Getting an XML Element's Depth

By David Adams

Technical Note 07-01

Abstract

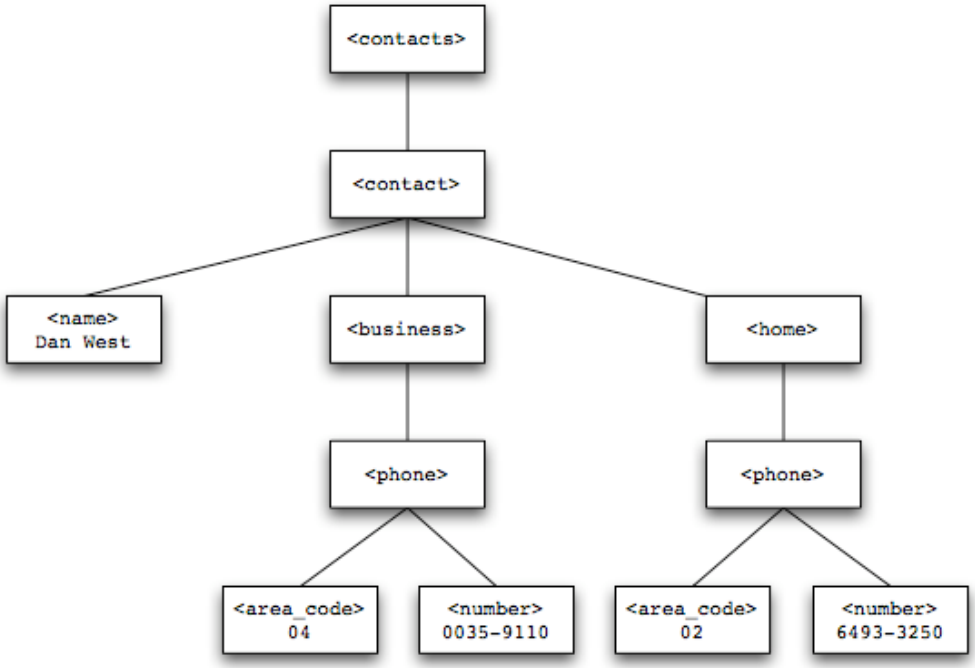
The 4th Dimension DOM commands do not include a function to return a node's level. Fortunately, this feature is easy to add using the **DOM Get parent XML element** command. This technical note describes the necessary code and includes a sample database with the method implemented.

Overview

XML documents read through the DOM (Document Object Model) commands are treated as a tree. 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>
```

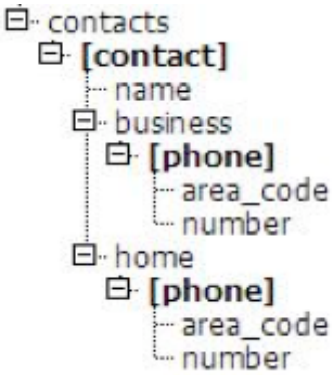
Rendered as a tree, the nodes are linked as diagramed below:



Each node within the tree has a level, such as 1 for the <contacts> element, 2 for the <contact> element, 3 for the <business> element, and so on. While processing XML trees and reading XML nodes, it is often convenient to know a node's level. However, the 4th Dimension DOM commands don't include a function to return a node's level. Fortunately, this feature is easy to add using the **DOM Get parent XML element** command. This technical note describes the necessary code and includes a sample database with the method implemented.

Uses for Retrieving the Current Node's Level

Getting an XML element's level within a DOM tree is useful whenever code needs to take special actions based on depth. As an example, a routine that copies an XML tree into a hierarchical list can format list items based on their depth. In the imaginary example below, nodes on level 2 or 4 are set in bold and have square braces placed around the element names:



Depending on the XML's structure and the processing task, other common reasons to need a node's depth include:

- Adding new records to hold embedded values, such as [Contact] and [Phone] records for the <contact> and <phone> elements in the sample.
- Ignoring elements based on level, such as ignoring the <contacts> element at level 1 in the example above.
- Formatting textual representations of an XML tree, such as adding tabs to reflect each element's level within the tree.

The Routines

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

DOM_GetNodeLevel

```
C_LONGINT($0;$nodeLevel_count)
C_STRING(16;$1;$noderef)
$noderef:=$1

$nodeLevel_count:=0

` 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($currentNodeName_text)
$currentNodeName_text:=""

` Default to starting element name.
DOM GET XML ELEMENT NAME($noderef;$currentNodeName_text)
While (OK=1)
  Case of
    : ($currentNodeName_text="")
      OK:=0` We're done scanning.

