Autodesk®
Scaleform®

ActionScript 2 XML User Guide

本書では、Scaleform 4.4 で使用可能な ActionScript 2 XML サポートの設定について説明しています。

著者: Prasad Silva
バージョン: 3.0
最終更新日: 2010 年 7 月 28 日

Autodesk®
GAMEWARE

Copyright Notice

Autodesk® Scaleform® 4.4

Autodesk Scaleform の連絡先:

ドキュメント	XML User Guide (XML ユーザー ガイド)
住所	Autodesk Scaleform Corporation 6305 Ivy Lane, Suite 310 Greenbelt, MD 20770, USA
ホームページ	www.scaleform.com
電子メール	info@scaleform.com
電話	(301) 446-3200（日本のお客様は、050-3685-7107 へ）
Fax	(301) 446-3199

目次

1 はじめに

Autodesk Scaleform SDK は、軽量で高性能なリッチ メディア Flash®ベクター グラフィック エンジンで、特にゲーム機や PC ゲームの開発者向けにクリーンルーム実装で構築されています。Scaleform は、Adobe® Flash® Studio などの実績のあるビジュアル オーサリング ツールのスケーラビリティと開発の容易さを、最先端のゲーム開発者が必要とする最新のハードウェア グラフィック アクセラレータに組み合わせたものです。

Flash による ActionScript 2 XML のサポートは、AS2 XML API で提供されます。XML および XMLNode クラスは、ロード、解析、DOM ツリー管理などの機能をラップします。こうしたコア XML のサポートは、Scaleform 4.4 でも忠実に再現されます。Scaleform インプリメンテーションには DOM ツリー管理機能が組み込まれており、SAX2 ベースのインターフェイスを公開することにより、カスタムの XML パーサーをプラグインします。デフォルトでは、Scaleform Player アプリケーションは、オープン ソースの Expat パーサーをラップするパーサー インプリメンテーションを使用します。Scaleform をサポートする以下のプラットフォームであれば、どれでも XML をサポートします:Windows®、Linux®、MacOS®、Xbox 360®、PSP® (PlayStation®Portable)、PlayStation®2 (PS2)、PlayStation®3 (PS3™)、Wii®

本書では Scaleform XML アーキテクチャについて説明し、アプリケーションでそれを使用する方法について紹介します。カスタム パーサーの設定方法と既知の制限については、最後に記載しています。

2 XML の使用

以下にサンプル コード スニペットと、ユーザーが Scaleform の XML インプリメンテーションを始める際に役立つ簡単なガイドを記載します。

2.1 ActionScript の XML サポートを有効にする

ActionScript の XML 処理のサポートを有効にするには、Scaleform の初期化段階で XML パーサーのステートを Scaleform ローダー用に設定する必要があります。以下は、デフォルトの FxPlayer.cpp ファイルからの引用です。同様のパターンをカスタム パーサー インプリメンテーションにも使用できます:

```
…
#include "XML/XML_Expat.h"
using namespace Scaleform;
…
Ptr<XML::Parser> pexpatXmlParser = *new XML::ParserExpat;
Ptr<XML::SupportBase> pxmlSupport = *new XML::Support(pexpatXmlParser);
mLoader.SetXMLSupport(pxmlSupport);…
```

2.2 C++言語のドキュメント解析

パーサーのステートを使って、C++から直接 XML ドキュメントを解析することができます。ActionScript XML のサポートが有効であれば、以下のコードを使用して XML ファイルから DOM ツリーを作成できます (Scaleform ソース SDK ライセンスが必要です)。

```
…
#include "GFx/XML/XML_DOM.h"
using namespace Scaleform;
…
// 空白文字を処理するDOMビルダを作成する
XML::DOMBuilder domBuilder(Loader.GetXMLSupport());
// または、空白文字を無視するDOMビルダを作成する
XML::DOMBuilder domBuilder2(Loader.GetXMLSupport(), true);
…
// このxmlファイルを処理して、DOMツリーのルートを返す
// (内部でオブジェクト マネージャを作成する)
Ptr<XML::Document> pdoc = domBuilder.ParseFile("inputfile.xml",
                                               mLoader.GetFileOpener());
// または、このxmlファイルを処理して、DOMツリーのルートを返す (提供されたオブジェクト マネージャを使用する)
Ptr<XML::ObjectManager> pobjMgr = *new XML::ObjectManager();
```

```
Ptr<XML::Document> pdoc2 = domBuilder.ParseFile("inputfile.xml",
                                                mLoader.GetFileOpener(), pobjMgr);
…
```

