Autodesk®
Scaleform®

Scaleform 4.2 AS2 Extensions Reference

本書では、Scaleform 4.2 で使用できる ActionScript 2.0 のエクステンションについて説明しています。

著者: Artem Bolgar, Prasad Silva
バージョン: 4.03
最終更新日: 2012 年 9 月 17 日

Autodesk®
GAMEWARE

Copyright Notice

Autodesk® Scaleform® 4.2

© 2012 Autodesk, Inc. All rights reserved. Except as otherwise permitted by Autodesk, Inc., this publication, or parts thereof, may not be reproduced in any form, by any method, for any purpose.

Certain materials included in this publication are reprinted with the permission of the copyright holder.

The following are registered trademarks or trademarks of Autodesk, Inc., and/or its subsidiaries and/or affiliates in the USA and other countries: 123D, 3ds Max, Algor, Alias, AliasStudio, ATC, AUGI, AutoCAD, AutoCAD Learning Assistance, AutoCAD LT, AutoCAD Simulator, AutoCAD SQL Extension, AutoCAD SQL Interface, Autodesk, Autodesk Homestyler, Autodesk Intent, Autodesk Inventor, Autodesk MapGuide, Autodesk Streamline, AutoLISP, AutoSketch, AutoSnap, AutoTrack, Backburner, Backdraft, Beast, Beast (design/logo) Built with ObjectARX (design/logo), Burn, Buzzsaw, CAiCE, CFdesign, Civil 3D, Cleaner, Cleaner Central, ClearScale, Colour Warper, Combustion, Communication Specification, Constructware, Content Explorer, Creative Bridge, Dancing Baby (image), DesignCenter, Design Doctor, Designer's Toolkit, DesignKids, DesignProf, DesignServer, DesignStudio, Design Web Format, Discreet, DWF, DWG, DWG (design/logo), DWG Extreme, DWG TrueConvert, DWG TrueView, DWFX, DXF, Ecotect, Evolver, Exposure, Extending the Design Team, Face Robot, FBX, Fempro, Fire, Flame, Flare, Flint, FMDesktop, Freewheel, GDX Driver, Green Building Studio, Heads-up Design, Heidi, Homestyler, HumanIK, i-drop, ImageModeler, iMOUT, Incinerator, Inferno, Instructables, Instructables (stylized robot design/logo),Inventor, Inventor LT, Kynapse, Kynogon, LandXplorer, Lustre, MatchMover, Maya, Mechanical Desktop, MIMI, Moldflow, Moldflow Plastics Advisers, Moldflow Plastics Insight, Moondust, MotionBuilder, Movimento, MPA, MPA (design/logo), MPI (design/logo), MPX, MPX (design/logo), Mudbox, Multi-Master Editing, Navisworks, ObjectARX, ObjectDBX, Opticore, Pipeplus, Pixlr, Pixlr-o-matic, PolarSnap, Powered with Autodesk Technology, Productstream, ProMaterials, RasterDWG, RealDWG, Real-time Roto, Recognize, Render Queue, Retimer, Reveal, Revit, RiverCAD, Robot, Scaleform, Scaleform GFx, Showcase, Show Me, ShowMotion, SketchBook, Smoke, Softimage, Sparks, SteeringWheels, Stitcher, Stone, StormNET, Tinkerbox, ToolClip, Topobase, Toxik, TrustedDWG, T-Splines, U-Vis, ViewCube, Visual, Visual LISP, Vtour, WaterNetworks, Wire, Wiretap, WiretapCentral, XSI.

All other brand names, product names or trademarks belong to their respective holders.

Disclaimer
THIS PUBLICATION AND THE INFORMATION CONTAINED HEREIN IS MADE AVAILABLE BY AUTODESK, INC. “AS IS.” AUTODESK, INC. DISCLAIMS ALL WARRANTIES, EITHER EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE REGARDING THESE MATERIALS.

Autodesk Scaleform の連絡先:

ドキュメント	ActionScript Extensions Reference (ActionScript エクステンション リファレンス)
住所	Scaleform Corporation 6305 Ivy Lane, Suite 310 Greenbelt, MD 20770, USA
ホームページ	http://www.scaleform.com/
電子メール	info@scaleform.com
電話	(301) 446-3200
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 は元来、Web 用に設計されており、ゲームやアプリケーションの開発用ではないため、フォーカス処理、マウスのサポート、テキスト フィールドのサポート、IME 入力の処理など、一部の分野で機能が制限されていました。Scaleform は、ActionScript のクラスにいわゆる「エクステンション」を追加して、このような分野の基本的な機能性を改善しています。Scaleform Player でエクステンションをサポートするには、`_global.gfxExtensions` 変数を `true` に設定してください。

```
_global.gfxExtensions = true;
```

ほとんどの場合、開発者はこのステートメントを、エクステンションを使用する FLA ファイルの最初のフレームに追加したいと思うでしょう。この変数が設定されない場合、Scaleform Player はエクステンションに対するすべてのリファレンスを無視して、Flash との互換性を最大限に達成しようとします。

開発者は、本書のエクステンションの機能は標準の Flash® Player では動作しないことを理解しておいてください。これは Scaleform に限って実装されるからです。それぞれのエクステンションには Scaleform のバージョン番号が付随しており、エクステンションが追加された Scaleform Player SDK のリリース番号を識別します。エクステンションは、それより前のバージョン番号の Scaleform Player では動作しません。

