WS-DOM-SEQ. In the third, it can be used to sequentially traverse all of the attribute nodes in the XML document in a top to bottom, left to right sequence.

Note that the sequence number of an attribute differs from the unique node identifier of an attribute. The sequence number is the relative position of a node in the tree and can change when nodes are inserted or deleted before it. The unique node identifier of a node (specified with WS-DOM-NODE ID) is a non-changing identifier that can be used to uniquely identify a node.

Format 1: This format is used to find how many attributes are in an XML document. Setting WS-DOM-TAG-NAME to a single asterisk instructs the SOLA DOM API to return a count of the number of attribute nodes in the XML document in the field WS-DOM-SEQ.

```
CALL WS-DOM-API USING
      WS-DOM-RC,        /* Halfword. Return Code */
      WS-DOM-MSG,       /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,    /* Pointer.  For internal use */
      WS-DOM-FUNCTION,  /* Char(25).  The name of the API */
      WS-DOM-SEQ        /* Number of attributes returned here*/
      WS-DOM-TAG-NAME,  /* Asterisk character(*) goes here */
```

The total number of attributes in the input XML document will be returned in the WS-DOM-SEQ field.

Format 2: Alternatively, you can specify an attribute's sequence in the WS-DOM-SEQ field and move spaces to WS-DOM-TAG-NAME. This will return the attribute's tag name in the WS-DOM-TAG-NAME field. The field WS-DOM-VALUE will contain the attribute's value and WS-DOM-VALUE-LENGTH will contain the length of the attribute's value

```
CALL WS-DOM-API USING
      WS-DOM-RC,        /* Halfword. Return Code */
      WS-DOM-MSG,       /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,    /* Pointer.  For internal use */
      WS-DOM-FUNCTION,  /* Char(25).  The name of the API */
      WS-DOM-SEQ,       /* Specify relative sequence # here */
      WS-DOM-TAG-NAME,    /* The tag name is returned here */
      WS-DOM-VALUE,       /* The text node is returned here */
      WS-DOM-VALUE-LENGTH, /* Length of the returned text node */
      WS-DOM-PARENT-NODE-ID, /* The ID of the parent of the node */
      WS-DOM-ATTRIBUTE-ID,/* The node id will be returned here */
      WS-DOM-PLACE-HOLDER,    /* Placeholder. */
      WS-DOM-PLACE-HOLDER,    /* Placeholder. */
      WS-DOM-PLACE-HOLDER,    /* Placeholder. */
      WS-DOM-MAX-LENGTH /* Can specify max length of data here */
```

Format 3: Another alternative for selectAttrNodes is to set WS-DOM-SEQ to zero and keep calling the SOLA DOM API iteratively. In this circumstance, the SOLA DOM API will internally increment a counter and will return the next attribute value in sequence for each call. The internal counter can be reset using the resetSelectAttrNodes function (page 64). We recommend using the resetSelectAttrNodes function before the first format 3 operation.

The default selection scope of the selectAttrNodes function is the entire XML document. If a parent node is specified with a sequence number of zero, the scope of the selection will be limited to only child attribute nodes of the specified parent node. In this case, the internal counter determines which child node is selected. If the specified parent node has no child attributes or no more child attributes remain to be selected, a return code of +2 will be returned.

```
CALL WS-DOM-API USING
      WS-DOM-RC,        /* Halfword. Return Code */
      WS-DOM-MSG,       /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,    /* Pointer.  For internal use */
      WS-DOM-FUNCTION,  /* Char(25).  The name of the API */
      WS-DOM-SEQ,       /* Set to zero */
      WS-DOM-TAG-NAME,    /* The tag name is returned here */
      WS-DOM-VALUE,       /* The text node is returned here */
      WS-DOM-VALUE-LENGTH, /* Length of the returned text node */
      WS-DOM-PARENT-NODE-ID, /* The ID of the parent of the node */
      WS-DOM-ATTRIBUE-ID,/* The node ID will be returned here */
      WS-DOM-PLACE-HOLDER,    /* Placeholder */
      WS-DOM-PLACE-HOLDER, /* Placeholder */
      WS-DOM-PLACE-HOLDER, /* Placeholder */
      WS-DOM-MAX-LENGTH /* Can specify max length of data here */
```

Variables:

WS-DOM-RC **Output** **Mandatory**

Returns status.

WS-DOM-MSG **Output** **Mandatory**
When the return code is not zero, this field will contain messages.

WS-DOM-HANDLE **Input** **Mandatory**
For internal use. The DOM handle field uniquely identifies the XML DOM tree from which the nodes will be retrieved.

WS-DOM-FUNCTION **Input** **Mandatory**
Must be set to **WS-DOM-SEL-ATTR-NODES**.

WS-DOM-SEQ **Input/Output** **Mandatory**
In format 1 requests, the total number of attributes in the DOM tree is returned in this variable. In format 2 requests, this variable is used as input only. The user specifies the attribute sequence number that will be used to select an attribute within the scope. In format 3 requests this variable is input only and must be set to zero by the user; the SOLA DOM API will maintain an internal counter.

WS-DOM-TAG-NAME **Input/Output** **Mandatory**
For format 1 requests, this field must be populated on input with a single asterisk character. For format 2 requests, this field must be populated with spaces on input and will return the attribute's tag name. For format 3 requests, when a +1 return code is issued, this field contains the name of the parent element that has ended.

WS-DOM-VALUE **Output** **Mandatory**
This field will contain the attribute's normalized text value.

WS-DOM-VALUE-LENGTH **Output** **Mandatory**
This field will contain the length of the attribute's normalized text value.

WS-DOM-PARENT-NODE-ID **Input/Output** **Optional**
This field is optional. It can be used to limit the scope of a DOM tree traversal (format 3) to all children of a specified parent. For format 2 and 3, when specified in the CALL, the SOLA DOM API will return the unique node identifier of the parent node.

WS-DOM-ATTRIBUTE-ID **Output** **Optional**
This field is optional. When specified in the CALL, the SOLA DOM API populates this variable with the unique node identifier of the attribute found. This variable is a redefinition of the WS-DOM-NODE-ID.

WS-DOM-PLACE-HOLDER **Place Holder** **Optional**

WS-DOM-PLACE-HOLDER **Place Holder** **Optional**

WS-DOM-PLACE-HOLDER **Place Holder** **Optional**

WS-DOM-MAX-LENGTH **Input** **Optional**
This optional field, when specified, tells SOLA DOM API to truncate data in WS-DOM-VALUE field if WS-DOM-VALUE-LENGTH exceeds WS-DOM-

MAX-LENGTH. A return code of +5 is returned if the returned data is truncated. In such case, WS-DOM-VALUE-LENGTH still contains the full length but data in WS-DOM-VALUE is truncated to WS-DOM-MAX-LENGTH.

The following return codes are possible:

RC Value	Meaning
0	Successful - No messages issued.
+1	A ComplexType element has ended or a Message Set has ended. The name of the ComplexType element that has ended can be found in the WS-DOM-TAG-NAME parameter.
+2	Entire DOM Tree is traversed. SOLA DOM API will reset the iterator.
+5	Returned data is bigger than allowed. Data truncated to the specified Length.
+8	Failed. The following are the possible values for WS-DOM-MSG field: • Seq too high (the sequence value is higher than the total number of attribute nodes in the document) • Input error Possible input errors could be: • The function specified is not supported.
+12	SOLA DOM API abended. Abend code - XXXX (four character code returned at run time).
+16	WS-DOM-HANDLE doesn't contain a valid handle.

Example

Format 1.

In this example, we are retrieving a count of the attributes in the DOM tree. The number of attribute nodes in the tree is returned in WS-DOM-SEQ.

```
SET  WS-DOM-SEL-ATTR-NODES TO TRUE
MOVE '*'                   TO WS-DOM-TAG-NAME

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-SEQ
                      WS-DOM-TAG-NAME
```

Format 2.
In this example, we are retrieving a specific attribute (attribute sequence number 3) from the DOM tree. In this format, we specify the relative sequence number of the attribute in the tree that we want to retrieve. The relative sequence number denotes the position of the attribute in the tree (relative to the beginning) if the tree were to be expressed as an XML stream. The first attribute in the XML stream has a relative sequence number of 1, the second attribute has a relative sequence number of 2, etc.

```
SET  WS-DOM-SEL-ATTR-NODES TO TRUE
MOVE SPACES                TO WS-DOM-TAG-NAME
MOVE +3                    TO WS-DOM-SEQ

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-SEQ,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE,
                      WS-DOM-VALUE-LENGTH
```

In this second format 2 example, we are again retrieving a specific attribute (attribute number 3) from the DOM tree. This time we are requesting that the unique node identifier of the attribute's parent be returned, along with the unique node identifier of the attribute. We are also telling the SOLA DOM API to truncate the attribute's text node value should the length of the value exceed 255 bytes.

```
SET  WS-DOM-SEL-ATTR-NODES TO TRUE
MOVE SPACES                TO WS-DOM-TAG-NAME
MOVE +3                    TO WS-DOM-SEQ
MOVE +255                  TO WS-DOM-MAX-LENGTH

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-SEQ,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE,
                      WS-DOM-VALUE-LENGTH,
                      WS-DOM-PARENT-NODE-ID,
                      WS-DOM-ATTRIBUTE-ID,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-MAX-LENGTH
```

Format 3.

In this example, we are traversing all of the attributes in the DOM tree. In this format, we set WS-DOM-SEQ to zero and then we iteratively call the SOLA DOM API until we receive a return code of +1 or +2. In this example, a +2 will be returned when the end of the document has been reached and a +1 will be returned when a complexType has ended. The name of the complexType element that ended will be returned in WS-DOM-TAG-NAME.

```
SET  WS-DOM-SEL-ATTR-NODES TO TRUE
MOVE ZERO                  TO WS-DOM-SEQ

PERFORM UNTIL WS-DOM-RC > +1
   CALL WS-DOM-API USING WS-DOM-RC,
                         WS-DOM-MSG,
                         WS-DOM-HANDLE,
                         WS-DOM-FUNCTION,
                         WS-DOM-SEQ,
                         WS-DOM-TAG-NAME,
                         WS-DOM-VALUE,
                         WS-DOM-VALUE-LENGTH
END-PERFORM
```

In this second format 3 example, we are again traversing all of the attributes in the DOM tree. This time we are requesting that the unique node identifier of the attributes's parent be returned, along with the unique node identifier of the attribute. We are also telling the SOLA DOM API to truncate the attribute's text node value should the length of the value exceed 255 bytes.

We set WS-DOM-SEQ to zero and then we iteratively call the SOLA DOM API until we receive a return code of +1 or +2. In this example, a +2 will be returned when the end of the document (the last attribute in the scope) has been reached and a +1 will be returned when a complexType has ended. The name of the complexType element that ended will be returned in WS-DOM-TAG-NAME.

```
SET  WS-DOM-SEL-ATTR-NODES TO TRUE
MOVE +255                  TO WS-DOM-MAX-LENGTH
MOVE ZERO                  TO WS-DOM-SEQ

PERFORM UNTIL WS-DOM-RC > +1
   CALL WS-DOM-API USING WS-DOM-RC,
                         WS-DOM-MSG,
                         WS-DOM-HANDLE,
                         WS-DOM-FUNCTION,
                         WS-DOM-SEQ,
                         WS-DOM-TAG-NAME,
                         WS-DOM-VALUE,
                         WS-DOM-VALUE-LENGTH,
                         WS-DOM-PARENT-NODE-ID,
                         WS-DOM-ATTRIBUTE-ID,
```

```
                              WS-DOM-PLACE-HOLDER,
                              WS-DOM-PLACE-HOLDER,
                              WS-DOM-PLACE-HOLDER,
                              WS-DOM-MAX-LENGTH
END-PERFORM
```

In this third format 3 example, we are limiting the scope of the traversal of the DOM tree to all of the child attributes of a specified parent. We limit the scope by specifying the parent's unique node identifier and setting WS-DOM-SEQ to zero. We are also are requesting the unique node identifier of the attribute and telling the SOLA DOM API to truncate the attribute's text node value should the length of the value exceed 255 bytes.

We iteratively call the SOLA DOM API until we receive a return code of +1 or +2. In this example, a +2 will be returned when the end of the document has been reached (the last attribute in the scope) and a +1 will be returned when a complexType has ended. The name of the complexType element that ended will be returned in WS-DOM-TAG-NAME.

```
SET  WS-DOM-SEL-ATTR-NODES TO TRUE
MOVE +255                  TO WS-DOM-MAX-LENGTH
MOVE ZERO                  TO WS-DOM-SEQ
MOVE SAVED-PARENT-ID       TO WS-DOM-PARENT-NODE-ID

PERFORM UNTIL WS-DOM-RC > +1
   CALL WS-DOM-API USING WS-DOM-RC,
                         WS-DOM-MSG,
                         WS-DOM-HANDLE,
                         WS-DOM-FUNCTION,
                         WS-DOM-SEQ,
                         WS-DOM-TAG-NAME,
                         WS-DOM-VALUE,
                         WS-DOM-VALUE-LENGTH,
                         WS-DOM-PARENT-NODE-ID,
                         WS-DOM-ATTRIBUTE-ID,
                         WS-DOM-PLACE-HOLDER,
                         WS-DOM-PLACE-HOLDER,
                         WS-DOM-PLACE-HOLDER,
                         WS-DOM-MAX-LENGTH
END-PERFORM
```

selectAttrNodesAddress

CICS BATCH

The selectAttrNodesAddress function is identical to selectAttrNodes, except that the address of the attribute is returned instead of the value of the attribute. This can be useful if you don't want to provide storage for the attribute value, or where you want to use the XMLDOMWS copybook variables and the attribute value is greater than 256 bytes.

The SOLA DOM API distinguishes between attribute node identifier and element node identifier.

The selectAttrNodesAddress function is used to inquire on either the total number of attribute nodes in an XML document, to get the address of a particular attribute in the sequence or retrieve attribute addresses iteratively.

The selectAttrNodesAddress function can be used in CICS and batch.

The selectAttrNodesAddress function has three different modes of operation, each of which is invoked using different formats of the selectAttrNodesAddress function. In the first, it can be used to return a count of the total number of attribute nodes in the XML document. In the second, it can be used to return the address of a particular attribute in the DOM tree by specifying the relative position of the node in WS-DOM-SEQ. In the third, it can be used to sequentially traverse all of the attribute nodes in the XML document in a top to bottom, left to right sequence.

Note that the sequence number of an attribute differs from the unique node identifier of an attribute. The sequence number is the relative position of a node in the tree and can change when nodes are inserted or deleted before it. The unique node identifier (specified with WS-DOM-NODE ID) is a non-changing identifier that can be used to uniquely identify a node.

Format 1: This format is used to find how many attributes are in an XML document. Setting WS-DOM-TAG-NAME to a single asterisk instructs the SOLA DOM API to return a count of the number of attribute nodes in the XML document in the field WS-DOM-SEQ.

```
CALL WS-DOM-API USING
      WS-DOM-RC,         /* Halfword. Return Code */
      WS-DOM-MSG,        /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,     /* Pointer.  For internal use */
      WS-DOM-FUNCTION,   /* Char(25).  The name of the API */
      WS-DOM-SEQ,         /* Number of attributes returned here*/
      WS-DOM-TAG-NAME  /* Asterisk character(*) goes here */
```

The total number of attributes in the input XML document will be returned in the WS-DOM-SEQ field.

