# Enhancing the DOM XML Reading Functions

By David Adams

TN 06-40

## Overview

4th Dimension 2004 には、DOM(Document Object Model)による XML 解析コマンドのフルセットが揃っています。DOM コマンドを使用すれば、XML ツリーをスキャンし、ノードごとに要素名、要素値、属性を読み取ることができます。その意味で DOM コマンドには十分の機能が備わっていますが、さらに利便性を高めることができればと思い、このテクニカルノートを執筆しました。このテクニカルノートには、エラーハンドリング、ホワイトスペースクリーニング、および DOM コマンドにはないノードマッチングオプションを追加されたサンプルデータベースが収録されています。このシステムで使用する主なメソッドは次のとおりです：

***DOM_AttributesToArraysDOM_CountAttributes***
***DOM_CountElementByName***
***DOM_ElementExists***
***DOM_FindElementByName***
***DOM_GetElementName***
***DOM_GetElementValue***
***DOM_ReferenceIsValid String_EqualCaseSensitively***
***XML_CleanWhitespace***
***XML_InitWhitespaceCharacters***

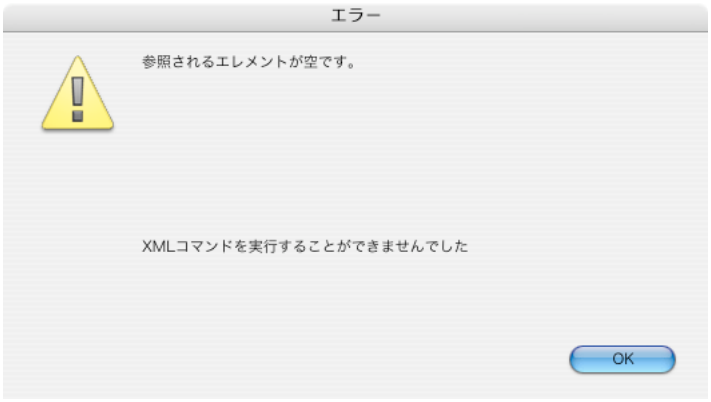
本文は、これらのメソッドにより強化される機能のおおまかな概観、およびメソッドのドキュメントで構成されています。データベース本体のメソッド DOM Read Me にも詳細なドキュメントが含まれており、それぞれの動作を確認するためのデモ画面も提供されています。

## Background Information and Feature Overview

### Tree Navigation and Bad Nodes

4th Dimension の DOM コマンドでは、XML ソースをドキュメントあるいは変数から読み込み、相互に結びついたノードで構成されるツリーとして扱います。DOM コマンドセットには、このツリーの中を移動するためのナビゲーションコマンド DOM Get Parent XML element、DOM Get first child XML element などが含まれており、ツリーから外れてしまった場合の通知には OK システム変数が使用されます。これにより、事前にツリーの構造を知らなくても、ノードからノードへ自由に移動することができるというわけです。

このメカニズムでは、ナビゲーションコマンドによりツリーから外れたノードが返されることがあり、無効なノードの場合、要素名、要素値、属性などを読み取るコマンドを実行するときになって問題が起こります。たとえば無効なノードで DOM GET XML ELEMENT NAME コマンドを使用すると次のようなダイアログが表示されます：




無効なノードに関係した問題を避けるため、テクニカルノートに収録されたデータベースではエラーハンドリングメソッドを使用して標準のコマンドで表示されるアラートを抑えるようにしています。加えてノードの有効性を確認するための簡単なツールが DOM_ReferenceIsValid メソッドという形で提供されています。

注記　DOM ノードリファレンスのチェック、および不正なノードの管理については、テクニカルノート「Avoiding Problems Reading DOM XML Nodes」を参照してください。

### Whitespace Handling

XML 要素の値には、大抵、XML を読みやすくするために挿入されたデータ前後のホワイトスペース、具体的には、インデントのために追加されたタブ記号やキャレッジリターン記号が含まれています。この種の記号には、特別な意味があるときとないときがあります。4th Dimension の DOM コマンドおよび SAX コマンドは、ホワイトスペースには必ず意味があるものとみなし、要素値の一部として返すようになっています。テクニカルノートに収録されたデータベースにはテキストブロック前後のホワイトスペースを取り除くメソッド XML_CleanWhitespace が含まれており、任意のメソッドからコールできるようになっています。内部的な話をすれば、メソッド DOM_GetElementValue は DOM GET XML ELEMENT VALUE コマンドとこのメソッドを組み合わせることにより、標準のコマンドにホワイトスペース除去を追加しています。

注記　XML ホワイトスペースの処理については、テクニカルノート「Cleaning Whitespace from XML Values」を参照してください。

### Attribute Handling

XML 要素には属性を持たせることができます。たとえば、次の要素 contact には属性 id が定義されています：

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

DOM コマンドは、ノードの属性を名前またはインデックスで参照できるようになっていますが、サンプルデータベースでは特定ノードの属性名と属性値を一括してテキスト配列に代入する代替メソッド *DOM_AttributesToArrays* を使用しています。