    : ($currentNodeName_text="#document")
    ` An artificial node above the tree. We're not treating this as a valid ancestor.
      OK:=0` We're done scanning.

    Else
      $nodeLevel_count:=$nodeLevel_count+1` Save this value as it may be the last parent.
```

```
    ` Try to get another parent.
    $noderef:=DOM Get parent XML element($noderef;$currentNodeName_text)
  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.

$0:=$nodeLevel_count
```

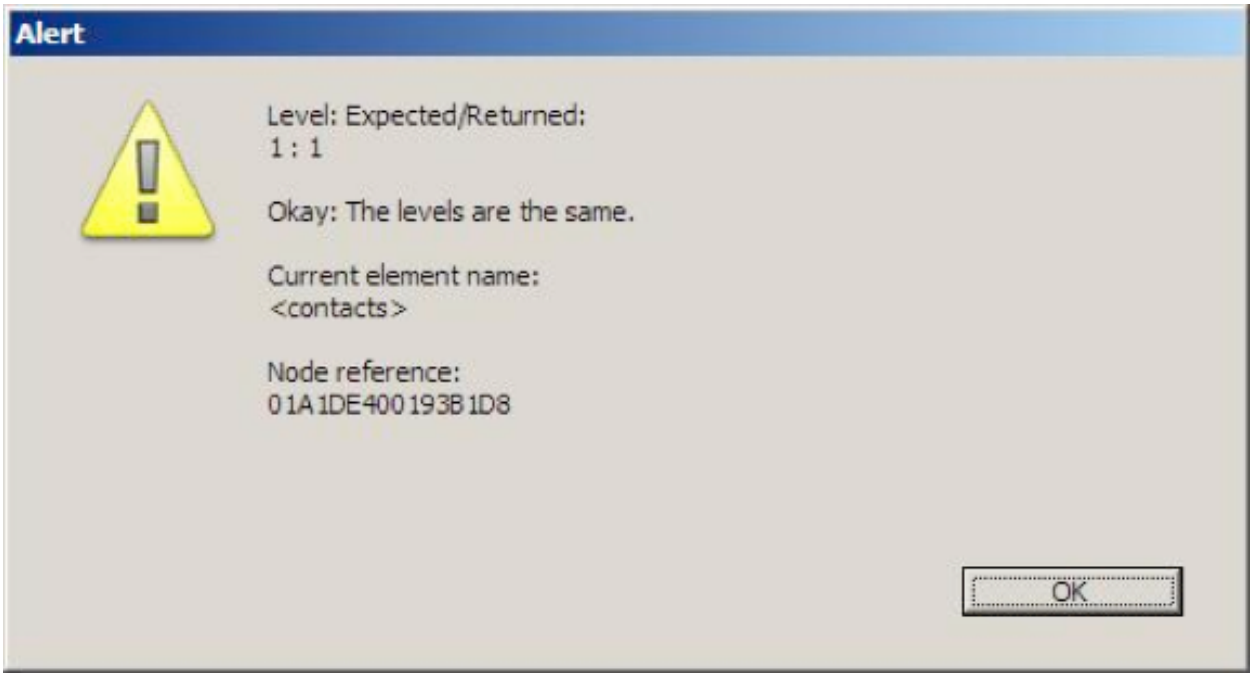
DOM_ErrorTrappingRoutine

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

```
OM_Error:=Error
```

The Sample Database

The sample database includes the code listed above and a simple test routine named **Test_GetNodeLevel**. The test code is designed to exercise the **DOM_GetNodeLevel** method 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 reporting if the test conditions were met, such as the screen shown below:



Summary

When processing DOM trees, it is often convenient to know an XML element's depth within the tree. This technical note describes how to implement this behavior using the native DOM Get parent XML element command and some custom code. The method's code is listed in this technical note and implemented in an accompanying sample database.