Format 2: Alternatively, you can specify an attribute's sequence in the WS-DOM-SEQ field and move spaces to WS-DOM-TAG-NAME. This will return the attribute's tag name in the WS-DOM-TAG-NAME field. The field WS-DOM-VALUE-PTR will contain the address of the attribute's value and WS-DOM-VALUE-LENGTH will contain the length of the attribute's value

```
CALL WS-DOM-API USING
      WS-DOM-RC,         /* Halfword. Return Code */
      WS-DOM-MSG,        /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,     /* Pointer.  For internal use */
      WS-DOM-FUNCTION,   /* Char(25).  The name of the API */
      WS-DOM-SEQ,        /* Specify relative sequence # here */
      WS-DOM-TAG-NAME,    /* The tag name is returned here */
      WS-DOM-VALUE-PTR,    /* The address is returned here */
      WS-DOM-VALUE-LENGTH, /* Length of the returned text node */
      WS-DOM-PARENT-NODE-ID, /* The ID of the parent of the node */
      WS-DOM-ATTRIBUTE-ID /* The node id will be returned here */
```

Format 3: Another alternative for selectAttrNodesAddress is to set WS-DOM-SEQ to zero and keep calling the SOLA DOM API iteratively. In this circumstance, the SOLA DOM API will internally increment a counter and will return the address of the next attribute value in sequence for each call. The internal counter can be reset using the resetSelectAttrNodes function (page 75). We recommend using the resetSelectAttrNodes function before the first format 3 operation.

The default selection scope of the selectAttrNodesAddress function is the entire XML document. If a parent node is specified with a sequence number of zero on the selectAttrNodesAddress request, the scope of the selection will be limited to only child attribute nodes of the specified parent node. In this case, the internal counter determines which child node is selected. If the specified parent node has no child attributes or no more child attributes remain to be selected, a return code of +2 will be returned.

```
CALL WS-DOM-API USING
      WS-DOM-RC,         /* Halfword. Return Code */
      WS-DOM-MSG,        /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,     /* Pointer.  For internal use */
      WS-DOM-FUNCTION,   /* Char(25).  The name of the API */
      WS-DOM-SEQ,        /* Set to zero */
      WS-DOM-TAG-NAME,    /* The tag name is returned here */
      WS-DOM-VALUE-PTR,   /* The address is returned here */
      WS-DOM-VALUE-LENGTH, /* Length of the returned text node */
      WS-DOM-PARENT-NODE-ID, /* The ID of the parent of the node */
      WS-DOM-ATTRIBUTE-ID /* The node ID will be returned here */
```

Variables:

WS-DOM-RC **Output** **Mandatory**

Returns status.

WS-DOM-MSG **Output** **Mandatory**
When the return code is not zero, this field will contain messages.

WS-DOM-HANDLE **Input** **Mandatory**
For internal use. The DOM handle field uniquely identifies the XML DOM tree from which the nodes will be retrieved.

WS-DOM-FUNCTION **Input** **Mandatory**
Must be set to **WS-DOM-SEL-ATTR-NODES-ADR**.

WS-DOM-SEQ **Input/Output** **Mandatory**
In format 1 requests, the total number of attributes in the DOM tree is returned in this variable. In format 2 requests, this variable is used as input only. The user specifies the attribute sequence number that will be used to select an attribute within the scope. In format 3 requests this variable is input only and must be set to 0 by the user; the SOLA DOM API will maintain an internal counter.

WS-DOM-TAG-NAME **Output** **Mandatory**
For format 1 requests, this field must be populated on input with a single asterisk character. For format 2 requests, this field must be populated with all spaces on input and will return the attribute's tag name. For format 3 request, when a +1 return code is issued, this field contains the name of the parent element that has ended.

WS-DOM-VALUE-PTR **Output** **Mandatory**
The attribute's address will be returned here.

WS-DOM-VALUE-LENGTH **Output** **Mandatory**
This field will contain the length of the attribute's normalized text value.

WS-DOM-PARENT-NODE-ID **Input/Output** **Optional**
This field is optional. It can be used to limit the scope of a DOM tree traversal (format 3) to all children of a specified parent. For format 2 and 3, when specified in the CALL, the SOLA DOM API will return the unique node identifier of the parent node.

WS-DOM-NODE-ID **Output** **Optional**
This field is optional. When specified in the CALL, the SOLA DOM API populates this variable with the unique node identifier of the attribute found.

The following return codes are possible:

RC Value	Meaning
0	Successful - No messages issued.
+1	A ComplexType element has ended or a Message Set has ended. The name of the ComplexType element that has ended can be found in the WS-DOM-TAG-NAME parameter.
+2	Entire DOM Tree is traversed. SOLA DOM API will reset the iterator.

RC Value	Meaning
+8	Failed. The following are the possible values for WS-DOM-MSG field • Seq too high (the sequence value is higher than the total number of attribute nodes in the document) • Input error Possible input errors could be: • The function specified is not supported.
+12	SOLA DOM API abended. Abend code - XXXX (four character code returned at run time).
+16	WS-DOM-HANDLE doesn't contain a valid handle.

Example

Format 1.
In this example, we are retrieving a count of the attributes in the DOM tree. The number of attribute nodes in the tree is returned in WS-DOM-SEQ.

```
SET  WS-DOM-SEL-ATTR-NODES-ADR TO TRUE
MOVE '*'                    TO WS-DOM-TAG-NAME

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-SEQ,
                      WS-DOM-TAG-NAME
```

Format 2.
In this example, we are retrieving the address of a specific attribute (attribute sequence number 3) from the DOM tree. In this format, we specify the relative sequence number of the attribute in the tree that we want to retrieve. The relative sequence number denotes the position of the attribute in the tree (relative to the beginning) if the tree were to be expressed as an XML stream. The first attribute in the XML stream has a relative sequence number of 1, the second attribute has a relative sequence number of 2, etc.

```
SET  WS-DOM-SEL-ATTR-NODES-ADR TO TRUE
MOVE SPACES                 TO WS-DOM-TAG-NAME
MOVE +3                     TO WS-DOM-SEQ

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
```

```
                          WS-DOM-SEQ,
                          WS-DOM-TAG-NAME,
                          WS-DOM-VALUE-PTR,
                          WS-DOM-VALUE-LENGTH
```

In this second format 2 example, we are again retrieving the address of a specific attribute (attribute number 3) from the DOM tree. This time we are requesting that the unique node identifier of the attribute's parent be returned, along with the unique node identifier of the attribute.

```
SET  WS-DOM-SEL-ATTR-NODES-ADR TO TRUE
MOVE SPACES                 TO WS-DOM-TAG-NAME
MOVE +3                     TO WS-DOM-SEQ

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-SEQ,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE-PTR,
                      WS-DOM-VALUE-LENGTH,
                      WS-DOM-PARENT-NODE-ID,
                      WS-DOM-ATTRIBUTE-ID
```

Format 3.
In this example, we are traversing all of the attributes in the DOM tree. In this format, we set WS-DOM-SEQ to zero and initialize the internal counter using the resetSelectAttrNodes function (see page 64). We then iteratively call the SOLA DOM API until we receive a return code of +1 or +2. In this example, a +2 will be returned when the end of the document has been reached and a +1 will be returned when a complexType has ended. The name of the complexType element that ended will be returned in WS-DOM-TAG-NAME..

```
SET  WS-DOM-SEL-ATTR-NODES-ADR TO TRUE
MOVE ZERO                   TO WS-DOM-SEQ

PERFORM UNTIL WS-DOM-RC > +1
   CALL WS-DOM-API USING WS-DOM-RC,
                         WS-DOM-MSG,
                         WS-DOM-HANDLE,
                         WS-DOM-FUNCTION,
                         WS-DOM-SEQ,
                         WS-DOM-TAG-NAME,
                         WS-DOM-VALUE-PTR,
                         WS-DOM-VALUE-LENGTH
END-PERFORM
```

In this second format 3 example, we are again traversing all of the attributes in the DOM tree. This time we are requesting that the unique node identifier of the attributes's parent be returned, along with the unique node identifier of the attribute.

We set WS-DOM-SEQ to zero and then we iteratively call the SOLA DOM API until we receive a return code of +1 or +2. In this example, a +2 will be returned when the end of the document has been reached (the last attribute in the scope) and a +1 will be returned when a complexType has ended. The name of the complexType element that ended will be returned in WS-DOM-TAG-NAME.

```
SET  WS-DOM-SEL-ATTR-NODES-ADR TO TRUE
MOVE ZERO                      TO WS-DOM-SEQ

PERFORM UNTIL WS-DOM-RC > +1
   CALL WS-DOM-API USING WS-DOM-RC,
                         WS-DOM-MSG,
                         WS-DOM-HANDLE,
                         WS-DOM-FUNCTION,
                         WS-DOM-SEQ,
                         WS-DOM-TAG-NAME,
                         WS-DOM-VALUE-PTR,
                         WS-DOM-VALUE-LENGTH,
                         WS-DOM-PARENT-NODE-ID,
                         WS-DOM-ATTRIBUTE-ID
END-PERFORM
```

In this third format 3 example, we are limiting the scope of the traversal of the DOM tree to all of the child attributes of a specified parent. We limit the scope by specifying the parent's unique node identifier and setting the WS-DOM-SEQ to zero. We are also are requesting the unique node identifier of the attribute.

Having set WS-DOM-SEQ to zero, we iteratively call the SOLA DOM API until we receive a return code of +1 or +2. In this example, a +2 will be returned when the end of the document has been reached (the last attribute in the scope) and a +1 will be returned when a complexType has ended. The name of the complexType element that ended will be returned in WS-DOM-TAG-NAME.

```
SET  WS-DOM-SEL-ATTR-NODES-ADR TO TRUE
MOVE ZERO                      TO WS-DOM-SEQ
MOVE SAVED-PARENT-ID           TO WS-DOM-PARENT-NODE-ID

PERFORM UNTIL WS-DOM-RC > +1
   CALL WS-DOM-API USING WS-DOM-RC,
                         WS-DOM-MSG,
                         WS-DOM-HANDLE,
                         WS-DOM-FUNCTION,
                         WS-DOM-SEQ,
                         WS-DOM-TAG-NAME,
                         WS-DOM-VALUE-PTR,
                         WS-DOM-VALUE-LENGTH,
                         WS-DOM-PARENT-NODE-ID,
                         WS-DOM-NODE-ID
```

```
END-PERFORM
```

selectNodes

CICS BATCH

The selectNodes function is used to inquire on either the total number of element nodes in an XML document, to get a particular element in the sequence or retrieve elements iteratively. Attribute nodes and namespace nodes are not selected using this function.

The selectNodes function can be used in CICS and batch.

The selectNodes function has three different modes of operation, each of which is invoked using different formats of the selectNodes function. In the first it can be used to return a count of the total number of element nodes in the XML document. In the second it can be used to return a particular element in the DOM tree by specifying the relative position of the element node in WS-DOM-SEQ. In the third it can be used to sequentially traverse all of the element nodes in the XML document in a top to bottom, left to right sequence.

Note that the sequence number of an element differs from the unique node identifier of an element. The sequence number is the relative position of an element node in the tree and can change when element nodes are inserted or deleted before it. The unique node identifier of an element (specified with WS-DOM-NODE ID) is a non-changing identifier that can be used to uniquely identify an element node.

Format 1: This format is used to find how many elements are in an XML document. Setting WS-DOM-TAG-NAME to a single asterisk instructs the SOLA DOM API to return a count of the number of element nodes in the XML document in the field WS-DOM-SEQ.

```
CALL WS-DOM-API USING
      WS-DOM-RC,        /* Halfword. Return Code */
      WS-DOM-MSG,       /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,    /* Pointer.  For internal use */
      WS-DOM-FUNCTION,  /* Char(25).  The name of the API */
      WS-DOM-SEQ,        /* Number of elements returned here*/
      WS-DOM-TAG-NAME  /* Asterisk character (*) goes here. */
```

The total number of elements in the input XML document will be returned in the WS-DOM-SEQ field.

Format 2: Alternatively, you can specify an element sequence in the WS-DOM-SEQ field and move spaces to WS-DOM-TAG-NAME. This will return the element tag name in the WS-DOM-TAG-NAME field. The field WS-DOM-VALUE will contain the element value and WS-DOM-VALUE-LENGTH will contain the length of the element value.

```
CALL WS-DOM-API USING
```

```
      WS-DOM-RC,         /* Halfword. Return Code */
      WS-DOM-MSG,        /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,     /* Pointer.  For internal use */
      WS-DOM-FUNCTION,   /* Char(25).  The name of the API */
      WS-DOM-SEQ,        /* Specify relative sequence # here */
      WS-DOM-TAG-NAME,     /* The tag name is returned here */
      WS-DOM-VALUE,        /* The text node is returned here */
      WS-DOM-VALUE-LENGTH, /* Length of the returned text node */
      WS-DOM-PARENT-NODE-ID, /* The ID of the parent of the node */
      WS-DOM-ELEMENT-ID, /* The node id will be returned here */
      WS-DOM-NMS-NODE-ID,    /* Unique id of the namespace node */
      WS-DOM-CHILD-ELEM-COUNT, /* Num of children returned here */
      WS-DOM-ATTR-NODE-COUNT, /* Num of attributes returned here */
      WS-DOM-MAX-LENGTH /* Can specify max length of data here */
```

Format 3: Another alternative for selectNodes is to set WS-DOM-SEQ to zero and keep calling the SOLA DOM API iteratively. In this circumstance, the SOLA DOM API will internally increment a counter and will return the next element node in sequence for each call. The internal counter can be reset using the resetSelectNodes function (page 66). We recommend using the resetSelectNodes function before the first format 3 operation.

The default selection scope of the selectNodes function is the entire XML document. If a parent node is specified with a sequence number of zero on the selectNodes request, the scope of the selection will be limited to only child nodes of the specified parent node. In this case the first child of the specified parent node will be the first node selected. In this case, the internal counter determines which child node is selected. If the specified parent node has no children or no more children to be selected, a return code of +2 will be returned.

```
CALL WS-DOM-API USING
      WS-DOM-RC,         /* Halfword. Return Code */
      WS-DOM-MSG,        /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,     /* Pointer.  For internal use */
      WS-DOM-FUNCTION,   /* Char(25).  The name of the API */
      WS-DOM-SEQ,        /* Set to zero */
      WS-DOM-TAG-NAME,     /* The tag name is returned here */
      WS-DOM-VALUE,        /* The text node is returned here */
      WS-DOM-VALUE-LENGTH, /* Length of the returned text node */
      WS-DOM-PARENT-NODE-ID, /* The ID of the parent of the node */
      WS-DOM-ELEMENT-ID, /* The node id will be returned here */
      WS-DOM-NMS-NODE-ID,    /* Unique id of the namespace node */
      WS-DOM-CHILD-ELEM-COUNT, /* Num of children returned here */
      WS-DOM-ATTR-NODE-COUNT, /* Num of attributes returned here */
      WS-DOM-MAX-LENGTH /* Can specify max length of data here */
```

Variables:

WS-DOM-RC **Output** **Mandatory**

Returns status.

WS-DOM-MSG **Output** **Mandatory**
When the return code is not zero, this field will contain messages.

WS-DOM-HANDLE **Input** **Mandatory**
For internal use. The DOM handle field uniquely identifies the XML DOM tree from which the nodes will be retrieved.

WS-DOM-FUNCTION **Input** **Mandatory**
Must be set to **WS-DOM-SELECT-NODES**.

WS-DOM-SEQ **Input/Output** **Mandatory**
In format 1 requests, the total number of elements in the DOM tree is returned in this variable. In format 2 requests, this variable is used as input only. The user specifies the element sequence number that will be used to select an element within the scope. In format 3 requests this variable is input only and must be set to 0 by the user; the SOLA DOM API will maintain an internal counter.

WS-DOM-TAG-NAME **Input/Output** **Mandatory**
For format 1 requests, this field must be populated on input with a single asterisk character. For format 2 requests, this field must be populated on input with all spaces and returns the element tag name. If using format 3 and in a case where a return code of +1 is returned, the parent element name of the complex type that just ended will be returned here.

WS-DOM-VALUE **Output** **Mandatory**
When an element has a text node attached, this field will contain the normalized text value.

WS-DOM-VALUE-LENGTH **Output** **Mandatory**
When an element has a text node attached, this field will contain the length of the text value of the attribute.

WS-DOM-PARENT-NODE-ID **Input/Output** **Optional**
This field is optional. It can be used to limit the scope of a DOM tree traversal (format 3) to all children of a specified parent. For formats 2 and 3, when specified in the CALL, the SOLA DOM API will return the unique node identifier of the parent node.

WS-DOM-ELEMENT-ID **Output** **Optional**
This field is optional. When specified in the CALL, the SOLA DOM API populates this variable with the unique node identifier of the element found. This variable is a redefinition of WS-DOM-NODE-ID.