Scaleform はエクステンションの API が常にサポートされるように、さらにリリース番号が異なっていても一貫するように最大限の努力を払いますが、今後リリースされる Scaleform では、エクステンションの API の変更、名称変更、または削除を行う場合があります。重要な変更が行われる場合、その変更をできるだけメジャー バージョン番号のリリース時に行い、可能な限り早い時期に告知するように努めます。

2 Mouse クラスのエクステンション

標準の Mouse クラス メソッドのフルセットをサポートするだけでなく、Scaleform は複数のマウス カーソルのトラッキングと、右、中央、その他のボタン (現在最大 16 個) を検出できるエクステンションも導入しています。この章ではまず、基本的な Mouse 機能の一部について触れ、次に Scaleform マウス エクステンションの詳細について説明します。Mouse オブジェクトの詳細については、Flash の Mouse クラスの資料を参照することをお勧めします。

2.1 複数のカーソルのサポート

Wii などの一部のプラットフォームでは、複数のマウス カーソルをサポートする必要があります。Scaleform 4.0 は最大で 4 つのマウス カーソルをサポートします。
GFx::Movie::SetMouseCursorCount(unsigned n) メソッドは、カーソルの数を設定します。パラメータとして GFx::MouseEvent オブジェクトを伴う GFx::Movie::HandleEvent メソッドを使って、マウス イベントを Scaleform コアに送る必要があります。GFx::MouseEvent と GFx::MouseCursorEvent のコンストラクタは、マウスのゼロベースのインデックスを指定する追加パラメータを備えています。

2.2 Button イベントのエクステンション

onRollOver、onRollOut、onDragOver、onDragOut、onPress、onRelease、onReleaseOutside などの Button イベントは、Scaleform エクステンションがオンの場合、1 つか 2 つのパラメータを受け取ります。

最初のパラメータは、そのイベントを発生させたマウスのゼロベースのインデックスです。したがって、onPress イベントがインデックス 1 のマウス カーソルによって発生した場合、このパラメータは以下のように 1 という値を含みます:

```
mc.onPress = function(mouseIdx:Number)
{
      if (mouseIndex == 0)
            . . .
      else if (mouseIndex == 1)
            . . .
      . . .
}
```

onPress および onRelease の 2 番目のパラメータは、イベントがマウス／カーソルまたはキーボードによって引き起こされたことを表す NUMERIC プロパティです。-1 がキーボード、0 がマウス／カーソルを表します。このプロパティはイベントの原因を判断するのに有用です。

```
mc.onPress = function(mouseIdx:Number, keyboardOrMouse:Number)
```

```
{
	if (keyboardOrMouse == 0)
		. . .
	else
		. . .
	. . .
}
```

2 番目のパラメータは、onRollOver/Out イベントと onDragOver/Out イベントのオプションです。このパラメータは、同じ文字上のネストされた rollover/dragover イベントのインデックスを指定します。このパラメータが関数ハンドラに対して宣言されていない場合、onRollOver/onRollOut イベントと onDragOver/onDragOut イベントのペアが一度だけ発生します。onRollOver/onDragOver は、最初のカーソルがその文字上に移動したときに発生し、onRollOut/onDragOut は最後のカーソルがその文字を離れたときに発生します。

2 番目のパラメータが、これらのハンドラに対して宣言される場合、ネストされた onRollOver/onRollOut イベントと onDragOver/onDragOut イベントが生成され (それぞれのマウス カーソルに対して別々に)、さらにそのパラメータはネストのゼロベースのインデックスを表します。最初の onRollOver/onDragOver イベントは、2 番目のパラメータとして 0 を持ちます。2 番目のカーソルが同じ文字上に移動すると、2 番目の onRollOver/onRollOut イベントは、1 に設定された 2 番目のパラメータで発生します。カーソルが文字を離れると、onRollOut/onDragOut は 1 に設定された 2 番目のパラメータで発生します。最後の onRollOut/onDragOut イベントは、0 に設定された 2 番目のパラメータで発生します。

したがって、2 番目のパラメータの宣言は、onRollOver/onRollOut イベントと onDragOver/onDragOut イベントが発生する方法を変更することになります。2 番目のパラメータが宣言される場合、ネストされたイベントの処理はユーザーの決定事項です。

```
mc.onRollOver = function(mouseIdx:Number, nestingIdx:Number)
{
	// nestingIdx == 0  の時だけロールオーバー アニメーションを行う
	if (nestingIdx == 0)
		doOverAnimation();
	. . .
}
mc.onRollOut = function(mouseIdx:Number, nestingIdx:Number)
{
	// nestingIdx == 0  の時だけロールアウト アニメーションを行う
	if (nestingIdx == 0)
		doOutAnimation();
	. . .
}
```

on(rollOver)、on(rollOut) などの旧式のイベントは、追加パラメータ付きで提供されないので、どのマウス カーソルによってそのイベントが発生したのかを区別する方法はありません。したがって、このようなイベントを、複数のマウス カーソルで使用することはお勧めしません。

onPress や onRelease などの左ボタン以外のマウスボタンをサポートするために、Button Events の補助バージョンを追加しています。

- onPressAux
- onReleaseAux
- onReleaseOutsideAux
- onDragOver
- onDragOut

イベントターゲットでこれら補助イベントハンドラーを定義すると、これら補助イベントハンドラーが呼び出されます。呼び出されるのは index != 0 のボタンに対してのみです（0 インデックスは左マウスボタンを表す）。補助イベントハンドラーの有無によらず、標準ハンドラーは左マウスボタンのみに対して呼び出されます。補助イベントハンドラーは対応する標準ハンドラーと同一の関数シグネチャを持っており、その詳細はこのセクションの前半で定義したとおりです。

```
mc.onPressAux = function(mouseIdx:Number, keyboardOrMouse:Number, buttonIdx:Number)
{
      if (buttonIdx == 1)  // Right mouse button
            . . .
      . . .
}
```

