Getting an XML Element's Full Path

By David Adams
Technical Note 07-03

Abstract

This technical note explains how to build the full path to an XML node using the built-in 4th Dimension DOM commands and includes a sample database with the code implemented.

Overview

XML has become nearly universal as a scheme for formatting data for exchange between systems and as a storage format for program settings and other data. One of the challenges of dealing with XML is its best feature: the ability to store and organize information hierarchically. As an example, consider the short XML sample below:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<contacts>
   <contact>
      <name>Dan West</name>
      <business>
         <phone>
            <area_code>04</area_code>
            <number>0035-9110</number>
         </phone>
      </business>
      <home>
         <phone>
            <area_code>02</area_code>
            <number>6493-3250</number>
         </phone>
      </home>
   </contact>
</contacts>
```

In the example above, the following elements appears twice:

```
      <phone>
         <area_code>
         <number>
```

These elements appear as children of the `<business>` and the `<home>` elements. When processing XML documents, repeated element names are commonplace. In such cases, knowing the full path to the node helps distinguish between identically named elements. For example, the paths to the two instances of the `<area_code>` element are listed below:

```
/contacts/contact/business/phone/area_code/
/contacts/contact/home/phone/area_code/
```

This technical note explains how to build the full path to a node and includes a sample database with the code implemented.

Review: Native DOM Commands

4th Dimension already includes native path-oriented DOM (Document Object Model) commands. For example, selecting the two <area_code> elements mentioned above is straightforward, as shown in the fragment below (whitespace adjusted for legibility):

```
C_STRING($16;$businessPhone_xmlref)
C_STRING($16;$homePhone_xmlref)
$businessPhone_xmlref:=DOM Find XML element($root_xmlref;
    "/contacts/contact/business/phone/area_code/")
$homePhone_xmlref:=DOM Find XML element($root_xmlref;
    "/contacts/contact/home/phone/area_code/")
```

The **DOM Find XML element** function is simple and effective but only works in situation where the full path is known beforehand. The command addresses many XML processing needs, but not all. For example:

- XML trees are often examined generically using a top-down tree walk, a technique used in several existing technical notes, sample databases, and within several of the examples in *The 4D Web Services Companion*. Within a generic tree walk, the full path is not known from the outset. Instead, the code works with each node in sequence. The **DOM Find XML element** function works from the point of view of the top of the tree down while generic tree walks work from the point of view of the node up.
- The **DOM Find XML element** function path syntax is not expressive enough to match all patterns. As an example, it offers no way to locate nodes that end with the path `/phone/area_code/`, regardless of antecedents. It is often convenient or necessary to deal with nodes and tree fragments consistently, regardless of the exact document root or full path.

Fortunately it is easy to add a function that generates a full path for the current node using other native commands.

Note *XML paths, element names, and attribute names are all case-sensitive. See 4D Technical Note 05-41,* Case-Sensitive Operations in 4th Dimension*, for code to handle case-sensitive comparisons.*

The Routines

The code of the *DOM_GetFullPath* routine and its error handler are listed below.

DOM_GetFullPath

The code below calculates the full path to the current node. If an invalid or null XML node reference is passed into the routine, an empty string is returned.

```
C_TEXT($0;$fullPath_text)
C_STRING(16;$1;$noderef)`  <-- The calling routine must pass in this parameter.

$noderef:=$1

$fullPath_text:=""

` Store existing error/error handling state.
If (Undefined(Error))
  Error:=0
End if
C_LONGINT($previousValueOfErrorVariable_l)
$previousValueOfErrorVariable_l:=Error
C_STRING(31;$previousErrorMethodName_s)
$previousErrorMethodName_s:=Method called on error
ON ERR CALL("DOM_ErrorTrappingRoutine")

C_TEXT($name_text)
$name_text:=""
DOM GET XML ELEMENT NAME($noderef;$name_text)`  Default to starting element name.

While (OK=1)
  Case of
    : ($name_text="#document")` The #document node is a synthetic node above the tree.
      OK:=0` Get out of routine.

    : ($name_text="")` Bad reference.
      OK:=0` Get out of routine.

    Else  ` Valid name.
      $fullPath_text:="/"+$name_text+$fullPath_text
      OK:=1` Continue processing.

      $noderef:=DOM Get parent XML element($noderef;$name_text)`  Try to get a parent.
  End case

End while

` Restore previous error/error handling state.
Error:=$previousValueOfErrorVariable_l`Restore original error value.
ON ERR CALL($previousErrorMethodName_s)`Restore  original error handler.

If ($fullPath_text#"")` A path was built.
  $fullPath_text:=$fullPath_text+"/"` Add final slash.
End if

` Return result.
$0:=$fullPath_text
```

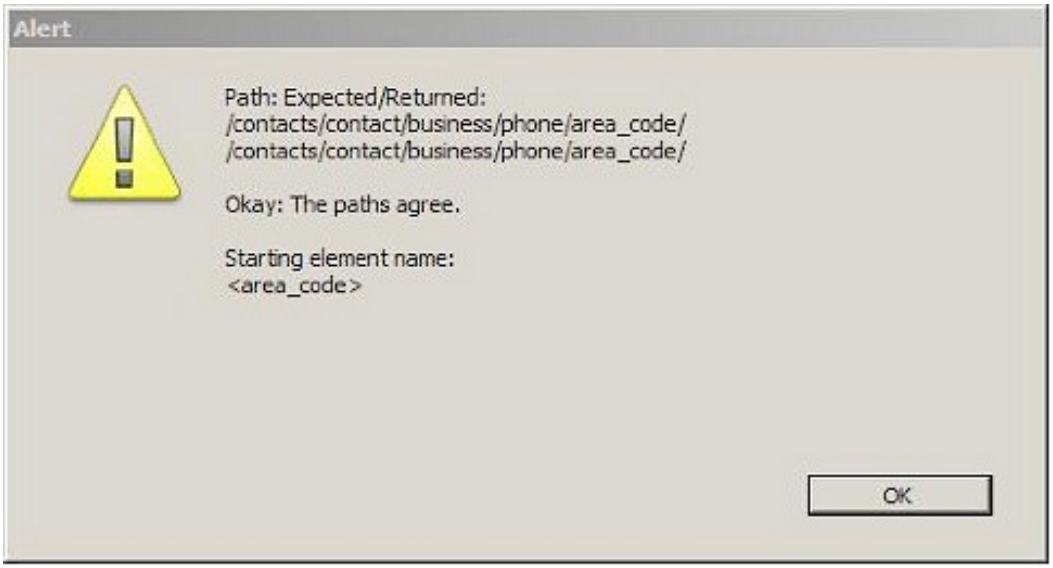
DOM_ErrorTrappingRoutine

The *DOM_ErrorTrappingRoutine* is installed by *DOM_GetFullPath* to trap errors arising from using invalid node references. The error handler includes a single line of code, listed below:

DOM_Error:=Error

The Sample Database

The sample database includes the code listed above and a simple test routine named *Test_GetFullPath*. The test code is designed to exercise the *DOM_GetFullPath* in various conditions, including passing in good nodes, bad nodes, and the synthetic `#document` node. For each test the code displays a simple status alert which reports if the test conditions were met, such as the screen shown below:



Summary

The native DOM XML commands do not include a function to return the full path to a specified XML node. Fortunately, this functionality is easy to write using the **DOM Get parent XML element** command. The sample database includes a method named *DOM_GetFullPath* that provides the full path building feature.