WS-DOM-NMS-NODE-ID **Output** **Optional**
This is the unique node identifier for a namespace. If there is a namespace for the node, the unique node identifier of the namespace will be returned in this variable, else null will be returned.

WS-DOM-CHILD-ELEM-COUNT **Output** **Optional**

If the element node has child elements (a Complex Type), the SOLA DOM API populates this field with the child element node count.

WS-DOM-ATTR-NODE-COUNT **Output** **Optional**

If the element node has attribute nodes, the SOLA DOM API populates this field with the attribute node count.

WS-DOM-MAX-LENGTH **Input** **Optional**

This optional field, when specified, tells SOLA DOM API to truncate data in WS-DOM-VALUE field if WS-DOM-VALUE-LENGTH exceeds WS-DOM-MAX-LENGTH. A return code of +5 is returned if the returned data is truncated. In such case, WS-DOM-VALUE-LENGTH still contains the full length but data in WS-DOM-VALUE is truncated to WS-DOM-MAX-LENGTH.

The following return codes are possible:

RC Value	Meaning
0	Successful - No messages issued.
+1	A ComplexType element has ended or a Message Set has ended. The name of the ComplexType element that has ended can be found in the WS-DOM-TAG-NAME parameter.
+2	Entire DOM Tree is traversed. SOLA DOM API will reset the iterator.
+5	Returned data is bigger than allowed. Data truncated to the specified Length.
+8	Failed. The following are the possible values for WS-DOM-MSG field: • Seq too high (the sequence value is higher than the total number of element nodes in the document) • Input error Possible input errors could be: • The function specified is not supported.
+12	SOLA DOM API abended. Abend code - XXXX (four character code returned at run time).
+16	WS-DOM-HANDLE doesn't contain a valid handle.

Example

Format 1.

In this example we are retrieving a count of the elements in the DOM tree. The number of element nodes in the tree is returned in WS-DOM-SEQ.

```
SET  WS-DOM-SELECT-NODES TO TRUE
MOVE '*'                 TO WS-DOM-TAG-NAME

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-SEQ,
                      WS-DOM-TAG-NAME
```

Format 2.
In this example we are retrieving a specific element (element sequence number 3) from the DOM tree. In this format, we specify the relative sequence number of the element in the tree that we want to retrieve. The relative sequence number denotes the position of the element in the tree (relative to the beginning) if the tree were to be expressed as an XML stream. The first element in the XML stream has a relative sequence number of 1, the second element has a relative sequence number of 2, etc.

```
SET  WS-DOM-SELECT-NODES TO TRUE
MOVE SPACES              TO WS-DOM-TAG-NAME
MOVE +3                  TO WS-DOM-SEQ

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-SEQ,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE,
                      WS-DOM-VALUE-LENGTH
```

In this second format 2 example we are again retrieving a specific element (element sequence number 3) from the DOM tree. This time we are requesting that the unique node identifier of the element's parent be returned, along with the unique node identifier of the element, the unique node identifier of the element's namespace (if any), the number of child elements and number of attributes belonging to the element. We are also telling the SOLA DOM API to truncate the element's text node value should the length of the value exceed 255 bytes.

```
SET  WS-DOM-SELECT-NODES TO TRUE
MOVE SPACES              TO WS-DOM-TAG-NAME
MOVE +3                  TO WS-DOM-SEQ
MOVE +255                TO WS-DOM-MAX-LENGTH

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-SEQ,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE,
```

```
                              WS-DOM-VALUE-LENGTH,
                              WS-DOM-PARENT-NODE-ID,
                              WS-DOM-ELEMENT-ID,
                              WS-DOM-NMS-NODE-ID,
                              WS-DOM-CHILD-ELEM-COUNT,
                              WS-DOM-ATTR-NODE-COUNT,
                              WS-DOM-MAX-LENGTH
```

Format 3.

In this example we are traversing all of the elements in the DOM tree. In this format, we set WS-DOM-SEQ to zero and initialize the internal counter to start iterating from the beginning using the resetSelectNodes function (see page 66). We then iteratively call the SOLA DOM API until we receive a return code of +2.

```
SET  WS-DOM-SELECT-NODES TO TRUE
MOVE ZERO                TO WS-DOM-SEQ

PERFORM UNTIL WS-DOM-RC > +1
   CALL WS-DOM-API USING WS-DOM-RC,
                         WS-DOM-MSG,
                         WS-DOM-HANDLE,
                         WS-DOM-FUNCTION,
                         WS-DOM-SEQ,
                         WS-DOM-TAG-NAME,
                         WS-DOM-VALUE,
                         WS-DOM-VALUE-LENGTH
END-PERFORM
```

In this second format 3 example we are again traversing all of the elements in the DOM tree. This time we are requesting that the unique node identifier of the element's parent be returned, along with the unique node identifier of the element, the unique node identifier of the element's namespace (if any), the number of child elements and the number of attributes belonging to the element. We are also telling the SOLA DOM API to truncate the element's text node value should the length of the value exceed 255 bytes.

We set WS-DOM-SEQ to zero and then we iteratively call the SOLA DOM API until we receive a return code of +1 or +2. In this example, a +2 will be returned when the end of the document has been reached (the last element in the scope) and a +1 will be returned when a complexType has ended. The name of the complexType element that ended will be returned in WS-DOM-TAG-NAME.

```
SET  WS-DOM-SELECT-NODES TO TRUE
MOVE +255                TO WS-DOM-MAX-LENGTH
MOVE ZERO                TO WS-DOM-SEQ
```

```
PERFORM UNTIL WS-DOM-RC > +1
   CALL WS-DOM-API USING WS-DOM-RC,
                         WS-DOM-MSG,
                         WS-DOM-HANDLE,
                         WS-DOM-FUNCTION,
                         WS-DOM-SEQ,
                         WS-DOM-TAG-NAME,
                         WS-DOM-VALUE,
                         WS-DOM-VALUE-LENGTH,
                         WS-DOM-PARENT-NODE-ID,
                         WS-DOM-ELEMENT-ID,
                         WS-DOM-NMS-NODE-ID,
                         WS-DOM-CHILD-ELEM-COUNT,
                         WS-DOM-ATTR-NODE-COUNT,
                         WS-DOM-MAX-LENGTH
END-PERFORM
```

In this third format 3 example we are limiting the scope of the traversal of the DOM tree to all of the child elements of a specified parent. We limit the scope by specifying the parent's unique node identifier and setting the WS-DOM-SEQ to zero. We are also are requesting the unique node identifier of the element, the unique node identifier of the element's namespace (if any), the number of child elements and number of attributes belonging to the element. Additionally, telling the SOLA DOM API to truncate the element's text node value should the length of the value exceed 255 bytes.

Having set WS-DOM-SEQ to zero, we iteratively call the SOLA DOM API until we receive a return code of +1 or +2. In this example, a +2 will be returned when the end of the document has been reached (the last element in the scope) and a +1 will be returned when a complexType has ended. The name of the complexType element that ended will be returned in WS-DOM-TAG-NAME.

```
SET  WS-DOM-SELECT-NODES TO TRUE
MOVE +255                TO WS-DOM-MAX-LENGTH
MOVE ZERO                TO WS-DOM-SEQ
MOVE SAVED-PARENT-ID     TO WS-DOM-PARENT-NODE-ID

PERFORM UNTIL WS-DOM-RC > +1
   CALL WS-DOM-API USING WS-DOM-RC,
                         WS-DOM-MSG,
                         WS-DOM-HANDLE,
                         WS-DOM-FUNCTION,
                         WS-DOM-SEQ,
                         WS-DOM-TAG-NAME,
                         WS-DOM-VALUE,
                         WS-DOM-VALUE-LENGTH,
                         WS-DOM-PARENT-NODE-ID,
                         WS-DOM-ELEMENT-ID,
                         WS-DOM-NMS-NODE-ID,
                         WS-DOM-CHILD-ELEM-COUNT,
                         WS-DOM-ATTR-NODE-COUNT,
                         WS-DOM-MAX-LENGTH
END-PERFORM
```

selectNodesAddress

CICS BATCH

The selectNodesAddress function is identical to selectNodes, except that the address of the element is returned instead of the value of the element. This can be useful if you don't want to provide storage for the element value, or where you want to use the XMLDOMWS copybook variables and the element value is greater than 256 bytes. Attribute nodes and namespace nodes are not selected using this function.

The selectNodesAddress function is used to inquire on either the total number of element nodes in an XML document, to get the address of a particular element in the sequence or retrieve element addresses iteratively.

The selectNodesAddress function can be used in CICS and batch.

The selectNodesAddress function has three different modes of operation, each of which is invoked using a different format of the selectNodesAddress function. In the first mode it can be used to return a count of the total number of element nodes in the XML document. In the second it can be used to return the address of a particular element in the DOM tree by specifying the relative position of the element node in WS-DOM-SEQ. In the third it can be used to sequentially traverse all of the element nodes in the XML document in a top to bottom, left to right sequence.

Note that the sequence number of an element differs from the unique node identifier of an element. The sequence number is the relative position of an element node in the tree and can change when element nodes are inserted or deleted before it. The unique node identifier of an element (specified with WS-DOM-NODE ID) is a non-changing identifier that can be used to uniquely identify an element node.

Format 1: This format is used to find how many elements are in an XML document. Setting WS-DOM-TAG-NAME to a single asterisk instructs the SOLA DOM API to return a count of the number of element nodes in the XML document in the field WS-DOM-SEQ.

```
CALL WS-DOM-API USING
      WS-DOM-RC,        /* Halfword. Return Code */
      WS-DOM-MSG,       /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,    /* Pointer.  For internal use */
      WS-DOM-FUNCTION, /* Char(25).  The name of the API */
      WS-DOM-SEQ,        /* Number of elements returned here*/
      WS-DOM-TAG-NAME  /* Asterisk character (*) goes here. */
```

The total number of elements in the input XML document will be returned in the WS-DOM-SEQ field.

Format 2: Alternatively, you can specify an element sequence in the WS-DOM-SEQ field and move spaces to WS-DOM-TAG-NAME. This will return the element tag name in the WS-DOM-TAG-NAME field. The field WS-DOM-VALUE-PTR will contain the address of the element value and WS-DOM-VALUE-LENGTH will contain the length of the element value.

```
CALL WS-DOM-API USING
      WS-DOM-RC,         /* Halfword. Return Code */
      WS-DOM-MSG,        /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,     /* Pointer.  For internal use */
      WS-DOM-FUNCTION,   /* Char(25).  The name of the API */
      WS-DOM-SEQ,        /* Specify relative sequence # here */
      WS-DOM-TAG-NAME,    /* The tag name is returned here */
      WS-DOM-VALUE-PTR,       /* The address is returned here */
      WS-DOM-VALUE-LENGTH, /* Length of the returned text node */
      WS-DOM-PARENT-NODE-ID, /* The ID of the parent of the node */
      WS-DOM-ELEMENT-ID,    /* The node id will be returned here */
      WS-DOM-NMS-NODE-ID,    /* Unique id of the namespace node */
      WS-DOM-CHILD-ELEM-COUNT, /* Num of children returned here */
      WS-DOM-ATTR-NODE-COUNT /* Num of attributes returned here */
```

Format 3: Another alternative for selectNodes is to set WS-DOM-SEQ to zero and keep calling the SOLA DOM API iteratively. In this circumstance, the SOLA DOM API will internally increment a counter and will return the address of the next element node in sequence for each call. The internal counter can be reset using the resetSelectNodes function (page 66). We recommend using the resetSelectNodes function before the first format 3 operation.

The default selection scope of the selectNodes function is the entire XML document. If a parent node is specified with a sequence number of zero on the selectNodes request, the scope of the selection will be limited to only child nodes of the specified parent node. In this case the first child of the specified parent node will be the first node selected. In this case, the internal counter determines which child node is selected. If the specified parent node has no children or no more children to be selected, a return code of +2 will be returned.

```
CALL WS-DOM-API USING
      WS-DOM-RC,         /* Halfword. Return Code */
      WS-DOM-MSG,        /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,     /* Pointer.  For internal use */
      WS-DOM-FUNCTION,   /* Char(25).  The name of the API */
      WS-DOM-SEQ,        /* Set to zero */
      WS-DOM-TAG-NAME,    /* The tag name is returned here */
      WS-DOM-VALUE-PTR,       /* The address is returned here */
      WS-DOM-VALUE-LENGTH, /* Length of the returned text node */
      WS-DOM-PARENT-NODE-ID, /* The ID of the parent of the node */
      WS-DOM-ELEMENT-ID,    /* The node id will be returned here */
      WS-DOM-NMS-NODE-ID,    /* Unique id of the namespace node */
      WS-DOM-CHILD-ELEM-COUNT, /* Num of children returned here */
      WS-DOM-ATTR-NODE-COUNT /* Num of attributes returned here */
```

Variables:

WS-DOM-RC **Output** **Mandatory**
Returns status.

WS-DOM-MSG **Output** **Mandatory**
When the return code is not zero, this field will contain messages.

WS-DOM-HANDLE **Input** **Mandatory**
For internal use. The DOM handle field uniquely identifies the XML DOM tree from which the nodes will be retrieved.

WS-DOM-FUNCTION **Input** **Mandatory**
Must be set to **WS-DOM-SEL-NODE-ADR**.

WS-DOM-SEQ **Input/Output** **Mandatory**
In format 1 requests, the total number of elements in the DOM tree is returned in this variable. In format 2 requests, this variable is used as input only. The user specifies the element sequence number that will be used to select an element within the scope. In format 3 requests this variable is input only and must be set to 0 by the user; the SOLA DOM API will maintain an internal counter.

WS-DOM-TAG-NAME **Input/Output** **Mandatory**
For format 1 requests, this field must be populated on input with a single asterisk. For format 2 requests, this field must be populated on input with spaces and returns the element tag name. If using format 3 and in a case where a return code of +1 is returned, the parent element name of the complex type that just ended will be returned here.

WS-DOM-VALUE-PTR **Output** **Mandatory**
The element's address will be returned here.

WS-DOM-VALUE-LENGTH **Output** **Mandatory**
When an element has a text node attached, this field will contain the length of the text value of the attribute.

WS-DOM-PARENT-NODE-ID **Input/Output** **Optional**
This field is optional. It can be used to limit the scope of a DOM tree traversal (format 3) to all children of a specified parent. For formats 2 and 3, when specified in the CALL, the SOLA DOM API will return the unique node identifier of the parent node.

WS-DOM-ELEMENT-ID **Output** **Optional**
This field is optional. When specified in the CALL, the SOLA DOM API populates this variable with the unique node identifier of the element found. This variable is a redefinition of WS-DOM-NODE-ID.

WS-DOM-NMS-NODE-ID **Output** **Optional**

This is the unique node identifier for a namespace. If there is a namespace for the node, the unique node identifier of the namespace will be returned in this variable, else a null value will be returned.

WS-DOM-CHILD-ELEM-COUNT **Output** **Optional**

If the element node has child elements (a Complex Type), the SOLA DOM API populates this field with the child element node count.

WS-DOM-ATTR-NODE-COUNT **Output** **Optional**

If the element node has attribute nodes, the SOLA DOM API populates this field with the attribute node count.

The following return codes are possible:

RC Value	Meaning
0	Successful - No messages issued.
+1	A ComplexType element has ended or a Message Set has ended. The name of the ComplexType element that has ended can be found in the WS-DOM-TAG-NAME parameter.
+2	Entire DOM Tree is traversed. SOLA DOM API will reset the iterator.
+8	Failed. The following are the possible values for WS-DOM-MSG field: • Seq too high (the sequence value is higher than the total number of element nodes in the document) • Input error Possible input errors could be: • The function specified is not supported.
+12	SOLA DOM API abended. Abend code - XXXX (four character code returned at run time).
+16	WS-DOM-HANDLE doesn't contain a valid handle.

Example

Format 1.