2.3 Mouse クラス イベント

ActionScript では、マウス リスナーをインストールして、マウスの移動、マウスの左ボタンの押下、マウス ホイールなどのイベントの通知を受け取ることができます。Flash では、このようなイベントは従来、Mouse.addListener メソッドで処理され、以下のメソッドを実装するリスナー オブジェクトをインストールします:

- onMouseMove   = function() { }
- onMouseDown   = function() { }
- onMouseUp   = function() { }
- onMouseWheel = function(delta : Number, targetPath : String) { }

このリスナー オブジェクトがインストールされた後で、そのオブジェクトの適切なメソッドが、対応するイベントが発生したときは常に呼び出されます。Flash では、onMouseDown メソッドと onMouseUp メソッドは、マウスの左ボタンに対してのみ呼び出され、他のマウス ボタン イベントの受け取りに対して、パブリック インターフェイスは提供されません。

この制限を解消するため (さらに、複数のマウス カーソルをサポートするため)、Scaleform は、このようなメソッドの呼び出しが、_global.gfxExtensions 変数が true に設定されたときは、追加引数を取ることを許可して、それらの呼び出しを拡張します。この新規関数のシグネチャは以下のとおりです:

- `onMouseDown  = function(button : Number, [targetPath : String], [mouseIdx : Number], [x : Number], [y : Number] , [dbClick : Boolean]) { }`
- `onMouseUp  = function(button : Number, [targetPath : String], [mouseIdx : Number], [x : Number], [y : Number]) { }`

割り当てられたリスナー関数が、少なくとも 1 つの追加引数を取るとき、Scaleform はマウスの右ボタン、中央ボタン、さらにその他のマウス ボタンに対してもそのリスナー メソッドを呼び出し、開発者自身のロジックでそのようなイベントを検出できるようにします。マウスの左ボタンの値は 1、マウスの右ボタンは 2、マウスの中央ボタンは 3 です。現在、最大で 16 個までのボタンがサポートされています。互換性のために、この関数が引数を取らないように宣言された場合、Flash と同様に、マウスの左ボタンに限ってこの関数は呼び出されます。以下はこのようなハンドラのインストール例です:

```
var mouseListener:Object = new Object;

mouseListener.onMouseDown = function(button, target)
{
   trace("mouseDown - button '" + button +
         "'  target = '" + target + "'");
}
mouseListener.onMouseUp = function(button, target)
{
   trace("mouseUp - button '" + button +
         "'  target = '" + target + "'");
}

Mouse.addListener(mouseListener);
```

この新規の mouseDown/Up ハンドラは、最低でも button と target の 2 つの引数を取ります。最初の引数である button は、押下されたマウス ボタンを表します。Mouse["LEFT"]、Mouse["RIGHT"]、Mouse["MIDDLE"] エクステンション定数と button を比較することで、そのボタンを解釈できます (Mouse["LEFT"] シンタックスが、ActionScript 2.0 コンパイラ (エラー メッセージ等) を抑制するために、Mouse.LEFT の代わりに使用されます)。2 番目の引数である target は、マウスが押下された時点でそのマウス カーソルの下にある最上位のオブジェクトへのパスを提供します。このようなパスが利用できない場合は、'undefined' にすることができます。

Flash では、追加引数はすべて、常に'undefined' です。ハンドラはマウスの左ボタンに限って呼び出されるからです。この 'undefined' という値をマウスの左ボタンとして解釈し、Flash Player との互換性を確保することができます。

複数のマウス カーソルをサポートするために、各ハンドラに対してさらに追加パラメータが存在します。

- `onMouseMove = function([mouseIdx : Number], [x : Number],[y : Number]) { }`
- `onMouseDown = function(button : Number, [targetPath : String],`

```
                        [mouseIdx :    Number], [x : Number], [y : Number] ,
        [dbClick : Boolean]) { }
```

- `onMouseUp   = function(button : Number, [targetPath : String],`
  `[mouseIdx : Number], [x : Number], [y : Number]) { }`
- `onMouseWheel = function(delta : Number, targetPath : String,`
  `[mouseIdx : Number], [x : Number], [y : Number]) { }`

パラメータ mouseIdx は、そのイベントを発生させたマウス カーソルのゼロベースのインデックスを含んでいます。

x と y のパラメータは (_root 座標空間の) マウス カーソルの座標を含んでいます。
onMouseDown ハンドラの dblClick パラメータは、ダブルクリックされたかどうかを表します。ダブルクリックされた場合、このパラメータは 'true' になります。

2.4 マウス カーソルの変更

Flash では以下の三種類のマウス カーソルが存在します:

- 矢印 – カーソルが最もノーマルなオブジェクトの上にあるときに使用されます。
- ハンド – useHandCursor プロパティが true の場合に、ボタンとリンクの上で使用されます。
- I ビーム – カーソルがテキスト フィールド上にあるときに使用されます。