ActionScript XML サポートが有効ではない場合、パーサーのスタンドアロン インスタンスを作成し、以下のように使用することができます:

```
…
#include "GFx/XML/XML_DOM.h"
#include "GFx/XML/XML_Expat.h"
using namespace Scaleform;
…
Ptr<XML::Parser> pexpatXmlParser = *new XML::ParserExpat;
Ptr<XML::Support> pxmlSupport = *new XML::Support(pexpatXmlParser);
…
// 空白文字を処理するDOMビルダを作成する
XML::DOMBuilder domBuilder(pxmlSupport);
// 空白文字を無視するDOMビルダを作成する
XML::DOMBuilder domBuilder2(pxmlSupport, true);
…
// 前のセクションの処理と同じ
…
```

2.3 C++言語で DOM ツリーをトラバースする

以下のコードで DOM ツリーの内容が出力されます (Scaleform ソース SDK ライセンスが必要です):

```
…
using namespace Scaleform;
…
void PrintDOMTree(XML::Node* proot)
{
    switch (proot->Type)
    {
        case XML::ElementNodeType:
        {
            XML::ElementNode* pnode =
                static_cast< XML::ElementNode*>(proot);
            if (pnode->Prefix.GetSize() > 0)
            {
                printf("ELEM - '%s:%s' ns:'%s' prefix:'%s'"
                       " localname: '%s'",
                       pnode->Prefix.ToCStr(),
                       pnode->Value.ToCStr(),
                       pnode->Namespace.ToCStr(),
```

```
                    pnode->Prefix.ToCStr(),
                    pnode->Value.ToCStr());
            }
            else
            {
                printf("ELEM - '%s' ns:'%s' prefix:''"
                    " localname: '%s'",
                    pnode->Value.ToCStr(),
                    pnode->Namespace.ToCStr(),
                    pnode->Value.ToCStr());
            }
            for (XML::Attribute* attr = pnode->FirstAttribute;
                  attr != NULL; attr = attr->Next)
            {
                printf(" {%s,%s}", attr->Name.ToCStr(),
                        attr->Value.ToCStr());
            }
            printf("\n");
            for (XML::Node* child = pnode->FirstChild; child != NULL;
                  child = child->NextSibling)
                PrintDOMTree(child);
            break;
        }
        case TextNodeType:
        {
            printf("TEXT - '%s'\n", proot->Value.ToCStr());
            break;
        }
        default:
        {
            printf("UNKN\n");
        }
    }
}
…
Ptr<XML::Document> pdoc = domBuilder.ParseFile("inputfile.xml",
                                                Loader.GetFileOpener());
PrintDOMTree(root);
```

2.4 デバッグレポートの設定

Scaleform ソースファイルにアクセスするユーザーは、ＸＭＬ解析・ＤＯＭ構築レポートのフラグを指定するため XML_DOM.cpp にフラグを設定/解除することができます。

```
// デバッグ出力フラグ & トレース フラグ
//
#ifdef SF_BUILD_DEBUG
//
// デバッグ出力のドキュメント ビルダのDOMツリー構造をトレースする
//
// #define SF_XML_DOCBUILDER_TRACE
//
// 作成されたDOMツリーをすべて標準出力にダンプする
// (警告: 大きなファイルの場合はこれを避けること)
//
// #define SF_XML_DOCBUILDER_DOM_TREE_DUMP
//
#endif  // #ifdef SF_BUILD_DEBUG
```