In this example we are retrieving a count of the elements in the DOM tree. The number of element nodes in the tree is returned in WS-DOM-SEQ.

```
SET  WS-DOM-SEL-NODE-ADR TO TRUE
MOVE '*'                 TO WS-DOM-TAG-NAME

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
```

```
                          WS-DOM-HANDLE,
                          WS-DOM-FUNCTION,
                          WS-DOM-SEQ,
                          WS-DOM-TAG-NAME
```

Format 2.
In this example we are retrieving the address of a specific element (element sequence number 3) from the DOM tree. In this format, we specify the relative sequence number of the element in the tree. The relative sequence number denotes the position of the element in the tree (relative to the beginning) if the tree were to be expressed as an XML stream. The first element in the XML stream has a relative sequence number of 1, the second element has a relative sequence number of 2, etc.

```
SET  WS-DOM-SEL-NODE-ADR TO TRUE
MOVE SPACES              TO WS-DOM-TAG-NAME
MOVE +3                  TO WS-DOM-SEQ

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-SEQ,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE-PTR,
                      WS-DOM-VALUE-LENGTH
```

Format 3.
In this example we are traversing all of the elements in the DOM tree. In this format, we set WS-DOM-SEQ to zero and initialize the internal counter using the resetSelectNodes function (see page 66). We then iteratively call the SOLA DOM API until we receive a return code of +1 or +2. In this example, a +2 will be returned when the end of the document has been reached (the last element in the scope) and a +1 will be returned when a complexType has ended. The name of the complexType element that ended will be returned in WS-DOM-TAG-NAME.

```
SET  WS-DOM-SEL-NODE-ADR TO TRUE
MOVE ZERO                TO WS-DOM-SEQ

PERFORM UNTIL WS-DOM-RC > +1
   CALL WS-DOM-API USING WS-DOM-RC,
                         WS-DOM-MSG,
                         WS-DOM-HANDLE,
                         WS-DOM-FUNCTION,
                         WS-DOM-SEQ,
                         WS-DOM-TAG-NAME,
                         WS-DOM-VALUE-PTR,
                         WS-DOM-VALUE-LENGTH
```

```
END-PERFORM
```

In this second format 3 example we are limiting the scope of the traversal of the DOM tree to all of the child elements of a specified parent. We limit the scope by specifying the parent's unique node identifier and setting the WS-DOM-SEQ to zero. We are also requesting the unique node identifier of the element, the unique node identifier of the element's namespace (if any), the number of child elements and number of attributes belonging to the element.

Having set WS-DOM-SEQ to zero, we iteratively call the SOLA DOM API until we receive a return code of +1 or +2. In this example, a +2 will be returned when the end of the document has been reached (the last element in the scope) and a +1 will be returned when a complexType has ended. The name of the complexType element that ended will be returned in WS-DOM-TAG-NAME.

```
SET  WS-DOM-SELECT-NODES TO TRUE
MOVE ZERO                TO WS-DOM-SEQ
MOVE SAVED-PARENT-ID     TO WS-DOM-PARENT-NODE-ID

PERFORM UNTIL WS-DOM-RC > +1
   CALL WS-DOM-API USING WS-DOM-RC,
                         WS-DOM-MSG,
                         WS-DOM-HANDLE,
                         WS-DOM-FUNCTION,
                         WS-DOM-SEQ,
                         WS-DOM-TAG-NAME,
                         WS-DOM-VALUE-PTR,
                         WS-DOM-VALUE-LENGTH,
                         WS-DOM-PARENT-NODE-ID,
                         WS-DOM-ELEMENT-ID,
                         WS-DOM-NMS-NODE-ID,
                         WS-DOM-CHILD-ELEM-COUNT,
                         WS-DOM-ATTR-NODE-COUNT
END-PERFORM
```

API DESCRIPTIONS – XML CREATION AND MODIFICATION FUNCTIONS

appendChild

The appendChild function appends a new child element to a specified parent element. Optionally, a text node value can be attached to the element by specifying a value in WS-DOM-VALUE and a value length in WS-DOM-VALUE LENGTH.

The appendChild function is used when you are building a new XML document or updating an existing XML document. If you are building a new XML document, then you must have created the document using the createDocument API. If you are updating an existing XML document, then that document must already exist in the DOM tree. The variable WS-DOM-HANDLE must be a valid handle that references the internal control blocks and tree maintained by the SOLA DOM API.

The owning parent element node where the child element will be appended can be specified in one of two ways. First, the parent element node can be specified by populating the WS-DOM-PARENT field with the name of the parent element. If multiple instances of the parent name exist within the XML document, the child element will be appended to the *first* occurrence of the parent element. The selected parent's unique node identifier will be returned in the WS-DOM-PARENT-NODE-ID field. Second, the parent node identifier can be specified by populating the WS-DOM-PARENT-NODE-ID field with the unique node identifier that uniquely identifies a node within the XML DOM. Since the parent's node identifier is more specific than the parent name, the specification of a parent node identifier in the appendChild request will take precedence over a parent name. The SOLA DOM API will consider any value other than null value in the WS-DOM-PARENT-NODE-ID field to be an input that will take precedence over WS-DOM-PARENT.

After the appendChild function call is executed is executed successfully, if the parent is a complexType element, then the child will added as the last child of the element. If the element is not a complexType element, it will become a complex type (with one child).

The appendChild function can be used in CICS and batch.

```
CALL WS-DOM-API USING
      WS-DOM-RC,        /* Halfword. Return Code */
      WS-DOM-MSG,       /* Char(80). Output msg if return code <> 0 */
```

```
WS-DOM-HANDLE,    /* Pointer.  For internal use */
WS-DOM-FUNCTION, /* Char(25).  The name of the API */
WS-DOM-PARENT,   /* Name of the parent element */
WS-DOM-TAG-NAME, /* Space delimited. Names the child element */
WS-DOM-VALUE,    /* The child text value (if any) */
WS-DOM-VALUE-LENGTH, /* The child text value length */
WS-DOM-PARENT-NODE-ID, /* Unique ID of the parent node */
WS-DOM-ELEMENT-ID /* The child ID will be returned here */
```

Variables:

WS-DOM-RC **Output** **Mandatory**

Returns status.

WS-DOM-MSG **Output** **Mandatory**

When the return code is not zero, this field will contain messages.

WS-DOM-HANDLE **Input** **Mandatory**

For internal use. The DOM handle field uniquely identifies the XML DOM tree in which the child element will be appended.

WS-DOM-FUNCTION **Input** **Mandatory**

Must be set to **WS-DOM-APPEND-CHILD**.

WS-DOM-PARENT **Input** **Mandatory**

This field contains the parent element name. The value in this field is mutually exclusive with WS-DOM-PARENT-NODE-ID. When the parent name is duplicate, the SOLA DOM API picks the *first* element in the document that has the matching parent name. The selected parent's unique node identifier will be returned in the field WS-DOM-PARENT-NODE-ID after a successful call.

WS-DOM-TAG-NAME **Input** **Mandatory**

Populate this field with name of the element. The name of element must end with a space for delimiting purposes.

WS-DOM-VALUE **Input** **Mandatory**

This field can be used to set the text node while appending the element node in the same call. The length of this field is dependent on WS-DOM-VALUE-LENGTH. This variable can be replaced with any other variable of greater length.

WS-DOM-VALUE-LENGTH **Input** **Mandatory**

This field controls the length of the text node to be added to the element. When this value is set to zero, no text node is added.

WS-DOM-PARENT-NODE-ID **Input/Output** **Optional**

This is an optional field. This field contains the unique node identifier of the parent node. When specified (any value other than null) in the CALL statement, it takes precedence over WS-DOM-PARENT field. On input

(any value other than null is considered input), the SOLA DOM API uses this field to append a child element to the specified parent node. On output, this field contains the unique node-id of the parent element to which the child element is appended.

WS-DOM-ELEMENT-ID Output Optional

This is an optional field. When specified in the CALL statement the unique node identifier is returned in this variable. The unique node identifier is an immutable property that uniquely identifies the node that was just appended to the XML DOM. This node identifier can be used in other XML SOLA DOM API functions to uniquely identify the node to be operated on. This variable is a redefinition of WS-DOM-NODE-ID.

The following return codes are possible:

RC Value	Meaning
0	Successful - No messages issued.
+8	Failed. The following are the possible values for WS-DOM-MSG field: • Parent not found • Parent node identifier does not exist • Input error Possible input errors could be • The function specified is not supported • Element tag name not specified
+12	SOLA DOM API abended. Abend code - XXXX (four character code returned at run time).
+16	WS-DOM-HANDLE doesn't contain a valid handle.

Example

The following example appends the "Transaction" child element to the first occurrence of the "ResponseData" parent element in the XML document. Since the parent is specified by name the WS-DOM-PARENT-NODE-ID parameter is not used and is not specified. Additionally, the optional return value WS-DOM-NODE-ID is not used and is not specified.

```
SET  WS-DOM-APPEND-CHILD TO TRUE
MOVE 'Transaction'       TO WS-DOM-TAG-NAME
MOVE 'ResponseData'      TO WS-DOM-PARENT
MOVE SPACES              TO WS-DOM-VALUE
MOVE ZERO                TO WS-DOM-VALUE-LENGTH
```

```
CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PARENT,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE,
                      WS-DOM-VALUE-LENGTH
```

The next example is the same as the previous example in that it appends the "Transaction" child element to the first occurrence of the "ResponseData" parent element in the XML document. In this example the node identifier of the node to be appended to ("ResponseData", the parent node) has its unique node identifier saved in SAVED-PARENT-ID. This value is moved to WS-DOM-PARENT-NODE-ID, which is used to uniquely identify "ResponseData" as the parent to be appended to.

WS-DOM-PARENT is not used in this case and is replaced by WS-DOM-PLACE-HOLDER.

```
SET  WS-DOM-APPEND-CHILD TO TRUE
MOVE 'Transaction'       TO WS-DOM-TAG-NAME
MOVE SAVED-PARENT-ID     TO WS-DOM-PARENT-NODE-ID
MOVE SPACES              TO WS-DOM-VALUE
MOVE ZERO                TO WS-DOM-VALUE-LENGTH

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE,
                      WS-DOM-VALUE-LENGTH,
                      WS-DOM-PARENT-NODE-ID,
                      WS-DOM-ELEMENT-ID
```

appendChildBefore

CICS BATCH

The appendChildBefore function is used to add an element before a particular element. The element node to be added will share the same parent as the element that it is to be appended before. The appendChildBefore function requires you to specify the unique node identifier (WS-DOM-NODE-ID) before which an insertion will be made. You can obtain the unique node identifier with the getElementBy* function calls.

A text node for the child element can be specified in the function call by populating WS-DOM-VALUE with the value of the text to be added, and populating WS-DOM-VALUE-LENGTH with the length of the text. A length of zero specifies that no text is to be added.

The appendChildBefore function can be used in CICS and batch.

```
CALL WS-DOM-API USING
      WS-DOM-RC,         /* Halfword. Return Code */
      WS-DOM-MSG,        /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,     /* Pointer.  For internal use */
      WS-DOM-FUNCTION,   /* Char(25).  The name of the API */
      WS-DOM-PLACE-HOLDER, /* Placeholder */
      WS-DOM-TAG-NAME,   /* Space delimited. Names the child element */
      WS-DOM-VALUE,      /* The child text value (if any) */
      WS-DOM-VALUE-LENGTH, /* The child text value length */
      WS-DOM-PARENT-NODE-ID, /* Unique parent node id */
      WS-DOM-ELEMENT-ID  /*Input- node to add before. Output- node added */
```

Variables:

WS-DOM-RC **Output** **Mandatory**

Returns status.

WS-DOM-MSG **Output** **Mandatory**

When the return code is not zero, this field will contain messages.

WS-DOM-HANDLE **Input** **Mandatory**

For internal use. The DOM handle field uniquely identifies the XML DOM tree in which the child element will be appended.

WS-DOM-FUNCTION **Input** **Mandatory**

Must be set to **WS-DOM-APPEND-CHILD-BEFORE**.

WS-DOM-PLACE-HOLDER **Place holder** **Mandatory**

WS-DOM-TAG-NAME **Input** **Mandatory**

Populate this field with name of the element. The name of the element must end with a space for delimiting purposes.

WS-DOM-VALUE **Input** **Mandatory**

This field can be used to set the text node while appending the element node in the same call. The length of this field is dependent on WS-DOM-VALUE-LENGTH. This variable can be replaced with any other variable of greater length.

WS-DOM-VALUE-LENGTH **Input** **Mandatory**

This field controls the length of the text node to be added to the element. When this value is set to zero, no text node is added.

WS-DOM-PARENT-NODE-ID **Output** **Optional**

This is an optional field. This field contains the unique node identifier of the parent element node to which the child element is appended. This is the same parent element that is shared by the newly appended element and the reference element before which the new node is appended.

WS-DOM-ELEMENT-ID **Input/Output** **Mandatory**

On input, this field must be populated with the element node identifier that uniquely identifies the element node before which the new element will be added. After the call, the output value contains the node identifier of the newly added element node. This variable is a redefinition of WS-DOM-NODE-ID.

The following return codes are possible:

RC Value	Meaning
0	Successful - No messages issued.
+8	Failed The following are the possible values for WS-DOM-MSG field: • Reference node identifier doesn't exist • Input error Possible input errors could be: • The function specified is not supported. • Element tag name not specified.
+12	SOLA DOM API abended. Abend code - XXXX (four character code returned at run time).
+16	WS-DOM-HANDLE doesn't contain a valid handle.

Example

Use WS-DOM-NODE-ID to indicate which node the new element will be inserted before. In the example below, a new element 'Terminal' will be inserted before 'Transaction', which had its node identifier saved in SAVED-

NODE-ID. After the successful completion of the call, the node identifier of the new node will be returned in WS-DOM-ELEMENT-ID.

```
SET  WS-DOM-APPEND-CHILD-BEFORE TO TRUE
MOVE 'Terminal'           TO WS-DOM-TAG-NAME
MOVE SAVED-NODE-ID        TO WS-DOM-ELEMENT-ID
MOVE SPACES               TO WS-DOM-VALUE
MOVE ZERO                 TO WS-DOM-VALUE-LENGTH

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE,
                      WS-DOM-VALUE-LENGTH,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-ELEMENT-ID
```

appendChildNL

BATCH

The appendChildNL function is identical to appendChild, except that it will also insert a newline character into the document before appending the child element node. This feature is useful when you will be writing the XML document to a file and you want to break the document up into lines.

The appendChildNL function appends a new child element to a specified parent element (and inserts a newline character before doing so). Optionally, a text node value can be attached to the element by specifying a value in WS-DOM-VALUE and a value length in WS-DOM-VALUE LENGTH.

The appendChildNL function is used when you are building a new XML document or updating an existing XML document. If you are building a new XML document, then you must have created the document using the createDocument API. If you are updating an existing XML document, then that document must already exist in the DOM tree (your program must be running in batch). The variable WS-DOM-HANDLE must be a valid handle that references the internal control blocks and tree maintained by the SOLA DOM API.

The owning parent element node where the child element will be appended can be specified in one of two ways. First, the parent element node can be specified by populating the WS-DOM-PARENT field with the name of the parent element. If multiple instances of the parent name exist within the XML document, the child element will be appended to the *first* occurrence of the parent element. The selected parent's unique node identifier will be returned in the WS-DOM-PARENT-NODE-ID field. Second, the parent node identifier can be specified by populating the WS-DOM-PARENT-NODE-ID field with the unique node identifier that uniquely identifies a node within the XML DOM. Since the parent's node identifier is more specific than the parent name, the specification of a node identifier in the appendChildNL request will take precedence over a parent name. The SOLA DOM API will consider any value other than null value in the WS-DOM-PARENT-NODE-ID field to be an input that will take precedence over WS-DOM-PARENT.

After the appendChild function call is executed is executed successfully, if the parent is a complexType element, then the child will added as the last child of the element. If the element is not a complexType element, it will become a complex type (with one child).