Flash はカーソルの変化を簡単に検知する方法を備えていませんが、Scaleform はカスタムのカーソルを管理できる C++ API と、ActionScript Mouse クラス エクステンションの両方を提供しています。C++では、GFx::StateBag::SetUserEventHandler 関数を使って、マウス カーソルの変更が必要な場合は常に通知するイベント ハンドラをインストールすることができます。FxPlayer.cpp で表しているように、このハンドラを使って、ゲーム エンジンが描画するカーソル イメージ、またはシステム カーソルのいずれも変更することができます。あるいは、全く ActionScript からカスタム カーソルを実装するような選択をすることも可能です。そうすれば、アニメーションやデザイナーによる コントロールといった利点を追加することもできます。この方法の例は、SDK に添付されている新規の *Mouse.fla/swf* サンプルで提供しています。

この Mouse サンプルは、カスタマイズ可能なマウス カーソルを、'Cursor_M'と呼ばれる簡単なムービー クリップ オブジェクトとして、ステージ上に実装しています。このカーソルは "hand"、"arrow"、"ibeam"というフレーム ラベルで識別される複数のフレームを備えており、カーソル イメージは、タイムラインをそのようなフレームの 1 つに移動するだけで変更できます。マウスの移動が検知されたときは常に、このムービー クリップの_x 座標と_y 座標が調整されて、そのマウスに従います。

その下にあるオブジェクトに基づいてカーソルの種類を変えるために、この Mouse サンプルは、Scaleform エクステンションである Mouse.setCursorType 関数をオーバーライドします。この関数は、カーソルの変更が必要な場合は常に、cursorType と mouseIndex 引数で呼び出され、そのような変更を検知します。この Mouse サンプルは、cursorType をいずれかのマウス定数

(Mouse["ARROW"]、Mouse["HAND"]、Mouse["IBEAM"]) と比較し、カーソルのフレームを適切に変更して、このサンプルを解釈します。

以下はこの Mouse サンプルで、複数のカスタム カーソルの処理を担うソース コードです。詳細については、実際のサンプルを参照してください。

```
_global.gfxExtensions = 1;
// システム カーソルを隠す
Mouse.hide();

var mouseListener:Object = new Object;
mouseListener.onMouseMove = function(mouseIdx, x,y)
{
    if (mouseIdx == undefined)
  mouseIdx = 0;
    if (x == undefined)
    {
  x = _xmouse;
        y = _ymouse;
    }
    var cmc = eval("_root.Cursor_M"+(mouseIdx+1));

    cmc._x = x;
    cmc._y = y;
}
Mouse.addListener(mouseListener);

Mouse["setCursorType"] = function(cursorType, mouseIdx)
{
   if (mouseIdx == undefined)
      mouseIdx = 0;
   var cmc = eval("_root.Cursor_M"+(mouseIdx+1));
   switch(cursorType)
   {
      case Mouse["HAND"]:
         cmc.Cursor.gotoAndPlay("hand");
         break;
      case Mouse["ARROW"]:
         cmc.Cursor.gotoAndPlay("arrow");
         break;
      case Mouse["IBEAM"]:
         cmc.Cursor.gotoAndPlay("ibeam");
         break;
      default: return;
   }
}
```

```
// Mouse.show と Mouse.hide 関数をオーバーライドして
// 自分のカーソルに作用するようにする
ASSetPropFlags(Mouse, "show,hide", 0, 7);

Mouse.show = function(mouseIdx)
{
    if (mouseIdx == undefined)
        mouseIdx = 0;
    var cmc = eval("_root.Cursor_M"+(mouseIdx+1));
    cmc._visible = true;
}
Mouse.hide = function(mouseIdx)
{
    if (mouseIdx == undefined)
        mouseIdx = 0;
    var cmc = eval("_root.Cursor_M"+(mouseIdx+1));
    cmc._visible = false;
}
```

2.5 Mouse クラスの ActionScript エクステンション

Mouse ActionScript クラスは、複数の静的メソッドとプロパティを追加して、Scaleform で拡張されています。ActionScript 1.0 では、Mouse.setCursorType(…) で行うように、直接エクステンションを参照することができます。ActionScript 2.0 はそれほど寛容ではないので、このような呼び出しには不満を表します。ActionScript 2.0 コンパイラを抑制するために、別のアクセス シンタックスである、Mouse["setCursorType"](..) を使用する必要があります。

getButtonsState() 静的メソッド

```
public function getButtonsState(mouseIndex : Number) : Number
```

Scaleform のバージョン: 2.2 以降

マウス ボタンのステートを返します。戻り値はビットマスクで、ビットのインデックスはゼロベースのマウス ボタン インデックスと同じです。対応するビットが設定されている場合は、ボタンが押下されます。

パラメータ

`mouseIndex : Number` – ゼロベースのマウス インデックス

getTopMostEntity() 静的メソッド

```
public function getTopMostEntity([mouseIndex : Number]) : Object
public function getTopMostEntity(testAll : Boolean,
                                 [mouseIndex : Number]) : Object
public function getTopMostEntity(x : Number, y : Number) : Object
public function getTopMostEntity(x : Number, y : Number, testAll: Boolean) : Object
```

Scaleform のバージョン:1.2.34、複数のマウス カーソルのサポートは 2.2 以降

この静的メソッドは、マウス カーソルの下、または指定された座標にあるターゲットの文字を返します。また、このメソッドはボタン ハンドラ (onPress、onMouseDown、onRollOver など) 付きの文字と、そのようなハンドラがない文字で異なる場合もあります。この区別は、マウス イベントを処理するハンドラを持っていない文字を除外するのに便利な場合があります。