3 カスタム パーサーの実装

Scaleform は expat ライブラリを使用したデフォルトのパーサーインプリメンテーションを提供します。ターゲットのアプリケーションにすでに XML パーサーがある場合は、XML_Support.h で定義される GFx::XML::Parser インターフェイスにそのパーサーをラップして、Scaleform XML サブシステムに取り込むことができます。XML_Parser.h ファイルは、GFx::XML::ParserHandler クラスを SAX2 インターフェイス メカニズムとして定義します。

```
namespace Scaleform { namespace GFx { namespace XML {
//
// 接続可能なXMLパーサー
//
// ファイルの解析/文字列の呼び出しごとに、パーサーのインスタンスを作成する。
// これはスレッド セーフティのために追加された。各パーサー インスタンスは、
// シングル スレッド内での使用が保証される。パーサーのインスタンスが、
// XMLステートで想定されている。
//
//
class Parser : public RefCountBaseNTS<Parser, Stat_Default_Mem>
{
  public:
    virtual ~Parser() {}

  // 解析メソッド
    virtual bool    ParseFile(const char* pfilename, FileOpenerBase* pfo,
                              ParserHandler* pphandler) = 0;
    virtual bool    ParseString(const char* pdata, UPInt len,
                                ParserHandler* pphandler) = 0;
};
}}} //SF::GFx::XML
```

すべての XML パーサー インプリメンテーションは、何らかの解析を実行する前に、ParseFile と ParseString メソッドに渡した DOM ビルダ オブジェクトに、GFx::XML::ParserLocator インスタンスを登録する必要があります。適切な解析イベントを行うには、補完的な GFx::XML::ParserHandler コールバック メソッドを、必要なパラメータで呼び出す必要があります。エラーが発生した場合、その重大性に応じて適切なエラー ハンドラ メソッドを呼び出す必要があります。

```
namespace Scaleform { namespace GFx { namespace XML {
//
// SAX2統合ハンドラ
//
// DOMビルダはSAX2パーサー ハンドラに似たインターフェイスである。
// このパーサー インプリメンテーションは、特定のイベントに対して適切なコールバック
// メソッドを呼び出すことが想定されている。
class ParserHandler : public RefCountBase<ParserHandler, StatMV_XML_Mem>
```

```
{
  public:
    ParserHandler() {}
    virtual ~ParserHandler() {}

  // ドキュメントの先頭と末尾
    virtual void        StartDocument() = 0;
    virtual void        EndDocument() = 0;

  // タグ エレメントの先頭と末尾
    virtual void        StartElement(const StringRef& prefix,
                                     const StringRef& localname,
                                     const ParserAttributes& atts) = 0;
    virtual void        EndElement(const StringRef& prefix,
                                   const StringRef& localname) = 0;

  // 名前空間の宣言。次のエレメントは、このマッピングの親となる。
    virtual void        PrefixMapping(const StringRef& prefix,
                                      const StringRef& uri) = 0;

  // タグ エレメント間のテキスト データ
    virtual void        Characters(const StringRef& text) = 0;

  // 空白文字
    virtual void        IgnorableWhitespace(const StringRef& ws) = 0;

  // 未処理のエレメント
    virtual void        SkippedEntity(const StringRef& name) = 0;

  // GFx::XML::Parserインプリメンタは、何らかのコールバックが起きる前に、
  // ドキュメント ロケータを設定する必要がある。GFx::XML::ParserHandlerインプリメンテーションは、
  // エラーの報告と、エンコーディング、xmlversion、さらにスタンドアロン プロパティを
  // 正しく処理するために、ロケータ オブジェクトが必要である。
    virtual void    SetDocumentLocator(const ParserLocator* plocator) = 0;

  // コメント
    virtual void        Comment(const StringRef& text) = 0;

  // ErrorHandlerコールバック
    virtual void        Error(const ParserException& exception) = 0;
    virtual void        FatalError(const ParserException& exception) = 0;
    virtual void        Warning(const ParserException& exception) = 0;

};
}}} //SF::GFx::XML
```

4 既知の制限

4.1 ActionScript にカスタム *XML* ノード プロパティが存在しない

ActionScript XMLNode (および XML) オブジェクトへの参照がすべて消えてしまい、再度アクセスした場合、そのオブジェクトに割り当てられたカスタム プロパティは存続しません。すべてのリファレンスが消えた場合、この ActionScript オブジェクトだけが削除されます。背後の DOM ツリーは存続します。ただし、カスタム プロパティは DOM ではなく ActionScript オブジェクトに適用されているため、プロパティの寿命はその ActionScript オブジェクトの有効期間に直接左右されます。

