Enhanced Tools for Reading XML Attributes

By David Adams

Technical Note 06-43

Overview

XML element's may have any number of attributes, such as the `id` attribute in the `contact` element shown below:

```
<contact id="1">
```

The DOM (Document Object Model) section of the 4th Dimension language contains three commands for counting and reading XML attributes, listed below:

DOM Count XML attributes
DOM GET XML ATTRIBUTE BY INDEX
DOM GET XML ATTRIBUTE BY NAME

This technical note and the accompanying sample database add a number of enhancements and refinements to these basic commands, including:

- Automatic error handling to avoid problems from invalid node references.
- Automatic detection of the `#document` node, which can cause some versions of 4th Dimension to quit unexpectedly when read for attributes.
- Utilities that match elements by name, value, or name and value.
- A tool to copy a node's attribute names and their values to a pair of arrays.

This technical note provides background information on working with XML nodes and their attributes in 4th Dimension and then documents the utility routines included in the sample database, listed below:

DOM_AttributesToArrays
DOM_CountAttributes
DOM_GetAttributeNameWithValue
DOM_GetAttributeValueWithName
DOM_HasAttributeNameAndValue
DOM_HasAttributeNamed
DOM_HasAttributeWithValue
DOM_ReferenceIsValid

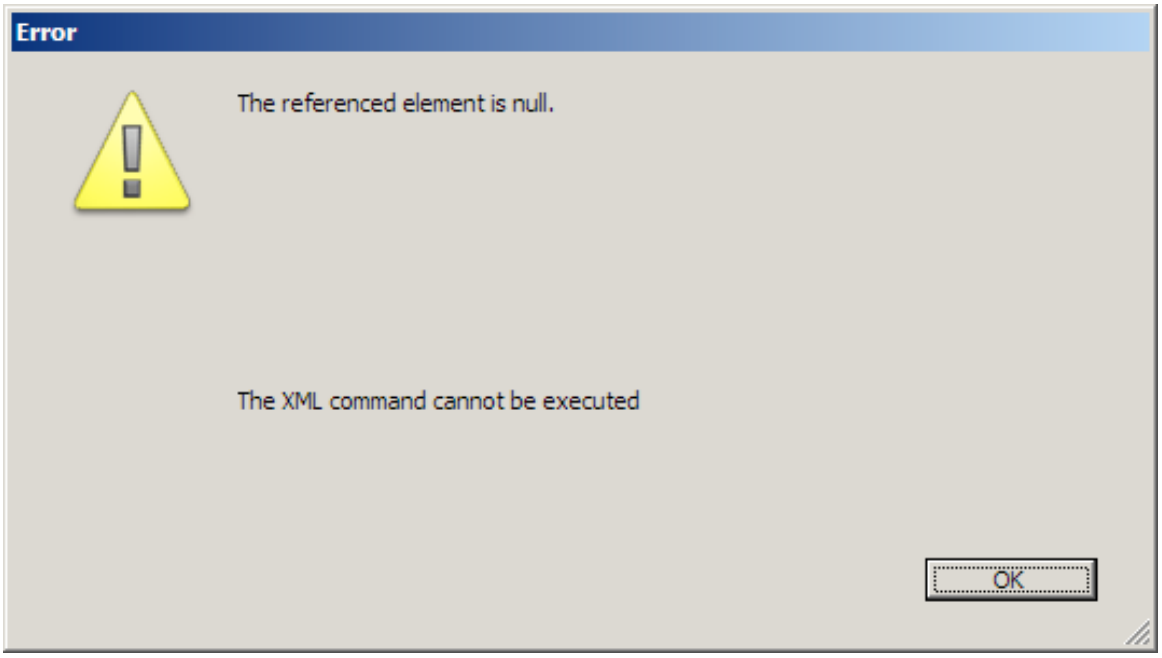
String_EqualCaseSensitively

Note *Some of the DOM routines described in this note are also documented in Technical Note xx-xx,* ***Enhancing the DOM XML Reading Functions****.*

Background Information on Nodes and Attributes

Tree Navigation and Bad Nodes

4th Dimension's DOM commands render a source XML document or variable as a tree of linked nodes. The DOM command set includes tools for navigating through the tree and reading information from nodes. The tree navigation commands, such as **DOM Get Parent XML element** and **DOM Get first child XML element**, use the OK system variable to indicate when the command has left the tree and reached a non-existent node. This feature makes it possible to navigate through the tree without knowing in advance exactly how many ancestors, siblings, or descendants a particular node has. However, it also leads to the navigation commands returning references to invalid nodes. This causes problems when reading information, such as an element's name, value, or attributes. For example, calling the built-in **DOM GET XML ELEMENT NAME** command on an invalid node reference leads to a dialog like the one pictured below:



To avoid problems from invalid node references, the database included with this technical note automatically wraps the standard node reading functions in an error handler that suppresses error displays. Additionally, the *DOM_ReferenceIsValid* function offers a simple tool for checking node validity.