このメソッドは MovieClip.hitTest とは逆になります。hitTest は x と y の座標が、特定のオブジェクトの内部にあるかどうかをチェックしますが、この getTopMostEntity は指定された x と y の座標にある実際のオブジェクトを戻すからです。したがって Mouse.getTopMostEntity(x, y).hitTest(x, y, true) == true ということになります。

パラメータ

`mouseIndex : Number` – ゼロベースのマウス インデックス

`testAll : Boolean` – ボタン ハンドラ付きの文字のみを検索する(false) か、または任意の文字を検索する (true) かを示します。このパラメータが指定されていない場合、`getTopMostEntity` は'true'であると仮定します。

`x : Number, y :Number` – 検索する文字が位置する別の座標

例:

```
var listenerObj = new Object;

listenerObj.onMouseMove = function()
{
   var target = Mouse["getTopMostEntity"]();
   trace("Mouse moved, target = "+target);
}

Mouse.addListener(listenerObj);

var target = Mouse["getTopMostEntity"](480, 10);
trace("The character at the (480, 10) is " + target);
```

getPosition 静的メソッド

```
public function getPosition(mouseIndex : Number) : flash.geom.Point
```

Scaleform のバージョン: 2.2 以降

このメソッドは、_root 座標空間の対応するマウス カーソルの座標を返します。この戻り値は、flash.geom.Point のインスタンスです。

パラメータ

`mouseIndex : Number` – ゼロベースのマウス インデックス

setCursorType() 静的メソッド

```
public function setCursorType(cursorType : Number, [mouseIndex : Number]) : void
```

Scaleform のバージョン: 1.2.32 以降

この静的メソッドは、cursorType パラメータに応じてマウス カーソルを変更します。詳細は「マウス カーソルの変更」を参照してください。

パラメータ

`cursorType :Number` – Mouse.ARROW、Mouse.HAND、Mouse.IBEAM のいずれかのマウス カーソルのタイプを表します。

`mouseIndex :Number` – ゼロベースのマウス インデックス

3 Button クラス エクステンション

hitTestDisable プロパティ

```
hitTestDisable:Boolean  [read-write]
```

Scaleform のバージョン: 2.1.51 以降

true に設定されると、MovieClip.hitTest 関数は、ヒット テストの検出中はこのボタンを無視します。さらに、その他のマウス イベントもすべて、そのボタンには伝わりません。

デフォルト値は false です。

関連項目:

```
TextField.hitTestDisable
MovieClip.hitTestDisable
```

topmostLevel プロパティ

```
topmostLevel:Boolean  [read-write]
```

Scaleform のバージョン: 2.1.50 以降

このプロパティが true に設定された場合、文字はその深度に関係なく、他のすべての文字の上に表示されます。カスタムのマウス カーソルを全レベルのオブジェクトの上に描画したい場合に、カーソルの実装に使えることもあります。デフォルト値は false です。

"topmostLevel”として、いくつかの文字にマークしたい場合には、描画順は以下のとおりになります：

- Scaleform3.0.71 までは、”topmostLevel”とマークした文字の描画順は、このプロパティを true と設定した順に依存します（ということは、最初に topmost とマークされた文字が最初に描画されます。）;
- Scaleform3.0.72 以降では、文字に topmost マーキングを行わなかったと同じ順に描画されます。つまり、オブジェクト A がオブジェクト B の下に描画されていたとすれば、topmost マーキングを行った後でも、オブジェクト A はオブジェクト B の下に描画されます。"topmostLevel"プロパティを true と設定した順序には関わりません。

注意： "topmostLevel"としてマークされると、マークされた文字に対して、SwapDepth ActionScript 関数は、有効性を持たなくなります。

関連項目:

```
TextField.topmostLevel
MovieClip.topmostLevel
```

focusGroupMask プロパティ

```
focusGroupMask : Number
```

Scaleform のバージョン: 3.3.84

ビットマスクをステージ・キャラクタと**すべての**チャイルドに設定するプロパティです。このビットマスクによって、フォーカスグループのオーナーシップがキャラクタに割り当てられます。すなわち、ビットマスクで示されるコントローラのみが、キャラクタにフォーカスを移動したりキャラクタ内部でフォーカスを移動できることを意味します。フォーカスグループは `setControllerFocusGroup` エクステンションメソッドを使ってコントローラに対応付けることができます。

たとえば、「button1」がコントローラ 0 からのみフォーカス可能で、「movieclip2」がコントローラ 0 とコントローラ 1 からフォーカス可能と仮定します。このような挙動を実現するには、フォーカスグループとコントローラとを以下のように対応付けます。

```
Selection.setControllerFocusGroup(0, 0);
Selection.setControllerFocusGroup(1, 1);

button1.focusGroupMask = 0x1; // bit 0 – focus group 0
movieclip2.focusGroupMask = 0x1 | 0x2 // bits 0 and 1 – focus groups 0 and 1
```

「`focusGroupMask`」ビットマスクは親ムービークリップに設定される場合があります。その場合、マスク値はすべてのチャイルドに伝搬します。

4 MovieClip クラス エクステンション

hitTest () メソッド

```
public hitTest(x:Number, y:Number, [shapeFlag:Boolean],
               [ignoreInvisibleChildren:Boolean]):Boolean
```