The appendChildNL function is a batch only function.

```
CALL WS-DOM-API USING
      WS-DOM-RC,        /* Halfword. Return Code */
      WS-DOM-MSG,       /* Char(80). Output msg if return code <> 0 */
```

```
WS-DOM-HANDLE,    /* Pointer.  For internal use */
WS-DOM-FUNCTION, /* Char(25).  The name of the API */
WS-DOM-PARENT,  /* Name of the parent element */
WS-DOM-TAG-NAME, /* Space delimited. Names the child element */
WS-DOM-VALUE,  /* The child text value (if any) */
WS-DOM-VALUE-LENGTH, /* The child text value length */
WS-DOM-PARENT-NODE-ID, /* Unique ID of the parent node */
WS-DOM-ELEMENT-ID /* The child ID will be returned here */
```

Variables:

WS-DOM-RC **Output** **Mandatory**

Returns status.

WS-DOM-MSG **Output** **Mandatory**

When the return code is not zero, this field will contain messages.

WS-DOM-HANDLE **Input** **Mandatory**

For internal use. The DOM handle field uniquely identifies the XML DOM tree in which the child element will be appended.

WS-DOM-FUNCTION **Input** **Mandatory**

Must be set to **WS-DOM-APPEND-CHILDNL**.

WS-DOM-PARENT **Input** **Mandatory**

This field contains the parent element name. The value in this field is mutually exclusive with WS-DOM-PARENT-NODE-ID. When the parent name is duplicate, the SOLA DOM API picks the *first* element in the document that has the duplicate parent name. The selected parent's unique node identifier will be returned in the field WS-DOM-PARENT-NODE-ID after a successful call.

WS-DOM-TAG-NAME **Input** **Mandatory**

Populate this field with name of the element. The name of element must end with a space for delimiting purposes.

WS-DOM-VALUE **Input** **Mandatory**

This field can be used to set the text node while appending the element node in the same call. The length of this field is dependent on WS-DOM-VALUE-LENGTH. This variable can be replaced with any other variable of greater length.

WS-DOM-VALUE-LENGTH **Input** **Mandatory**

This field controls the length of the text node to be added to the element. When this value is set to zero, no text node is added.

WS-DOM-PARENT-NODE-ID **Input/Output** **Optional**

This is an optional field. When specified (any value other than null) in the CALL statement, it takes precedence over WS-DOM-PARENT field. On input (any value other than null is considered input), the SOLA DOM API uses this field to append a child element to the specified parent

node. On output, this field contains the unique node-id of the parent element to which the child element is appended.

WS-DOM-ELEMENT-ID **Output** **Optional**

This is an optional field. When specified in the CALL statement the unique node identifier of the node is returned in this variable. The unique node identifier is an immutable property that uniquely identifies the node that was just appended to the XML DOM. This node identifier can be used in other XML SOLA DOM API functions to uniquely identify the node to be operated on. This variable is a redefinition of WS-DOM-NODE-ID.

The following return codes are possible:

RC Value	Meaning
0	Successful - No messages issued.
+8	Failed. The following are the possible values for WS-DOM-MSG field: • Parent not found • Parent node identifier does not exist • Input error Possible input errors could be • The function specified is not supported • Element tag name not specified
+12	SOLA DOM API abended. Abend code - XXXX (four character code returned at run time).
+16	WS-DOM-HANDLE doesn't contain a valid handle.

Example

The following example appends the "Transaction" child element to the first occurrence of the "ResponseData" parent element in the XML document. Since the parent is specified by name the WS-DOM-PARENT-NODE-ID parameter is not used and is not specified. Additionally, the optional return value WS-DOM-NODE-ID is not used and is not specified.

```
SET  WS-DOM-APPEND-CHILDNL TO TRUE
MOVE 'Transaction'         TO WS-DOM-TAG-NAME
MOVE 'ResponseData'        TO WS-DOM-PARENT
MOVE SPACES                TO WS-DOM-VALUE
MOVE ZERO                  TO WS-DOM-VALUE-LENGTH

CALL WS-DOM-API USING WS-DOM-RC,
```

```
                                WS-DOM-MSG,
                                WS-DOM-HANDLE,
                                WS-DOM-FUNCTION,
                                WS-DOM-PARENT,
                                WS-DOM-TAG-NAME,
                                WS-DOM-VALUE,
                                WS-DOM-VALUE-LENGTH
```

The next example is the same as the previous example in that it appends the "Transaction" child element to the first occurrence of the "ResponseData" parent element in the XML document. In this example the node identifier of the node to be appended to ("ResponseData", the parent node) has its unique node identifier saved in SAVED-PARENT-ID. This value is moved to WS-DOM-PARENT-NODE-ID, which is used to uniquely identify "ResponseData" as the parent to be appended to.

WS-DOM-PARENT is not used in this case and is replaced by WS-DOM-PLACE-HOLDER.

```
SET  WS-DOM-APPEND-CHILDNL TO TRUE
MOVE 'Transaction'         TO WS-DOM-TAG-NAME
MOVE SAVED-PARENT-ID       TO WS-DOM-PARENT-NODE-ID
MOVE SPACES                TO WS-DOM-VALUE
MOVE ZERO                  TO WS-DOM-VALUE-LENGTH

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE,
                      WS-DOM-VALUE-LENGTH,
                      WS-DOM-PARENT-NODE-ID,
                      WS-DOM-NODE-ID
```

appendTextNode

CICS BATCH

The appendTextNode function is used to add a child text node to an element node. Please note that appendTextNode is used only to add a text node to an element node. In order to add text to an attribute node use the setAttribute function.

If an element is a complexType and does not have a text node and a text node is added to it, the XML will become a mixed type XML. The text node will be added after the last child.

If a text node already exists for the element, a return code of +3 will be returned and no action will be taken.

The specification of a parent node identifier as search criteria for the appendTextNode request overrides any parent name specified as search criteria. The SOLA DOM API will consider any value other than a null value in the WS-DOM-PARENT-NODE-ID field to be an input value and will ignore WS-DOM-PARENT.

In the XML fragment below, the element "School" has two attributes, "Location" and "Type". It has no text node.

```
<School Location="NJ" Type="Public"></School>
```

The appendTextNode function can be used to attach a text node to an element. As can be seen in the XML fragment below, the "School" element has a text node value of "Rutgers".

```
<School Location="NJ" Type="Public">Rutgers</School>
```

The appendTextNode function can be used in CICS and batch.

```
CALL WS-DOM-API USING
      WS-DOM-RC,         /* Halfword. Return Code */
      WS-DOM-MSG,        /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,     /* Pointer.  For internal use */
      WS-DOM-FUNCTION,   /* Char(25).  The name of the API */
      WS-DOM-PARENT,     /* The tag name of the parent node */
      WS-DOM-PLACE-HOLDER, /* Placeholder */
      WS-DOM-VALUE,      /* The text value of the node to be added */
      WS-DOM-VALUE-LENGTH, /* Length of the text value */
      WS-DOM-PARENT-NODE-ID /* Unique id of the parent node */
```

Variables:

WS-DOM-RC **Output** **Mandatory**

Returns status.

WS-DOM-MSG **Output** **Mandatory**

When the return code is not zero, this field will contain messages.

WS-DOM-HANDLE **Input** **Mandatory**

For internal use. The DOM handle field uniquely identifies the XML DOM tree in which the text node will be appended.

WS-DOM-FUNCTION **Input** **Mandatory**

Must be set to **WS-DOM-APPEND-TEXT-NODE**.

WS-DOM-PARENT **Input** **Mandatory**

The SOLA DOM API will use this variable to search for a parent element with a matching name. If there are duplicates, the *first* parent with a matching name will be selected. The selected parent's unique node identifier will be returned in the field WS-DOM-PARENT-NODE-ID after a successful call.

WS-DOM-PLACE-HOLDER **Place holder** **Mandatory**

WS-DOM-VALUE **Input** **Mandatory**

Populate this variable with a text string and the SOLA DOM API will use this value to append the text node to an element node.

WS-DOM-VALUE-LENGTH **Input** **Mandatory**

Specify the length of text string to be appended.

WS-DOM-PARENT-NODE-ID **Input/Output** **Optional**

This field is optional. When specified (any value other than null) in the CALL statement, it takes precedence over WS-DOM-PARENT field. On input (any value other than null is considered input), the SOLA DOM API uses this field to append a text node to the specified parent node. On output, this field contains the unique node-id of the parent element to which the child text node is appended.

The following return codes are possible:

RC Value	Meaning
0	Successful - No messages issued.
+3	Text node already exists. No action taken.
+8	Failed. The following are the possible values for WS-DOM-MSG field: • Parent not found • Specified parent node doesn't exist • Input error Possible input errors could be: • The function specified is not supported. • Neither parent tag name nor parent node identifier are specified.

RC Value	Meaning
+12	SOLA DOM API abended. Abend code - XXXX (four character code returned at run time).
+16	WS-DOM-HANDLE doesn't contain a valid handle.

Example

In the example below a text node value is being appended. The element to have a text node appended to it is identified by the unique node identifier of the parent node (in this case "Transaction") that was stored in SAVED-PARENT-NODE. The value of the text node is "TR01" which has a length of 4 bytes.

```
SET  WS-DOM-APPEND-TEXT-NODE TO TRUE
MOVE SAVED-PARENT-NODE       TO WS-DOM-PARENT-NODE-ID
MOVE 'TR01'                  TO WS-DOM-VALUE
MOVE +4                      TO WS-DOM-VALUE-LENGTH

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-VALUE,
                      WS-DOM-VALUE-LENGTH,
                      WS-DOM-PARENT-NODE-ID
```

In this next example the same text node value is being appended. This time the element node to have a text node appended to it is identified by specifying the tag name of the parent node (in this case "Transaction"). If there is more than one element that matches this criterion then the text node will be appended to the *first* element. The value of the text node is "TR01" which has a length of 4 bytes.

```
SET  WS-DOM-UPDATE-TEXT-NODE TO TRUE
MOVE 'Transaction'           TO WS-DOM-PARENT
MOVE 'TR01'                  TO WS-DOM-VALUE
MOVE +4                      TO WS-DOM-VALUE-LENGTH

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PARENT,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-VALUE,
                      WS-DOM-VALUE-LENGTH
```

createDocument

CICS BATCH

The createDocument function creates the root element of a document. Whenever you need to create a new XML document, use this function. When creating a new XML document, before you can add elements or attributes to the document you must first create the internal XML tree and control blocks using the createDocument function.

The name of the root element of the document is specified in the WS-DOM-TAG-NAME variable. This value will be used as the parent value in subsequent appendChild functions.

The WS-DOM-CONTROL variable is used to specify special document processing instructions.

Pay special attention to the pointer variable, WS-DOM-HANDLE. This is a pointer variable that the SOLA DOM API uses to manage the storage (XML tree and control blocks) associated with an XML document. On the first call to the SOLA DOM API, you should set WS-DOM-HANDLE to null. This will force the SOLA DOM API to acquire storage and return you a valid handle in WS-DOM-HANDLE. To continue working on a given document the same handle must be passed to the SOLA DOM API on subsequent calls. If you are creating multiple documents in sequence (as you might do in a batch job) then you can reuse the handle on subsequent createDocument calls to force the SOLA DOM API to reuse the acquired storage. If you no longer need the storage acquired (or the tree itself), you can use the freeStorage function call. If you need to manipulate multiple XML documents simultaneously, you can do this by maintaining multiple handles.

The createDocument function can be used in CICS and batch.

```
CALL WS-DOM-API USING
      WS-DOM-RC,         /* Halfword. Return Code */
      WS-DOM-MSG,        /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,     /* Pointer.  For internal use */
      WS-DOM-FUNCTION,   /* Char(25).  The name of the API */
      WS-DOM-CONTROL,    /* Specifies special processing */
      WS-DOM-TAG-NAME,   /* Space delimited. Names the root element */
      WS-DOM-PLACE-HOLDER, /* Placehoder. */
      WS-DOM-PLACE-HOLDER,  /* Placehoder. */
      WS-DOM-PLACE-HOLDER,  /* Placehoder. */
      WS-DOM-ELEMENT-ID /* The node id of the root will be returned */
```

When coding the parameters for this API call please refer to the Invocation of Services section on page 16.

Variables:

WS-DOM-RC **Output** **Mandatory**

Returns status.

WS-DOM-MSG **Output** **Mandatory**

When the return code is not zero, this field will contain messages.

WS-DOM-HANDLE **Input/Output** **Mandatory**

For internal use. Must be set to null in the very first call. The SOLA DOM API populates this variable, when null, to be used in subsequent calls. The handle can be reused for multiple XML document creation. If you wish to manipulate multiple documents concurrently you can do this by maintaining multiple DOM handles. The DOM handle returned by the createDocument function uniquely defines the XML DOM tree that subsequent functions operate on.

WS-DOM-FUNCTION **Input** **Mandatory**

Must be set to **WS-DOM-CREATE-DOC**.

WS-DOM-CONTROL **Input** **Mandatory**

WS-DOM-CONTROL is used with the createDocument function to add comments in the beginning of the XML document. WS-DOM-CONTROL will honor the following values:

0 No special processing is performed.

2 Passing this value will add the following at the beginning of the XML:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

3 Passing this value will add the following at the beginning of the XML:

```
<?xml version="1.0" encoding="cp037" ?>
```

6 Passing this value will add the following at the beginning of the XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
```

7 Passing this value will add the following at the beginning of the XML:

```
<?xml version="1.0" encoding="cp037" standalone="yes" ?>
```

WS-DOM-TAG-NAME **Input** **Mandatory**

Contains the name of the root element.

WS-DOM-PLACE-HOLDER **Place holder** **Optional**

```
WS-DOM-PLACE-HOLDER          Place holder            Optional

WS-DOM-PLACE-HOLDER          Place holder            Optional

WS-DOM-ELEMENT-ID            Output                  Optional
```

After the call, the output value will contain the unique node identifier of the newly added root element node. This variable is a redefinition of WS-DOM-NODE-ID.

The following return codes are possible:

RC Value	Meaning
0	Successful - No messages issued.
+8	Failed. The following are the possible values for WS-DOM-MSG field: • Not enough memory to create the tree • Invalid DOM Control Value • Input error Possible input errors could be: • The function specified is not supported. • Root element tag name not specified.
+12	SOLA DOM API abended. Abend code - XXXX (four character code returned at run time).
+16	WS-DOM-HANDLE doesn't contain a valid handle.

Example

In the following example we are creating a new XML document. The root element of the document is called "ResponseData".

We are setting WS-DOM-HANDLE to null (we are telling the SOLA DOM API to acquire storage for the XML tree and control blocks). If we had wanted to reuse the storage from an already existing DOM tree we could have used an already valid WS-DOM-HANDLE.

Because we don't want special processing WS-DOM-CONTROL is set to zero.

```
SET  WS-DOM-CREATE-DOC TO TRUE
MOVE 'ResponseData'    TO WS-DOM-TAG-NAME
SET  WS-DOM-HANDLE     TO NULL
MOVE ZERO              TO WS-DOM-CONTROL

CALL WS-DOM-API USING
      WS-DOM-RC,
      WS-DOM-MSG,
```

```
      WS-DOM-HANDLE,
      WS-DOM-FUNCTION,
      WS-DOM-CONTROL,
      WS-DOM-TAG-NAME
```

In the following example we are creating a new XML document. The root element of the document is called "Widget".

We want to reuse the storage from an already existing DOM tree, therefore we are supplying a used and already valid WS-DOM-HANDLE that was saved in WS-SAVED-DOM-HANDLE.

Because we don't want special processing WS-DOM-CONTROL is set to zero.

After the call, we will receive the unique node identifier of the newly created root element.