Note *For more details on testing DOM node references for validity and managing bad nodes, see Technical Note xx-xx,* ***Avoiding Problems Reading DOM XML Nodes****.*

XML Attributes and the #document Node

The number of attributes associated with a node can be determined with a call to **DOM Count XML attributes**. As with the other DOM reading commands, 4th Dimension should be expected to throw an error if the node reference is invalid. However, in the case of **DOM Count XML attributes**, there is an additional special case that needs to be detected. Using **DOM Get parent XML element** it is possible to navigate above the root of the tree to an artificial node that holds the XML document information, such as XML version and encoding type. The **DOM GET XML ELEMENT NAME** command treats this artificial node as a valid reference and returns the node name `#document`. Unfortunately, in some versions of 4th Dimension, calling **DOM Count XML attributes** on the `#document` element causes 4th Dimension to quit unexpectedly. Fortunately, this problem is easy to fix by testing the node name before calling **DOM Count XML attributes**. This functionality is implemented in the *DOM_CountAttributes* function included in the sample database.

Method Documentation

DOM_AttributesToArrays

DOM_AttributesToArrays (Alpha [16];Pointer;Pointer)
DOM_AttributesToArrays (XML reference;->Names array;->Values array)

This routine copies the attributes of an XML node into a pair of text arrays.

Note: The routine doesn't accept string arrays in the place of text arrays.

DOM_CountAttributes

DOM_CountAttributes (Alpha [16]) : Longint
DOM_CountAttributes (XML reference) : Count of attributes

This routine counts the number of attributes associated with an XML element. It enhances the behavior of **DOM Count XML attributes** with automatic error handling and avoiding reading the attributes of illegal nodes.

DOM_GetAttributeNameWithValue

DOM_GetAttributeNameWithValue (Alpha [16];Text;{Boolean}) : Text
DOM_GetAttributeNameWithValue (XML reference;Value to match;{Compare values case-sensitively?}) : Matching name or empty string

This routine looks for an attribute based on value and, if found, returns its name.

Values are compared case-insensitively, by default.

DOM_GetAttributeValueWithName

DOM_GetAttributeValueWithName (Alpha [16];Text) : Text
DOM_GetAttributeValueWithName (XML reference;Name to match) : Matching value or empty string

Looks for an attribute based on name and, if found, returns its value.

DOM_HasAttributeNamed

DOM_HasAttributeNamed (Alpha [16];Text) : Boolean
DOM_HasAttributeNamed (XML reference;Attribute name): Attribute found?

Tests if an XML node includes a node with the specified name, regardless of value.

Names are always compared case-sensitively.

DOM_HasAttributeNameAndValue

DOM_HasAttributeNameAndValue (Alpha [16];Text;Text;{Boolean}) : Boolean
DOM_HasAttributeNameAndValue (XML reference;Attribute name;Attribute value;{Compare values case-sensitively?}): Attribute found?

Tests if an XML node includes a node with the specified name and value.

Names are always compared case-sensitively, values are compared case-insensitively, by default.

DOM_HasAttributeWithValue

DOM_HasAttributeWithValue (Alpha [16];Text;{Boolean}) : Boolean
DOM_HasAttributeWithValue (XML reference;Attribute value;{Compare values case-sensitively?}): Attribute found?

Tests if an XML node includes a node with the specified value, regardless of name.

Values are compared case-insensitively, by default.

DOM_ReferenceIsValid

DOM_ReferenceIsValid (Alpha [16]) : Boolean
DOM_ReferenceIsValid (XML reference) : Element reference is valid?

This routine tests if a node reference is valid.

DOM_ReferenceIsValidOnError

This custom error handler is used internally as a custom error handler by the *DOM_ReferenceIsValid* routine.

DOM_StartCustomErrorHandling

This routine is used internally to record the current error handler name, and the current value of the Error system variable, as well as to install a custom error handler.

DOM_StopCustomErrorHandling

This routine is used internally to reverse the operations of the *DOM_StartCustomErrorHandling* routine.

String_EqualCaseSensitively

String_EqualCaseSensitively (Text;Text): Boolean
String_EqualCaseSensitively (Base text;Comparison text) : Equal?

This routine tests if two strings/texts are equal case-sensitively. This functionality is required to compare XML element names, which are always case-sensitive.

Summary

4th Dimension's native DOM commands include tools for counting and reading the attributes of an XML node. The sample database implements an expanded and enhanced suite of attribute-related tools for reuse in any database. The enhanced routines automatically handle errors, avoid dangerous calls, and simplify reading and testing attributes and their values.