無効なノードがエラーを返すことは前述しましたが、その問題とは別に DOM Count XML attributes コマンドには、一部バージョンの 4th Dimension で #document 要素に遭遇するとクラッシュするという問題がありました。#document は、XML ツリーのルートよりも上位に配置される特殊なノードです。サンプルデータベースでは、この問題を回避した代替メソッド *DOM_CountAttributes* を使用しています。

注記　XML 属性の処理については、テクニカルノート「Enhanced Tools for Reading XML Attributes」を参照してください。

## Method Documentation

### DOM_AttributesToArrays

*DOM_AttributesToArrays* (文字[16];ポインタ;ポインタ)
*DOM_AttributesToArrays* (XML リファレンス;->Names 配列;->Values 配列)
XML リファレンスで渡されたノードの属性名と属性値を一組のテキスト配列に返します。
注記　テキスト配列の代わりに文字例配列は使用できません。

### DOM_CountAttributes

*DOM_CountAttributes* (文字[16]) → 倍長整数
*DOM_CountAttributes* (XML リファレンス) → 属性の数
XML リファレンスで渡されたノードの属性数を返します。 DOM Count XML attributes にエラーハンドリングを追加し、と不正なノードに読み取りを回避するように改良しました。

### DOM_CountElementByName

*DOM_CountElementByName* (文字[16];テキスト;{倍長整数}) → 倍長整数
*DOM_CountElementByName* (XML リファレンス;要素の名前;{最大値}) → 要素数
XML ツリーの中で渡された名前の要素が出現する回数を返します。任意の引数を使用すれば、検索を終了する最大値を指定することができます。デフォルトの最大値は MAXLONG です。

### DOM_ElementExists

*DOM_ElementExists* (文字[16];テキスト;{倍長整数}) → ブール
*DOM_ElementExists* (XML リファレンス;要素の名前;{インスタンス}) → 要素のインスタンスが存在すれば True
XML ツリーの中で渡された名前の要素が存在するかを調べます。任意の引数を使用すれば、特定のインスタンス(x 個目の出現)が存在するかを調べることができます。デフォルトのインスタンスは 1 です。

### DOM_FindElementByName

*DOM_FindElementByName*(文字[16];テキスト;倍長整数) → 文字[16]
*DOM_FindElementByName* (XML リファレンス;要素の名前;インスタンス) → 要素の XML リファレンスまたは空の文字列
XML ツリーの中で渡された名前の要素を探し、そのノードの XML リファレンスを返します。インスタンスを指定することにより、特定のインスタンス(x 個目の出現)を探すことができます。

### DOM_GetElementName

*DOM_GetElementName*(文字[16]) → テキスト
*DOM_GetElementName* (XML リファレンス) → 要素名
XML リファレンスで渡されたノードの要素名を返します。DOM GET XML ELEMENT NAME と同じですが、エラーハンドリングが自動処理です。

### DOM_GetElementValue

*DOM_GetElementValue*(文字[16];{ブール}) → テキスト
*DOM_GetElementValue*(XML リファレンス;{ホワイトスペース除去？}) → 要素値
XML リファレンスで渡されたノードの要素値を返します。DOM GET XML ELEMENT VALUE と同じですが、エラーハンドリングが自動処理であり、またホワイトスペースを除去することができます。この引数のデフォルトの値は False です。

### DOM_ReferenceIsValid

*DOM_ReferenceIsValid*(文字[16]) → ブール
*DOM_ReferenceIsValid*(XML リファレンス) → 要素の XML リファレンスは有効？
XML リファレンスで渡されたノードの有効性を調べます。

### DOM_ReferenceIsValidOnError

*DOM_ReferenceIsValid* で使用される内部的なエラーハンドリングメソッドです。

### DOM_StartCustomErrorHandling

インストールされているエラーハンドリングメソッドの名前と ERROR システム変数の値を保存した上でカスタムエラーハンドリングメソッドをインストールするメソッドです。

### DOM_StopCustomErrorHandling

*DOM_StartCustomErrorHandling* による処理の逆転するメソッドです。

### String_EqualCaseSensitively

*String_EqualCaseSensitively*(テキスト;テキスト) → ブール
*String_EqualCaseSensitively*(元となるテキスト;比較するテキスト) → テキストは一致？
大文字と小文字を区別して文字列/テキストを比較します。XML 要素名では大文字と小文字を区別するため、このような処理が必要です。

### XML_CleanWhitespace

*XML_CleanWhitespace* (テキスト) → テキスト
*XML_CleanWhitespace* (元のテキスト) → 処理されたテキスト
前後のホワイトスペース文字を除去されたテキストを返します。

### XML_InitWhitespaceCharacters

除去の対象となるホワイトスペース文字の配列を初期化します。

## Summary

ビルトインの DOM コマンドセットには必要な機能が揃っていますが、簡単なコードを加えれば、さらに利便性が向上します。このテクニカルノートに収録されたサンプルデータベースでは、無効なノードを自動的にエラーハンドリングし、ホワイトスペースを除去し、一部の不具合を回避し、ノードに対するアクセスを快適にするメソッドセットが提供されています。