```
SET  WS-DOM-CREATE-DOC TO TRUE
MOVE 'Widget'          TO WS-DOM-TAG-NAME
SET WS-DOM-HANDLE      TO WS-SAVED-DOM-HANDLE
MOVE ZERO              TO WS-DOM-CONTROL

CALL WS-DOM-API USING
      WS-DOM-RC,
      WS-DOM-MSG,
      WS-DOM-HANDLE,
      WS-DOM-FUNCTION,
      WS-DOM-CONTROL,
      WS-DOM-TAG-NAME,
      WS-DOM-PLACE-HOLDER,
      WS-DOM-PLACE-HOLDER,
      WS-DOM-PLACE-HOLDER,
      WS-DOM-ELEMENT-ID
```

finalize

CICS BATCH

The finalize function is used to finalize and optionally retrieve a newly created XML document. Finalize will convert the DOM tree into an XML stream, as it does so all open tags will be closed and the document will be made ready to be externalized. The finalize function returns a pointer to the XML document. The SOLA DOM API will acquire storage for the XML document and will then convert the DOM tree into an XML stream, storing it in the acquired storage. WS-DOM-VALUE-LENGTH will contain the length of the XML document. The finalize function does not release the storage used for the document, the tree and its own control block structure. One way to release the storage acquired is by using the freeStorage (see miscellaneous functions on page - 139) function call.

Once finalized, the document can be accessed with the address pointer. Alternatively, the document can be accessed using the retrieve function.

If you experience a S878-10 abend in batch then a likely cause could be failing to use the freeStorage function to release storage while finalizing multiple documents.

The finalize function can be used in CICS and batch.

```
CALL WS-DOM-API USING
      WS-DOM-RC,        /* Halfword. Return Code */
      WS-DOM-MSG,       /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,    /* Pointer.  For internal use */
      WS-DOM-FUNCTION,  /* Char(25).  The name of the API */
      WS-DOM-PLACE-HOLDER, /* Placeholder */
      WS-DOM-PLACE-HOLDER, /* Placeholder */
      WS-DOM-VALUE-PTR,      /* Output pointer to the XML document */
      WS-DOM-VALUE-LENGTH /* Output length of the XML document */
```

Variables:

WS-DOM-RC **Output** **Mandatory**

Returns status.

WS-DOM-MSG **Output** **Mandatory**

When the return code is not zero, this field will contain messages.

WS-DOM-HANDLE **Input** **Mandatory**

For internal use. The DOM handle field uniquely identifies the XML DOM tree that will be finalized.

WS-DOM-FUNCTION **Input** **Mandatory**

Must be set to **WS-DOM-FINALIZE**.

WS-DOM-PLACE-HOLDER **Place holder** **Mandatory**

WS-DOM-PLACE-HOLDER **Place Holder** **Mandatory**

WS-DOM-VALUE-PTR **Output** **Mandatory**

The SOLA DOM API will acquire storage and populate the area with an XML string (the completed XML document). This variable contains the pointer to the XML document.

WS-DOM-VALUE-LENGTH **Output** **Mandatory**

The SOLA DOM API will populate this variable with the length of the newly built XML document.

The following return codes are possible:

RC Value	Meaning
0	Successful - No messages issued.
+8	Failed. The following are the possible values for WS-DOM-MSG field: • Input error Possible input errors could be: • The function specified is not supported.
+12	SOLA DOM API abended. Abend code - XXXX (four character code returned at run time).
+16	WS-DOM-HANDLE doesn't contain a valid handle.

Example

In this example we are using the finalize function to complete the XML document and return a pointer (WS-DOM-VALUE-PTR) to the newly created document. Exercise caution when using finalize as it does not release the the storage acquired by the SOLA DOM API. This storage can be released with the freeStorage (refer to page 139) function.

```
SET  WS-DOM-FINALIZE TO TRUE

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-VALUE-PTR,
                      WS-DOM-VALUE-LENGTH
```

getXMLLength

BATCH

The getXMLLength function retrieves the length of the completed XML document. The getXMLLength function is useful if you need to determine the amount of storage you will need to store the XML document.

This function can be executed at any time after the createDocument function has been executed.

The getXMLLength function only works in batch. It is designed to work with the appendChildNL, and retrieve APIs in batch.

```
CALL WS-DOM-API USING
      WS-DOM-RC,         /* Halfword. Return Code */
      WS-DOM-MSG,        /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,     /* Pointer.  For internal use */
      WS-DOM-FUNCTION,   /* Char(25).  The name of the API */
      WS-DOM-PLACE-HOLDER, /* placeholder */
      WS-DOM-PLACE-HOLDER, /* placeholder */
      WS-DOM-PLACE-HOLDER, /* placeholder */
      WS-DOM-VALUE-LENGTH /* Length of the XML doc */
```

Variables:

WS-DOM-RC **Output** **Mandatory**

Returns status.

WS-DOM-MSG **Output** **Mandatory**

When the return code is not zero, this field will contain messages.

WS-DOM-HANDLE **Input** **Mandatory**

For internal use. The DOM handle field uniquely identifies the XML DOM tree from which the XML length will be retrieved.

WS-DOM-FUNCTION **Input** **Mandatory**

Must be set to **WS-DOM-GET-XML-LEN**.

WS-DOM-PLACE-HOLDER **Place holder** **Mandatory**

WS-DOM-PLACE-HOLDER **Place holder** **Mandatory**

WS-DOM-PLACE-HOLDER **Place Holder** **Mandatory**

WS-DOM-VALUE-LENGTH **Output** **Mandatory**

This field will return the length of the XML at the time of the call.

The following return codes are possible:

RC Value	Meaning
0	Successful - No messages issued.

RC Value	Meaning
+8	Failed. The following are the possible values for WS-DOM-MSG field: • Input error • This function is only supported in batch mode Possible input errors could be: • The function specified is not supported.
+12	SOLA DOM API abended. Abend code - XXXX (four character code returned at run time).
+16	WS-DOM-HANDLE doesn't contain a valid handle.

Example

In this example, we are getting the length of the XML document referenced by WS-DOM-HANDLE.

```
SET  WS-DOM-GET-XML-LEN TO TRUE

CALL WS-DOM-API USING WS-DOM-RC,
       WS-DOM-MSG,
       WS-DOM-HANDLE,
       WS-DOM-FUNCTION,
       WS-DOM-PLACE-HOLDER,
       WS-DOM-PLACE-HOLDER,
       WS-DOM-PLACE-HOLDER,
       WS-DOM-VALUE-LENGTH
```

removeNode

CICS BATCH

The removeNode function removes any type of node (element, attribute or namespace node) from an XML document. The node to be removed is identified by the node's unique identifier, which is referenced in WS-DOM-NODE-ID. The type of node to be removed is specified in WS-DOM-NODE-TYPE.

If the node to be removed is a complex type element then the node and all nodes that depend on that node (its children) will be removed from the tree. If an attribute node is removed, it is removed from its parent, and the parent no longer has that attribute attached to it. If a namespace node is removed, all elements that reference the removed namespace will use the default namespace.

The removeNode function cannot remove text nodes. To remove textNodes, use the updateTextNode function.

The allowable values for WS-DOM-NODE-TYPE are:

Node Type	Value
Element	1
Attribute	2
Namespace	3

The removeNode function can be used in CICS and batch.

```
CALL WS-DOM-API USING
      WS-DOM-RC,        /* Halfword. Return Code */
      WS-DOM-MSG,       /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,    /* Pointer.  For internal use */
      WS-DOM-FUNCTION,  /* Char(25).  The name of the API */
      WS-DOM-NODE-ID,   /* Unique ID of the node to be removed */
      WS-DOM-NODE-TYPE  /* Type of the node to be removed */
```

Variables:

WS-DOM-RC **Output** **Mandatory**

Returns status.

WS-DOM-MSG **Output** **Mandatory**

When the return code is not zero, this field will contain messages.

WS-DOM-HANDLE **Input** **Mandatory**

For internal use. The DOM handle field uniquely identifies the XML DOM tree in which the node will be removed.

WS-DOM-FUNCTION **Input** **Mandatory**

Must be set to **WS-DOM-REMOVE-NODE**.

WS-DOM-NODE-ID **Input** **Mandatory**

Specify the unique node identifier representing the node identifier of the element, attribute or namespace to be removed. To remove a text node of an element node use the updateTextNode function and set the text length to zero. To remove the text of an attribute node use the updateAttribute function and set the text length to zero.

WS-DOM-NODE-TYPE **Input** **Mandatory**

Set this variable to the appropriate node type; 1 for an element, 2 for an attribute or 3 for a namespace node.

The following return codes are possible:

RC Value	Meaning
0	Successful - No messages issued.
+8	Failed The following are the possible values for WS-DOM-MSG field: • Node identifier doesn't exist • Input error Possible input errors could be: • The function specified is not supported. • Node type not recognized.
+12	SOLA DOM API abended. Abend code - XXXX (four character code returned at run time).
+16	WS-DOM-HANDLE doesn't contain a valid handle.

Example

In the example below, an element node is being removed. An element node is specified by setting WS-DOM-ELEMENT-NODE. The element node to be removed is referenced by the node identifier that has been saved in SAVED-NODE-ID.

```
SET  WS-DOM-REMOVE-NODE  TO TRUE
SET  WS-DOM-ELEMENT-NODE TO TRUE
MOVE SAVED-NODE-ID       TO WS-DOM-NODE-ID

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-NODE-ID,
                      WS-DOM-NODE-TYPE
```

retrieve

CICS BATCH

The retrieve function retrieves (gets) XML records.

The retrieve function finalizes the document and then gets the complete (if appendChildNL was not used) document and copies it to a data area that you provide. On zero return code, the SOLA DOM API will then release any storage space used for internal usage such as storage for storing the DOM tree and the control blocks etc. After retrieve has completed, the WS-DOM-HANDLE will no longer be valid.

If appendChildNL was used then the retrieve function will fetch the document in pieces, each piece ending in a newline. In this case you will receive a return code of +1 when the data you are retrieving is delimited by a newline and you will have to call the retrieve function repeatedly until you receive a zero return code. The storage occupied by the SOLA DOM API will be released once the entire document has been retrieved and a zero return code has been issued.

The retrieve function can be used in CICS and batch.

```
CALL WS-DOM-API USING
      WS-DOM-RC,        /* Halfword. Return Code */
      WS-DOM-MSG,       /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,    /* Pointer.  For internal use */
      WS-DOM-FUNCTION, /* Char(25).  The name of the API */
      WS-DOM-PLACE-HOLDER, /* placeholder */
      WS-DOM-PLACE-HOLDER, /* placeholder */
      your-data-area,     /* The document will be returned here */
      WS-DOM-VALUE-LENGTH, /* Length of the returned document */
      WS-DOM-PLACE-HOLDER, /* placeholder */
      WS-DOM-PLACE-HOLDER, /* placeholder */
      WS-DOM-PLACE-HOLDER, /* placeholder */
      WS-DOM-PLACE-HOLDER, /* placeholder */
      WS-DOM-PLACE-HOLDER, /* placeholder */
      WS-DOM-MAX-LENGTH /* Can specify max length of data here */
```

Variables:

WS-DOM-RC **Output** **Mandatory**

Returns status.

WS-DOM-MSG **Output** **Mandatory**

When the return code is not zero, this field will contain messages.

WS-DOM-HANDLE **Input** **Mandatory**

For internal use. The DOM handle field uniquely identifies the XML DOM tree from which XML will be retrieved.

WS-DOM-FUNCTION **Input** **Mandatory**

Must be set to **WS-DOM-RETRIEVE**.

WS-DOM-PLACE-HOLDER **Place holder** **Mandatory**

WS-DOM-PLACE-HOLDER **Place Holder** **Mandatory**

Your-data-Area **Output** **Mandatory**

The SOLA DOM API will populate this variable with the retrieved XML string.

WS-DOM-VALUE-LENGTH **Output** **Mandatory**

The SOLA DOM API will populate this variable with the length of the newly built XML.

WS-DOM-PLACE-HOLDER **Place holder** **Mandatory**

WS-DOM-PLACE-HOLDER **Place Holder** **Mandatory**

WS-DOM-PLACE-HOLDER **Place holder** **Mandatory**

WS-DOM-PLACE-HOLDER **Place Holder** **Mandatory**

WS-DOM-PLACE-HOLDER **Place Holder** **Mandatory**

WS-DOM-MAX-LENGTH **Input** **Optional**

This optional field, when specified, tells the SOLA DOM API to truncate data in the WS-DOM-VALUE field (to WS-DOM-MAX-LENGTH) if WS-DOM-VALUE-LENGTH exceeds WS-DOM-MAX-LENGTH. A return code of +5 is returned if the returned data is truncated. In this case, WS-DOM-VALUE-LENGTH contains the full length of the data but the data in your data area is truncated to WS-DOM-MAX-LENGTH. Subsequent calls to the retrieve function will retrieve the data that was truncated (data will not be lost as a result of truncation).

The following return codes are possible:

RC Value	Meaning
0	Successful - No messages issued.
+1	Returned data is truncated at a newline character. More data is available.
+5	Returned data is bigger than allowed. Data truncated to the specified Length.

RC Value	Meaning
+8	Failed. The following are the possible values for WS-DOM-MSG field: • Input error Possible input errors could be: • The function specified is not supported.
+12	SOLA DOM API abended. Abend code - XXXX (four character code returned at run time).
+16	WS-DOM-HANDLE doesn't contain a valid handle.

Example

In the example below, a completed XML document is being retrieved into WS-DATA-AREA. On successful completion of the retrieve function, the internal storage will be released and WS-DOM-HANDLE will no longer be valid. WS-DATA-AREA is 32,000 bytes long so WS-DOM-MAX-LENGTH is set to 32,000 bytes. If the XML document exceeds 32,000 bytes then the first 32,000 bytes are retrieved into WS-DATA-AREA and WS-DOM-RC is set to +5.

```
SET  WS-DOM-RETRIEVE TO TRUE
MOVE 32000           TO WS-DOM-MAX-LENGTH
CALL WS-DOM-API USING WS-DOM-RC,
       WS-DOM-MSG,
       WS-DOM-HANDLE,
       WS-DOM-FUNCTION,
       WS-DOM-PLACE-HOLDER,
       WS-DOM-PLACE-HOLDER,
       WS-DATA-AREA,
       WS-DOM-VALUE-LENGTH,
       WS-DOM-PLACE-HOLDER,
       WS-DOM-PLACE-HOLDER,
       WS-DOM-PLACE-HOLDER,
       WS-DOM-PLACE-HOLDER,
       WS-DOM-PLACE-HOLDER,
       WS-DOM-MAX-LENGTH

IF WS-DOM-RC = 5 THEN
  ...
END-IF
```

In this second example below the appendChildNL function was used while building the XML. The retrieve function honors the newline characters that were inserted by the appendChildNL function and returns the document line

by line. The retrieve function is called repeatedly until the return code is not equal to +1.

```
SET  WS-DOM-RETRIEVE TO TRUE
PERFORM WHILE WS-DOM-RC = 1
    CALL WS-DOM-API USING WS-DOM-RC,
           WS-DOM-MSG,
           WS-DOM-HANDLE,
           WS-DOM-FUNCTION,
           WS-DOM-PLACE-HOLDER,
           WS-DOM-PLACE-HOLDER,
           WS-DATA-AREA

END-PERFORM
```

setAttribute

CICS BATCH

The setAttribute function is used to add and attach attributes (name/value pairs) to an element. The element to have attributes attached to it can be identified either by specifying the parent's name (in WS-DOM-PARENT) or by using the unique node identifier for the parent in WS-DOM-PARENT-NODE-ID.

If you have specified the parent node using WS-DOM-PARENT field with the name of the parent element and multiple instances of the parent name exist within the XML document, then the attribute will be appended to the *first* occurrence of the parent element. The selected parent's unique node identifier will be returned in the WS-DOM-PARENT-NODE-ID field.

Alternatively, the parent node identifier can be specified by populating the WS-DOM-PARENT-NODE-ID field with the unique node identifier of the parent element node. Since the node identifier is more specific than the parent name, the specification of a node identifier in the setAttribute request will take precedence over the parent name. The SOLA DOM API will consider any value other than null value in the WS-DOM-PARENT-NODE-ID field to be an input that will take precedence over WS-DOM-PARENT.

The setAttribute function can be used in CICS and batch.