Scaleform のバージョン: 2.1.51 以降

標準の 3 つのパラメータを持つ hitTest は、エクステンションとしてさらにもう 1 つオプションでブール値のパラメータを備えており、これは非表示の子クリップを無視するか否かを示します。x, y 座標のポイントが非表示の子クリップに属しているとしても、hitTest のデフォルトの Flash の動作は “true”を返します。 (つまり、その_visible プロパティは false に設定されています。ゼロに設定された_alpha を持つ子クリップは、非表示として処理されません)。

パラメータ

`ignoreInvisibleChildren:Boolean` – 非表示のすべての子クリップを無視するには、true に設定する必要があります。

hitTestDisable プロパティ

```
hitTestDisable:Boolean  [read-write]
```

Scaleform のバージョン: 1.2.32 以降

true に設定すると、MovieClip.hitTest 関数は、ヒット テストの検出中はこのムービー クリップを無視します。さらに、その他のマウス イベントもすべて、そのムービー クリップには伝播されません。

デフォルト値は false です。

関連項目:

`TextField.hitTestDisable`
`Button.hitTestDisable`

noAdvance プロパティ

```
noAdvance : Boolean
```

Scaleform のバージョン: 2.1.52 以降

true に設定された場合、このプロパティはこのムービー クリップとそのすべての子クリップの進行をオフにします。これは、パフォーマンスの向上に使用できる場合があります。Flash は常にムービー クリップを進めます (タイムライン アニメーションの実行、フレームの ActionScript コードの呼び出しなど)。したがって、このプロパティを “true” に設定すると、Scaleform と Flash の動作に違いが出る場合があります。

関連項目:

```
_global.noInvisibleAdvance
```

topmostLevel プロパティ

```
topmostLevel:Boolean  [read-write]
```

Scaleform のバージョン: 2.1.50 以降

このプロパティが true に設定された場合、文字はその深度に関係なく、他のすべての文字の上に表示されます。カスタムのマウス カーソルを全レベルのオブジェクトの上に描画する必要がある場合、これはそのようなカーソルの実装に便利なこともあります。

デフォルト値は false です。

"topmostLevel”として、いくつかの文字にマークしたい場合には、描画順は以下のとおりになります：

- Scaleform3.0.71 までは、”topmostLevel”とマークした文字の描画順は、このプロパティを true と設定した順に依存します（ということは、最初に topmost とマークされた文字が最初に描画されます。）；
- Scaleform3.0.72 以降では、文字に topmost マーキングを行わなかったと同じ順に描画されます。つまり、オブジェクト A がオブジェクト B の下に描画されていたとすれば、topmost マーキングを行った後でも、オブジェクト A はオブジェクト B の下に描画されます。"topmostLevel"プロパティを true と設定した順序には関わりません。

注意： "topmostLevel"としてマークされると、マークされた文字に対して、SwapDepth ActionScript 関数は、有効性を持たなくなります。
関連項目:

```
TextField.topmostLevel
Button.topmostLevel
```

rendererString プロパティ

```
rendererString:String  [read-write]
```

Scaleform のバージョン: 2.2.55 以降

このプロパティを使って、任意の MovieClip インスタンスのために、カスタムのディレクティブを ActionScript からレンダラーに送ることができます。このプロパティが設定されている場合、ストリング値は、ユーザー データとしてレンダラーに送信されます。

デフォルト値はありません。

例:

```
myMovieInstance.rendererString = “SHADER_Blur”
```

rendererFloat プロパティ

```
rendererFloat:Number  [read-write]
```

Scaleform のバージョン: 2.2.55 以降

このプロパティを使って、任意の MovieClip インスタンスのために、カスタムのディレクティブを ActionScript からレンダラーに送ることができます。このプロパティが設定されている場合、フロート値は、ユーザー データとしてレンダラーに送信されます。

デフォルト値はありません。

例:

```
myMovieInstance.rendererFloat = 4; // 例えば、C++では列挙値です
```

disableBatching プロパティ

```
disableBatching:Boolean
```

Scaleform のバージョン: 4.0.17

このプロパティは、カスタム描画に対するメッシュ生成のバッチ処理を無効にします。

focusGroupMask プロパティ

```
focusGroupMask : Number
```

Scaleform version: 3.3.84

ビットマスクをステージ・キャラクタと**すべての**チャイルドに設定するプロパティです。このビットマスクによって、フォーカスグループのオーナーシップがキャラクタに割り当てられます。すなわち、ビットマスクで示されるコントローラのみが、キャラクタにフォーカスを移動したりキャラクタ内部でフォーカスを移動できることを意味します。フォーカスグループは `setControllerFocusGroup` エクステンションメソッドを使ってコントローラに対応付けることができます。

たとえば、「button1」がコントローラ 0 からのみフォーカス可能で、「movieclip2」がコントローラ 0 とコントローラ 1 からフォーカス可能と仮定します。このような挙動を実現するには、フォーカスグループとコントローラとを以下のように対応付けます。

```
Selection.setControllerFocusGroup(0, 0);
Selection.setControllerFocusGroup(1, 1);

button1.focusGroupMask = 0x1; // bit 0 – focus group 0
movieclip2.focusGroupMask = 0x1 | 0x2 // bits 0 and 1 – focus groups 0 and 1
```