このような事態は、ドキュメントが XML.load から読み込まれた場合に発生します。XML.load は、背後の完全な DOM ツリーをシャドウする最上部の XMLNode オブジェクトを作成します。ユーザーが一時的な XMLNode 参照を使用して ActionScript で DOM ツリーをトラバースし、カスタム プロパティをそのような一時参照に割り当てた場合、後でアクセスした時にはもうこのようなプロパティは存在しなくなります。例 (XML、または妥当な XML データを持つ XMLNode オブジェクトがルートと仮定):

```
// DOMノードのリファレンスを取得する
XMLNode temp = root.firstChild;
// そのノードにカスタムのプロパティを設定する
temp.someProperty = someValue;
// リファレンス (すべてのリファレンス) を破棄する
XMLNode temp = temp.nextSibling;
// リファレンスを戻す
temp = temp.prevSibling;
// カスタムのプロパティを探す
trace(temp.someProperty)
```

Flash®では someValue が出力されますが、Scaleform では出力は未定義になります。この XMLNode オブジェクトは、以前に割り当てられたプロパティをもう保有していないからです。カスタム プロパティの割り当てや trace ステートメントを通じて、そのプロパティに対する参照が持続していれば、プロパティは存続します。

注意:これは、XMLNode に付属する属性オブジェクトには影響しません。ActionScript XMLNode 参照のステートに関係なく、このオブジェクトにプロパティを設定すると、そのまま持続されます。例えば:

```
// DOMノードのリファレンスを取得する
XMLNode temp = root.firstChild;
// そのノードにカスタムの属性を設定する
temp.attributes.someProperty = someValue;
// リファレンス (すべてのリファレンス) を破棄する
XMLNode temp = temp.nextSibling;
```

```
// リファレンスを戻す
temp = temp.prevSibling;
// カスタムの属性を探す
trace(temp.attributes.someProperty)
```

Scaleform で someValue が出力されます。

4.2 ActionScript XML.status プロパティが存在しない

カスタムのパーサー ライブラリのエラー コードを、ActionScript 2.0 XML のエラー コードにマッピングするときにさまざまな不一致項目が発生するため、このプロパティは実装されませんでした。

ActionScript エラー コード:

0 エラーなし; 解析は問題なく完了しました。

-2 CDATA セクションが適切に終了していません。

-3 XML 宣言が適切に終了していません。

-4 DOCTYPE 宣言が適切に終了していません。

-5 コメントが適切に終了していません。

-6 XML エレメントの形式に問題がありました。

-7 メモリが不足しています。

-8 属性値が適切に終了していません。

-9 開始タグと終了タグが一致しませんでした。

-10 一致する開始タグのない終了タグが検出されました。

Expat から ActionScript へのマッピング：

```
//
// XML_ERROR_NONE           = 0
// XML_ERROR_INVALID_TOKEN  = 0 (例: XML.parse("http://www.google.com"))
// XML_ERROR_UNCLOSED_CDATA_SECTION = -2
// XML_ERROR_UNCLOSED_TOKEN = -3
// XML_ERROR_INVALID_TOKEN  = -4
// XML_ERROR_INVALID_TOKEN  = -5
// XML_ERROR_UNCLOSED_TOKEN = -8
// XML_ERROR_NO_ELEMENTS    = -9
// XML_ERROR_TAG_MISMATCH   = -10
//
```

エラー コードを 1 対 1 でマッピングしなければ、一貫性のあるステータス プロパティを実装することはできません。*Scaleform::XML::DOMBuilder* は冗長なエラー メッセージを表示し、XML パーサー インスタンスによって登録されている *GFx::XML::ParserLocator* オブジェクトのデータを使用して、出力をデバッグします。

4.3 ActionScript XML.docTypeDecl プロパティが存在しない

Flash®の関連資料より:「ActionScript の XML パーサーは、妥当性検証用パーサーではありません。DOCTYPE 宣言はパーサーにより読み取られ、XML.docTypeDecl プロパティに格納されますが、DTD の妥当性検査は行われません。」このプロパティは Flash にも Scaleform にも必要ないので、削除されています。