```
CALL WS-DOM-API USING
      WS-DOM-RC,         /* Halfword. Return Code */
      WS-DOM-MSG,        /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,     /* Pointer.  For internal use */
      WS-DOM-FUNCTION,   /* Char(25).  The name of the API */
      WS-DOM-PARENT,     /* The tag name of the parent node */
      WS-DOM-TAG-NAME,   /* The attribute tag name to be added */
      WS-DOM-VALUE,      /* The text value of the node to be added */
      WS-DOM-VALUE-LENGTH, /* Length of the text value */
      WS-DOM-PARENT-NODE-ID, /* Unique id of the parent node */
      WS-DOM-ATTRIBUTE-ID /* The node identifier will be returned here
      */
```

Variables:

WS-DOM-RC **Output** **Mandatory**

Returns status.

WS-DOM-MSG **Output** **Mandatory**

When the return code is not zero, this field will contain messages.

WS-DOM-HANDLE **Input** **Mandatory**

For internal use. The DOM handle field uniquely identifies the XML DOM tree in which the attribute will be set.

WS-DOM-FUNCTION **Input** **Mandatory**
Must be set to **WS-DOM-SET-ATTRIBUTE**.

WS-DOM-PARENT **Input** **Mandatory**
SOLA DOM API will use this variable to search for a parent element with a matching name. If there are duplicates, then the *first* parent with a matching name will be picked. The selected parent's unique node identifier will be returned in the field WS-DOM-PARENT-NODE-ID after a successful call.

WS-DOM-TAG-NAME **Input** **Mandatory**
Populate this field with name of the attribute. The name of the attribute must end with a space for delimiting purposes.

WS-DOM-VALUE **Input** **Mandatory**
Populate this variable with a text string and SOLA DOM API will add this value as the text of the attribute node.

WS-DOM-VALUE-LENGTH **Input** **Mandatory**
Specify the length of the text string to be added.

WS-DOM-PARENT-NODE-ID **Input/Output** **Optional**
This field is optional. This field contains the unique node identifier of the parent node. When specified (any value other than null) in the CALL statement, it takes precedence over WS-DOM-PARENT field. On input (any value other than null is considered input), the SOLA DOM API uses this field to add and attach a child attribute to the specified parent node. On output, this field contains the unique parent-node-id of the parent element to which the child attribute is attached.

WS-DOM-ATTRIBUTE-ID **Output** **Optional**
This field, when specified in the CALL, will be used as an output field for the SOLA DOM API. It will be populated with the unique node identifier of the newly created attribute node. This variable is a redefinition of WS-DOM-NODE-TYPE.

The following return codes are possible:

RC Value	Meaning
0	Successful - No messages issued.

RC Value	Meaning
+8	Failed The following are the possible values for WS-DOM-MSG field: • Parent not found • Specified parent node doesn't exist • Input error Possible input errors could be: • The function specified is not supported. • Attribute tag name not specified.
+12	SOLA DOM API abended. Abend code - XXXX (four character code returned at run time).
+16	WS-DOM-HANDLE doesn't contain a valid handle.

Example

In the example below an attribute node is being added to the element node named "ResponseData". If more than one "ResponseData" element exists in the DOM tree then the attribute will be added to the *first* instance of that element in the tree. The attribute name is "Program" and the value of the attribute is "CICS1234".

```
SET  WS-DOM-SET-ATTRIBUTE TO TRUE
MOVE 'Program'            TO WS-DOM-TAG-NAME
MOVE 'ResponseData'       TO WS-DOM-PARENT
MOVE 'CICS1234'           TO WS-DOM-VALUE
MOVE +8                   TO WS-DOM-VALUE-LENGTH

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PARENT,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE,
                      WS-DOM-VALUE-LENGTH
```

This second example below is functionally equivalent to the example above in that an attribute node is being added to the element node named "ResponseData". In this example the element node is referenced by its node id, not by name. Because the SOLA DOM API detects that the parent node identifier is being specified it will ignore the value in the variable WS-DOM-PARENT. For reasons of clarity the example below shows that WS-DOM-PLACE-HOLDER is used in place of WS-DOM-PARENT.

```
SET  WS-DOM-SET-ATTRIBUTE TO TRUE
MOVE 'Program'            TO WS-DOM-TAG-NAME
```

```
MOVE SAVED-PARENT-ID       TO WS-DOM-PARENT-NODE-ID
MOVE 'CICS1234'            TO WS-DOM-VALUE
MOVE +8                    TO WS-DOM-VALUE-LENGTH

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE,
                      WS-DOM-VALUE-LENGTH
                      WS-DOM-PARENT-NODE-ID,
                      WS-DOM-ATTRIBUTE-ID
```

setNamespace

CICS BATCH

The setNamespace function is used to add and attach a namespace (when not present) or just attach a namespace (when present) to an element. The element that is to have a namespace attached to it can be identified either by specifying the element's name (in WS-DOM-PARENT) or by using its unique node identifier in WS-DOM-PARENT-NODE-ID.

If you have specified the parent node name using WS-DOM-PARENT and multiple instances of the parent name exist within the XML document, then the namespace will be appended to the *first* occurrence of the parent element. The selected parent's unique node identifier will be returned in the WS-DOM-PARENT-NODE-ID field.

Alternatively, the parent node identifier can be specified by populating the WS-DOM-PARENT-NODE-ID field with the unique node identifier of the parent element node. Since the node identifier is more specific than the parent name, the specification of a node identifier in the setNamespace request will take precedence over the parent name.

The setNamespace function can be used in CICS and batch.

```
CALL WS-DOM-API USING
      WS-DOM-RC,         /* Halfword. Return Code */
      WS-DOM-MSG,        /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,     /* Pointer.  For internal use */
      WS-DOM-FUNCTION,   /* Char(25).  The name of the API */
      WS-DOM-PARENT,     /* The tag name of the parent node */
      WS-DOM-NMS-ALIAS,  /* The alias name to be added */
      WS-DOM-NAMESPACE,      /* The value of namespace to be added */
      WS-DOM-VALUE-LENGTH,  /* Length of the namespace value */
      WS-DOM-PARENT-NODE-ID, /* Unique id of the parent node */
      WS-DOM-PLACE-HOLDER,  /* placeholder */
      WS-DOM-NMS-NODE-ID  /* The ID of the namespace node */
```

Variables:

WS-DOM-RC **Output** **Mandatory**

Returns status.

WS-DOM-MSG **Output** **Mandatory**

When the return code is not zero, this field will contain messages.

WS-DOM-HANDLE **Input** **Mandatory**

For internal use. The DOM handle field uniquely identifies the XML DOM tree in which the namespace will be set.

WS-DOM-FUNCTION **Input** **Mandatory**

Must be set to **WS-DOM-SET-NAMESPACE**.

WS-DOM-PARENT **Input** **Mandatory**

SOLA DOM API will use this variable to search for the parent element with a matching name. If there are duplicates, then the *first* parent with a matching name will be picked. The parent's unique node identifier of the selected parent will be returned in the field WS-DOM-PARENT-NODE-ID after successful call.

WS-DOM-NMS-ALIAS **Input** **Mandatory**

This field contains the value of the namespace alias that will be added and attached to an element. This field can contains spaces, when alias is not needed.

WS-DOM-NAMESPACE **Input** **Mandatory**

This field contains the value of the namespace that will be associated with the element node. This field can contain spaces when WS-DOM-NMS-NODE-ID is used as input. WS-DOM-PLACE-HOLDER can optionally be used to indicate absence of WS-DOM-NAMESPACE.

WS-DOM-VALUE-LENGTH **Input** **Mandatory**

Specify the length of the namespace text string to be added.

WS-DOM-PARENT-NODE-ID **Input/Output** **Optional**

This field is optional. This field contains the unique node identifier of the parent node. When specified (any value other than null) in the CALL statement, it takes precedence over WS-DOM-PARENT field. On input (any value other than null is considered input), the SOLA DOM API uses this field to add and attach (or just attach when present) a namespace node to the specified parent node. On output, this field contains the unique parent-node-id of the parent element to which the namespace was attached.

WS-DOM-PLACE-HOLDER **Place holder** **Optional**

WS-DOM-NMS-NODE-ID **Input** **Optional**

This field is optional. Specify the unique node identifier representing the node identifier of the namespace node and the SOLA DOM API will search for a namespace for this given node identifier to attach the parent element. When specified (value other than null) in the CALL statement, it takes precedence over WS-DOM-NAMESPACE and WS-DOM-NMS-ALIAS fields.

The following return codes are possible:

RC Value	Meaning
0	Successful - No messages issued.

RC Value	Meaning
+8	Failed The following are the possible values for WS-DOM-MSG field: • Parent not found • Specified parent node doesn't exist • Specified namespace node doesn't exist • Input error Possible input errors could be: • The function specified is not supported. • The namespace value is not specified
+12	SOLA DOM API abended. Abend code - XXXX (four character code returned at run time).
+16	WS-DOM-HANDLE doesn't contain a valid handle.

Example

In the example below, a namespace node is being added to the element node named "ResponseData". If more than one "ResponseData" elements exists in the DOM tree, then the namespace will be added to the *first* instance of that element in the tree. The namespace alias is "sola" and the value of the namespace is "http://www.sola.com".

```
SET  WS-DOM-SET-NAMESPACE TO TRUE
MOVE 'sola'                TO WS-DOM-NMS-ALIAS
MOVE 'ResponseData'        TO WS-DOM-PARENT
MOVE 'http://www.sola.com' TO WS-DOM-NAMESPACE
MOVE +19                   TO WS-DOM-VALUE-LENGTH

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PARENT,
                      WS-DOM-NMS-ALIAS,
                      WS-DOM-NAMESPACE,
                      WS-DOM-VALUE-LENGTH
```

This second example below is functionally equivalent to the example above in that a namespace node is being added to the element node named "ResponseData". In this example, the element node is referenced by its node id, not by name. Because the SOLA DOM API detects that the parent node identifier is being specified it will ignore the value in the variable WS-DOM-PARENT. For reasons of clarity, the example below shows that WS-DOM-PLACE-HOLDER is used in place of WS-DOM-PARENT.

```
SET  WS-DOM-SET-NAMESPACE  TO TRUE
MOVE 'sola'                TO WS-DOM-NMS-ALIAS
MOVE SAVED-PARENT-ID       TO WS-DOM-PARENT-NODE-ID
MOVE 'http://www.sola.com' TO WS-DOM-NAMESPACE
MOVE +19                   TO WS-DOM-VALUE-LENGTH

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-NMS-ALIAS,
                      WS-DOM-NAMESPACE,
                      WS-DOM-VALUE-LENGTH,
                      WS-DOM-PARENT-NODE-ID,
                      WS-DOM-PLACE-HOLDER,
                     WS-DOM-NMS-NODE-ID
```

This third example (below) ataches an existing namespace (whose node identifier is saved in the WS-SAVED-NMS-NODE-ID field) to an element node that is referenced by its node id, not by name. Because the SOLA DOM API detects that the parent node identifier is being specified it will ignore the value in the variable WS-DOM-PARENT. Similarly, the SOLA DOM API detects that WS-DOM-NMS-NODE-ID is specified and is not null, it will ignore values in the WS-DOM-NMS-ALIAS and WS-DOM-NAMESPACE fields. For reasons of clarity, the example below shows that WS-DOM-PLACE-HOLDER is used in place of WS-DOM-PARENT, WS-DOM-NMS-ALIAS and WS-DOM-NAMESPACE and WS-DOM-VALUE-LENGTH.

```
SET  WS-DOM-SET-NAMESPACE  TO TRUE
MOVE WS-SAVED-NMS-NODE-ID  TO WS-DOM-NMS-NODE-ID
MOVE SAVED-PARENT-ID       TO WS-DOM-PARENT-NODE-ID

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-PARENT-NODE-ID,
                      WS-DOM-PLACE-HOLDER,
                       WS-DOM-NMS-NODE-ID
```

updateAttribute

CICS BATCH

The updateAttribute function is used to update the text of an existing attribute. Changing an attribute name is not allowed (to change the name of an attribute you must first remove and then set the attribute).

The attribute to be updated can be identified by its tag name or attribute node identifier, as well as by its parent tag name or parent node identifier. The node identifier always takes precedence over tag name (for both attribute and parent). WS-DOM-PLACE-HOLDER is used to indicate absence of any search criteria.

The following combinations of criteria are possible. If both node identifier and tag name are provided for the attribute or parent search criteria, the node identifier will always override the tag name.

- By specifying attribute node identifier only.
- By specifying attribute node identifier along with parent node identifier (for verification).
- By specifying attribute node identifier along with parent (tag) name (for verification).
- By specifying attribute tag name only.
- By specifying attribute tag name along with parent node identifier.
- By specifying attribute tag name along with parent (tag) name.

If you have specified the parent name using WS-DOM-PARENT field with the name of the parent element along with attribute tag name and multiple instances of name combination exist within the XML document, then the text of the existing attribute associated with the *first* occurrence of the parent-attribute name combination will be updated. The selected parent's unique node identifier will be returned in the WS-DOM-PARENT-NODE-ID field.

If the attribute node identifier given belongs to a different parent node than the one specified, a return code of +4 will be returned and the attribute will not be updated.

The updateAttribute function can be used in CICS and batch.

```
CALL WS-DOM-API USING
      WS-DOM-RC,         /* Halfword. Return Code */
      WS-DOM-MSG,        /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,     /* Pointer.  For internal use */
      WS-DOM-FUNCTION,   /* Char(25).  The name of the API */
      WS-DOM-PARENT,     /* The tag name of the parent node */
      WS-DOM-TAG-NAME,   /* The tag name whose value to be updated  */
```

```
WS-DOM-VALUE,     /* The text value of the node to be updated */
WS-DOM-VALUE-LENGTH, /* Length of the text value */
WS-DOM-PARENT-NODE-ID, /* Unique id of the parent node */
WS-DOM-ATTRIBUTE-ID /* The ID of the attribute node */
```

Variables:

WS-DOM-RC **Output** **Mandatory**

Returns status.

WS-DOM-MSG **Output** **Mandatory**

When the return code is not zero, this field will contain messages.

WS-DOM-HANDLE **Input** **Mandatory**

For internal use. The DOM handle field uniquely identifies the XML DOM tree in which the attribute will be updated.

WS-DOM-FUNCTION **Input** **Mandatory**

Must be set to **WS-DOM-UPDATE-ATTRIBUTE**.

WS-DOM-PARENT **Input** **Mandatory**

The SOLA DOM API will use this variable to search for a parent element (parent of attribute node) with a matching name. If there are duplicates, then the *first* matching name combination will be selected. The selected parent's unique node identifier will be returned in the field WS-DOM-PARENT-NODE-ID after a successful call.

WS-DOM-TAG-NAME **Input** **Mandatory**

Populate this field with name of the attribute. The name of the attribute must end with a space for delimiting purposes. Absence of this search criteria can be specified by using spaces or null values.

WS-DOM-VALUE **Input** **Mandatory**

Populate this variable with an updated text string and the SOLA DOM API will use this value to replace the existing text of the attribute node.

WS-DOM-VALUE-LENGTH **Input** **Mandatory**

Specify the length of the text string to be updated

WS-DOM-PARENT-NODE-ID **Input/Output** **Optional**

This field is optional. This field contains the unique node identifier of the parent node. When specified (any value other than null) in the CALL statement, it takes precedence over WS-DOM-PARENT field. On input (any value other than null is considered input), the SOLA DOM API uses this field to update text value of an attribute node for the specified parent node. On output, this field contains the unique parent-node-id of the parent element for which the attribute node was updated.

WS-DOM-ATTRIBUTE-ID **Input** **Optional**

This field, when specified in the CALL, will be used as input field for the SOLA DOM API to retrieve the attribute by its unique node identifier. The text will be updated for this attribute node.
If WS-DOM-PARENT or WS-DOM-PARENT-NODE-ID is specified and does not have a null value, the SOLA DOM API will verify the parent. If the specified attribute node identifier has a different parent, a return code of +4 will be returned and the text value will not be updated. WS-DOM-ATTRIBUTE-ID takes precedence over WS-DOM-TAG-NAME during the search of the attribute. This variable is a redefinition of WS-DOM-NODE-ID.

The following return codes are possible:

RC Value	Meaning
0	Successful - No messages issued.
+4	The parent for given node identifier is different than specified, the text value is not updated.
+8	Failed The following are the possible values for WS-DOM-MSG field: • Parent not found • Specified parent node doesn't exist • Attribute not found • Specified attribute node doesn't exist • Input error Possible input errors could be: • The function specified is not supported. • Attribute tag name not specified.
+12	SOLA DOM API abended. Abend code - XXXX (four character code returned at run time).
+16	WS-DOM-HANDLE doesn't contain a valid handle.

Example

In the example below an attribute value is being updated. The attribute is identified by a node identifier that was saved in SAVED-NODE-ID. The new value of the attribute is "ECD" which has a length of 3 bytes.