「focusGroupMask」ビットマスクは親ムービークリップに設定される場合があります。その場合、マスク値はすべてのチャイルドに伝搬します。

5 NetStream クラス エクステンション

Scaleform はビデオ ファイルに埋め込まれたサブタイトルと、追加オーディオ トラックの操作を許可する NetStream クラスに、機能を追加します。

onMetaData イベント ハンドラ

```
onMetaData = function(info:Object) { }
```

Scaleform のバージョン: 3.0.63 以降

このイベント ハンドラは再生中のビデオ ファイルの情報を受け取り、 Scaleform に 2 つの追加オブジェクト プロパティをインクルードします。これらのプロパティは、ビデオ ファイルにエンコードされているサブタイトル トラックとオーディオ トラックに関する情報の取得に使用されます。

`onMetaData` イベント ハンドラに渡されるオブジェクトには、2 つの読み取り専用プロパティが追加されています。

1. `audioTracks` – ビデオ ファイルにエンコードされているオーディオ トラックを記述するオブジェクトの配列です。
   - `audioTracks` 配列の各オブジェクトは以下のプロパティを備えています:
     - `channelsNumber` – オーディオ トラックのチャンネル数です (1-モノラル、2- ステレオ、6 – 5.1 サラウンド)
     - `totalSamples` – このトラックのサウンド サンプルの数です。
     - `trackIndex` – トラックのインデックス番号です。インデックス番号を `NetStream.audioTrack` プロパティに割り当てることによって (以下の例を参照してください)、このプロパティはオーディオ トラックの選択に使用されます。
     - `sampleRate` – サウンド サンプル レートです。

2. `subtitleTracksNumber` – そのビデオ ファイルで有効なサブタイトル トラックの数です。

audioTrack プロパティ

```
audioTrack:Number  [read-write]
```

Scaleform のバージョン: 3.0.63 以降

このプロパティは、ビデオ ファイルにエンコードされている現在再生中のオーディオ トラックの設定/取得に使用されます。

例:

```
var ns:NetStream = new NetStream(nc);
var audioTracks;

ns.onMetaData = function(info:Object)
{
```

```
audioTracks = info.audioTracks;
}

if (audioTracks != undefined && audioTracks.length > 0)
ns.audioTrack = audioTracks[0].trackIndex;
```

subtitleTrack プロパティ

```
subtitleTrack:Number  [read-write]
```

Scaleform のバージョン: 3.0.63 以降

このプロパティは現在のサブタイトル トラックの設定と取得を許可します。サブタイトルをオフにする場合は、このプロパティを 0 に設定します。

onSubtitle イベント ハンドラ

```
onSubtitle = function(msg:String) {}
```

Scaleform のバージョン: 3.0.63 以降

このイベント ハンドラは、サブタイトル メッセージが表示できるようになると呼び出されます。

例:

```
var ns:NetStream = new NetStream(nc);
var subtitlesNumber = 0;

ns.onMetaData = function(info:Object)
{
subtitlesNumber = info. subtitleTracksNumber;
}

ns.onSubtitle = function(msg:String)
{
      sbTextField.text = msg;
}
if (subtitlesNumber > 0)
      ns.subtitleTrack = 1;
```

setNumberOfFramePools 関数

```
public function setNumberOfFramePools(pools : Number) : Void
```

Scaleform のバージョン: 3.0.68 以降

この関数は、内部のビデオ バッファの数を設定します。ビデオ バッファは、ビデオ デコード出力を保存するために使われ、“frame pools”（フレーム プール）と呼ばれます。CPU でデコーディングの読み込みが不安定になったような場合に、フレーム プールを多数持っていれば再生をスムーズに行うことが可能になります。デフォルト値は、1 です。

このメソッドは、ビデオ再生が始まる前に（つまり、NetStream.play()call の前に）呼び出さなければなりません。

setReloadThresholdTime 関数

```
public function setReloadThresholdTime(reloadTime : Number) : Void
```

Scaleform バージョン:　3.0.68 以降

この関数は、再読込みのタイミングを秒で設定します。Scaleform ビデオ ライブラリは、入力バッファのデータサイズが再読込みの閾値より小さくなったときに、次のファイルのリード リクエストを送出します。この閾値時間は、ビデオファイルのビットレートと再読込の時間設定によって自動的に決定されます。この再読込み時間のデフォルト値は、0.8（秒）です。

ビデオ再生中にゲームのデータを読み込んでいる場合には、このメソッドを使ってシーク リクエストの回数を減らすことができます。例をあげると、NetStream バッファ時間（NetStream.setBufferTime()メソッド）を 2 秒として、この閾値時間を 1 秒と設定しているとき、ゲーム データは続けて 1 秒間読み込むことができます。しかし、ゲーム データの読み込みは、再読込みタイミングまでに/の前に完了していなければなりません。ゲーム データが大きいような場合には、チャンクに分けて読み込むようにした方がよいでしょう。もし、ゲーム データの読み込みが、再読込みのタイミング以前に完了していないと、ビデオ データ バッファが空になっているので、ムービーの再生はカクカクしてしまいます。