```
SET  WS-DOM-UPDATE-ATTRIBUTE TO TRUE
MOVE SAVED-NODE-ID        TO WS-DOM-ATTRIBUTE-ID
MOVE 'ECD'                TO WS-DOM-VALUE
MOVE +3                   TO WS-DOM-VALUE-LENGTH

CALL WS-DOM-API USING WS-DOM-RC,
```

```
                          WS-DOM-MSG,
                          WS-DOM-HANDLE,
                          WS-DOM-FUNCTION,
                          WS-DOM-PLACE-HOLDER,
                          WS-DOM-PLACE-HOLDER,
                          WS-DOM-VALUE,
                          WS-DOM-VALUE-LENGTH,
                          WS-DOM-PLACE-HOLDER,
                          WS-DOM-ATTRIBUTE-ID
```

In this next example the same attribute value is being updated. This time the attribute is identified by specifying the attribute tag value (in this case 'Type') and the value of the parent tag (in this case "Program'). If there is more than one attribute that matches this criteria then the *first* attribute will be updated. The new value of the attribute is "ECD" which has a length of 3 bytes.

```
SET  WS-DOM-UPDATE-ATTRIBUTE TO TRUE
MOVE 'Type'                  TO WS-DOM-TAG-NAME
MOVE 'Program'               TO WS-DOM-PARENT
MOVE 'ECD'                   TO WS-DOM-VALUE
MOVE +3                      TO WS-DOM-VALUE-LENGTH

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PARENT,
                      WS-DOM-TAG-NAME,
                      WS-DOM-VALUE,
                      WS-DOM-VALUE-LENGTH
```

updateTextNode

CICS BATCH

The updateTextNode function is used to change the value of an existing text Node under an element node. Please note that updateTextNode is only applicable to a text node for an element node. In order to update text of an attribute use the updateAttribute function.

The updateTextNode function can be used on an element node that does not containt a text node. In such instances, it will add a text node to the element.

A text node can be removed with the updateTextNode function by setting WS-DOM-VALUE-LENGTH to zero.

The text node to be updated can be identified by its parent tag name or parent node identifier. The node identifier always takes precedence over tag name. If multiple instances of the parent tag name exist within the XML document, then the *first* occurrence of the parent tag name will be selected.

In the XML fragment shown below, "Transaction" is an element. "Type" is an attribute of "Transaction" with a value of "ECD". The "Transaction" element has a value (known as a text node) of "TR01".

```
<Transaction Type="ECD">TR01</Transaction>
```

The updateTextNode function can be used in CICS and batch.

```
CALL WS-DOM-API USING
      WS-DOM-RC,         /* Halfword. Return Code */
      WS-DOM-MSG,        /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,     /* Pointer.  For internal use */
      WS-DOM-FUNCTION,   /* Char(25).  The name of the API */
      WS-DOM-PARENT,     /* The tag name of the parent node */
      WS-DOM-PLACE-HOLDER, /* Placeholder */
      WS-DOM-VALUE,      /* The text value of the node to be added */
      WS-DOM-VALUE-LENGTH, /* Length of the text value */
      WS-DOM-PARENT-NODE-ID /* Unique id of the parent node */
```

Variables:

WS-DOM-RC **Output** **Mandatory**

Returns status.

WS-DOM-MSG **Output** **Mandatory**

When the return code is not zero, this field will contain messages.

WS-DOM-HANDLE **Input** **Mandatory**

For internal use. The DOM handle field uniquely identifies the XML DOM tree in which the text node will be updated.

WS-DOM-FUNCTION **Input** **Mandatory**

Must be set to **WS-DOM-UPDATE-TEXT-NODE**.

WS-DOM-PARENT **Input** **Mandatory**

The SOLA DOM API will use this variable to search for a parent element with a matching name. If there are duplicates, the *first* parent with a matching name will be selected.

WS-DOM-PLACE-HOLDER **Place holder** **Mandatory**

WS-DOM-VALUE **Input** **Mandatory**

Populate this variable with an updated text string and the SOLA DOM API will use this value to update an existing text node.

WS-DOM-VALUE-LENGTH **Input** **Mandatory**

Specify the length of the updated text string. To remove the text node set the length to zero.

WS-DOM-PARENT-NODE-ID **Input/Output** **Optional**

This field is optional. This field contains the unique node identifier of the parent node. When specified (any value other than null) in the CALL statement, it takes precedence over WS-DOM-PARENT field. On input (any value other than null is considered input), the SOLA DOM API uses this field to update a text node for the specified parent node. On output, this field contains the unique parent-node-id of the parent element for which the text node was updated.

The following return codes are possible:

RC Value	Meaning
0	Successful - No messages issued.
+8	Failed The following are the possible values for WS-DOM-MSG field: • Parent not found • Specified parent node doesn't exist • Input error Possible input errors could be: • The function specified is not supported.
+12	SOLA DOM API abended. Abend code - XXXX (four character code returned at run time).
+16	WS-DOM-HANDLE doesn't contain a valid handle.

Example

In the example below a text node value is being updated. The text node is identified by the unique node identifier of the parent node (in this case "Transaction"), which was stored in SAVED-PARENT-NODE. The new value of the text node is "TR01" which has a length of 4 bytes.

```
SET  WS-DOM-UPDATE-TEXT-NODE TO TRUE
MOVE SAVED-PARENT-ID         TO WS-DOM-PARENT-NODE-ID
MOVE 'TR01'                  TO WS-DOM-VALUE
MOVE +4                      TO WS-DOM-VALUE-LENGTH

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-VALUE,
                      WS-DOM-VALUE-LENGTH,
                      WS-DOM-PARENT-NODE-ID
```

In this next example the same text node value is being updated. This time the text node is identified by specifying the tag name of the parent element (in this case "Transaction"). If there is more than one element that matches these criteria then the text node of the *first* element will be updated. The new value of the text node is "TR01" which has a length of 4 bytes.

```
SET  WS-DOM-UPDATE-TEXT-NODE TO TRUE
MOVE 'Transaction'           TO WS-DOM-PARENT
MOVE 'TR01'                  TO WS-DOM-VALUE
MOVE +4                      TO WS-DOM-VALUE-LENGTH

CALL WS-DOM-API USING WS-DOM-RC,
                      WS-DOM-MSG,
                      WS-DOM-HANDLE,
                      WS-DOM-FUNCTION,
                      WS-DOM-PARENT,
                      WS-DOM-PLACE-HOLDER,
                      WS-DOM-VALUE,
                      WS-DOM-VALUE-LENGTH
```

freeStorage

CICS BATCH

The freeStorage function frees any storage that has been acquired by the SOLA DOM API. WS-DOM-HANDLE must contain a valid handle for this function to execute correctly.

The freeStorage function can be used in CICS and batch.

```
CALL WS-DOM-API USING
      WS-DOM-RC,        /* Halfword. Return Code */
      WS-DOM-MSG,       /* Char(80). Output msg if return code <> 0 */
      WS-DOM-HANDLE,    /* Pointer.  For internal use */
      WS-DOM-FUNCTION   /* Char(25).  The name of the API */
```

Variables:

WS-DOM-RC **Output** **Mandatory**

Returns status.

WS-DOM-MSG **Output** **Mandatory**

When the return code is not zero, this field will contain messages.

WS-DOM-HANDLE **Input** **Mandatory**

For internal use. The DOM handle field uniquely identifies the XML DOM tree to be freed.

WS-DOM-FUNCTION **Input** **Mandatory**

Must be set to **WS-DOM-FREE-STORAGE**.

The following return codes are possible:

RC Value	Meaning
0	Successful - No messages issued.
+8	Unable to free storage.
+12	SOLA DOM API abended. Abend code - XXXX (four character code returned at run time).
+16	WS-DOM-HANDLE doesn't contain a valid handle.

Example

In this example we are freeing the storage for the DOM tree that is referenced by WS-DOM-HANDLE.

```
SET  WS-DOM-FREE-STORAGE TO TRUE

CALL WS-DOM-API USING WS-DOM-RC,
       WS-DOM-MSG,
       WS-DOM-HANDLE,
       WS-DOM-FUNCTION
```

Data Conversion Utility XMLPC107

INTRODUCTION

SOLA provides a utility program, XMLPC107, to provide data conversion between native mainframe formats and XML. The utility is an assembler program that is implemented with a callable interface.

Don't use the FromString (FS) function of the utility when the source and target are both strings. This is because the utility will scan the string for invalid XML characters and convert them to XML entities. The SOLA DOM API uses the utility internally for XML encoding, and the effect of entitity encoding a string twice is unpredictable. For example, XMLPC107 will convert & to &amp, < to &lt, > to &gt, " to &quot and ' to &apos, and the result of entity encoding those values twice is likely to be ususable.

USING XMLPC107

XMLPC107 Copybook

The utility uses a copybook, XMLCONVT, which is shipped with the product and can be found in the SAMPLIB directory. The copybook is reproduced below:

```
01  CONV-FIELDS.
    05  CONV-METHOD             PIC  X(02).
    05  CONV-COLNAME            PIC  X(30).
    05  CONV-RC                 PIC S9(04) BINARY.
    05  CONV-VALUE              PIC  X(300).
    05  CONV-LEN                PIC S9(08) BINARY.
    05  CONVERTED-LEN           PIC S9(08) BINARY.
    05  CONV-PREC               PIC S9(08) BINARY.
    05  CONV-SCALE              PIC S9(08) BINARY.
    05  CONV-TYPE               PIC S9(08) BINARY.
        88  SQL-C-DEFAULT       VALUE 99.
        88  SQL-CHAR            VALUE 1.
        88  SQL-NUMERIC         VALUE 2.
        88  SQL-DECIMAL         VALUE 3 13 14.
        88  SQL-DECIMAL-SIGN    VALUE 13.
        88  SQL-DECIMAL-NOSIGN  VALUE 14.
        88  SQL-INTEGER         VALUE 4.
```

```
          88  SQL-SMALLINT          VALUE 5.
          88  SQL-FLOAT             VALUE 6.
          88  SQL-REAL              VALUE 7.
          88  SQL-DOUBLE            VALUE 8.
          88  SQL-DATE              VALUE 9.
          88  SQL-TIME              VALUE 10.
          88  SQL-TIMESTAMP         VALUE 11.
          88  SQL-VARCHAR           VALUE 12.
          88  SQL-NUMERIC-EDIT      VALUE 20.
          88  SQL-LONGVARCHAR       VALUE -1.
          88  SQL-BINARY            VALUE -2.
          88  SQL-VARBINARY         VALUE -3.
          88  SQL-LONGVARBINARY     VALUE -4.
      05  CONV-WORK-AREA            PIC X(250).
```

Calling XMLPC107

The XMLPC107 utility is called as follows:

```
CALL 'XMLPC107' USING
      CONV-RC         /* Fullword. Return Code */
      CONV-METHOD     /* Char(2). TS or FS */
      CONV-VALUE      /* User Specified. Input Value */
      CONV-LEN        /* Fullword. Length(Conv-Value) */
      CONV-TYPE       /* Fullword. Type of source (TS) or target (FS)*/
      CONV-PREC       /* Fullword. Precision */
      CONV-SCALE      /* Fullword. Scale */
      WS-CONVERTED-LEN    /* Fullword. User Specified. */
      WS-CONVERTED-VALUE  /* User Specified. */
      CONV-WORK-AREA      /* Char(256). Mandatory */
```

The allowable values for each of the fields is as follows:

CONV-RC	Halfword. The return code from the conversion utility. Return codes are: 0 OK <> 0 Error
CONV-METHOD	Supports 2 methods: TS – To String (native mainframe to XML) FS – From String (XML to native mainframe)
CONV-VALUE	The value to be converted
CONV-LEN	Fullword. Length of the source to be converted (character conversions only).

CONV-TYPE	Fullword. Data type of the source (TS) or target (FS). Data types are defined as condition names in the XMLCONVT copybook. The values are as follows: CHAR VALUE 1 CHARACTER STRING NUMERIC VALUE 2. DISPLAY NUMERIC DECIMAL VALUE 3 13 14. PACKED CATCH ALL DECIMAL-SIGN VALUE 13. SIGNED PACKED DECIMAL-NOSIGN VALUE 14. UNSIGNED PACKED INTEGER VALUE 4. FULLOWRD SMALLINT VALUE 5. HALFWORD FLOAT VALUE 6. FOR FUTURE USE REAL VALUE 7. FOR FUTURE USE DOUBLE VALUE 8. FOR FUTURE USE DATE VALUE 9. DB2 FORMAT. TIME VALUE 10. DB2 FORMAT. TIMESTAMP VALUE 11. DB2 FORMAT. VARCHAR VALUE 12. DB2 FORMAT. NUMERIC-EDIT VALUE 20. EDITED NUMERIC. C-DEFAULT VALUE 99. NOT SUPPORTED LONGVARCHAR VALUE -1. DB2 FORMAT. BINARY VALUE -2. BASE64. VARBINARY VALUE -3. BASE64. TS ONLY. LONGVARBINARY VALUE -4. BASE64. TS ONLY.
CONV-PREC	Fullword. For numeric TS conversions CONV-PREC and CONV-SCALE define the precision and scale of the source. For numeric FS conversions CONV-PREC and CONV-SCALE define the precision and scale of the target.
CONV-SCALE	See Fullword. CONV-PREC above.
WS-CONVERTED-LEN	Fullword. The actual length of the converted value returned by the utility.
WS-CONVERTED-VALUE	The converted data returned by the utility.
CONV-WORK-AREA	A 250 byte area that the utility uses for storing intermediate results. This area allows the utility to not depend on CICS Storage control for reentrancy.

XMLPC107 Usage Examples

The following code fragments are provided to illustrate the use of the XMLPC107 utility.

1) Convert from packed-decimal to string:

In this example, CURRENT-BALANCE contains the value to be converted. The converted value will be placed in WS-DOM-VALUE and the converted value length will be placed in WS-DOM-VALUE-LENGTH.

```
05  CURRENT-BALANCE             PIC S9(13)V9(2) PACKED-DECIMAL.
05  WS-DOM-VALUE-LENGTH         PIC S9(08) BINARY.
05  WS-DOM-VALUE                PIC X(256).

  MOVE 'TS'                 TO CONV-METHOD
  SET SQL-DECIMAL-SIGN      TO TRUE
  MOVE 15                   TO CONV-PREC
  MOVE 2                    TO CONV-SCALE
  CALL 'XMLPC107' USING CONV-RC,
                        CONV-METHOD,
                        CURRENT-BALANCE,
                        CONV-LEN,
                        CONV-TYPE,
                        CONV-PREC,
                        CONV-SCALE,
                        WS-DOM-VALUE-LENGTH,
                        WS-DOM-VALUE,
                        CONV-WORK-AREA
```

2) Convert from string to packed-decimal:

In this example, WS-DOM-VALUE contains the string to be converted and the length of the string is in WS-DOM-VALUE-LENGTH. The converted value will be placed in CURRENT-BALANCE.

```
  MOVE 'FS'                 TO CONV-METHOD
  SET SQL-DECIMAL-SIGN      TO TRUE
  MOVE 15                   TO CONV-PREC
  MOVE 2                    TO CONV-SCALE
  MOVE WS-DOM-VALUE-LENGTH TO CONV-LEN
  CALL 'XMLPC107' USING CONV-RC,
                        CONV-METHOD,
                        WS-DOM-VALUE,
                        WS-DOM-VALUE-LENGTH,
                        CONV-TYPE,
                        CONV-PREC,
                        CONV-SCALE,
                        CONV-LEN,
                        CURRENT-BALANCE,
                        CONV-WORK-AREA
```