このメソッドは、ビデオ再生が始まる前（NetStream.play()呼び出しの前）に、呼び出さなければなりません。

6 Selection クラス エクステンション

Selection クラスは、他にも機能はありますが、テキスト フィールド、ムービー クリップ、ボタンの中で入力フォーカスを管理することができます。

Scaleform は Selection クラスのこのフォーカス機能を拡張します。ActionScript 1.0 では、`Selection.captureFocus()`のように、Selection エクステンション関数を直接参照することができます。ActionScript 2.0 はそれほど寛容ではないので、このような呼び出しには不満を表します。ActionScript 2.0 コンパイラを抑制するために、以下の別のアクセス シンタックスを使用する必要があります: `Selection["captureFocus"]()`

alwaysEnableArrowKeys 静的プロパティ

```
alwaysEnableArrowKeys : Boolean
```

Scaleform のバージョン: 1.2.34 以降

この静的プロパティを使うと、"_focusrect" プロパティが false に設定されていても、矢印キーでフォーカスを変更することができます (フォーカスを取得したときに適用されます)。デフォルトでは、"_focusrect = false;" で黄色のフォーカス矩形が無効になっている場合、Flash では矢印キーを使ってフォーカスを変更することはできません。この動作を変更するには、この `alwaysEnableArrowKeys` プロパティを "true" に設定します。

例:

```
Selection["alwaysEnableArrowKeys"] = true;
```

alwaysEnableKeyboardPress 静的プロパティ

```
alwaysEnableKeyboardPress : Boolean
```

Scaleform のバージョン: 3.0.63 以降

この静的プロパティは、"_focusrect" プロパティが false に設定されていても、スペース キーや Enter キーが押下されると、onPress / onRelease イベントを発生させます (フォーカスを取得したときに適用されます)。デフォルトでは、"`_focusrect = false;`″ で黄色のフォーカス矩形が無効になっている場合、Flash でキーボードのキーを使用することはできません。この動作を変更するには、この `alwaysEnableKeyboardPress` プロパティを "true" に設定します。

例:

```
Selection["alwaysEnableKeyboardPress"] = true;
```

captureFocus() 静的メソッド

```
public function captureFocus([doCapture:Boolean, controllerIdx:Number]) : void
```

Scaleform のバージョン: 1.2.34 以降

この静的メソッドを使って、キーボード フォーカスをプログラム的に取得／解放します。

doCapture パラメータが true に設定された (または一切パラメータのない) captureFocus は、キーボード フォーカスを取得して、矢印キーにフォーカスの変更を許可します。これは、Tab キーを最初に押したときと似ています。

doCapture パラメータが false に設定された captureFocus は、キーボード フォーカスを解放して、フォーカス矩形が有効なときのマウスのように動作します。

パラメータ

`doCapture:Boolean` – オプションで、キーボード フォーカスを取得するか、解放するかを示します。このパラメータが省略された場合、captureFocus は、このパラメータが true に設定されているときのように動作します。

`controllerIdx:Number` – オプションで、キャプチャ動作に使われるキーボード／コントローラを示します。デフォルトではコントローラ 0 を使用します。

例:

```
Selection["captureFocus"](); // "true"を渡すのと同じです
Selection["captureFocus"](false);
```

disableFocusAutoRelease 静的プロパティ

```
disableFocusAutoRelease : Boolean
```

Scaleform のバージョン: 1.2.34 以降

この静的プロパティを使って、マウスの移動でキーボード フォーカスを解放する (標準の Flash の動作) か否かをコントロールします。デフォルトでは、フォーカスが取得され、黄色の矩形が表示されている場合 (矢印キーでフォーカスを変更できます)、マウスの移動でフォーカスが解放されます。この動作を回避するには、この静的プロパティを true に設定します。

例:

```
Selection["disableFocusAutoRelease"] = true;
```

disableFocusKeys 静的プロパティ

```
disableFocusKeys : Boolean
```

Scaleform のバージョン: 2.2.58 以降

この静的プロパティは、すべてのフォーカス キー (Tab、Shift-Tab、矢印キー) の処理を無効にするので、ユーザーは自分自身のフォーカス キー管理を実装することができます。

例:

```
Selection["disableFocusKeys"] = true;
```

関連項目:

```
moveFocus()
```

disableFocusRolloverEvent 静的プロパティ

```
disableFocusRolloverEvent : Boolean
```

Scaleform のバージョン: 2.0.37 以降

この静的プロパティは、フォーカスがキーによって変更される場合、rollover/out イベントの発生を無効にするために使用します。デフォルトでは、フォーカスが矢印キーの押下で変更される場合、rollover/rollout イベントが発生します。この動作を回避するには、この静的プロパティを true に設定します。

例:

```
Selection["disableFocusRolloverEvent "] = true;
```

modalClip 静的プロパティ

```
modalClip : MovieClip
```

Scaleform のバージョン: 2.2.58 以降

この静的プロパティは、指定されたムービー クリップを、フォーカス管理の点から「モーダル」クリップとして設定します。つまり、フォーカス キー (Tab、Shift-Tab、矢印キー) は、指定されたムービー クリップの内部でのみ、つまり、そのムービー クリップの「Tab キーの使用可能な」子クリップ間に限ってフォーカス矩形を移動するということです。

例:

```
Selection["modalClip"] = _root.mc;
...
Selection["modalClip"] = undefined;
```

関連項目:

```
disableFocusKeys
moveFocus()
```

moveFocus() 静的メソッド

```
public function moveFocus(keyToSimmulate : String [, startFromMovie:Object,
                includeFocusEnabledChars : Boolean = False, controllerIdx :
                Number]) : Object
```

Scaleform のバージョン: 2.2.58 以降

この静的メソッドは、フォーカス キー (Tab、Shift-Tab、矢印キー) のいずれかのキー押下をシミュレートして、フォーカス矩形を移動する場合に使用します。このメソッドは、`disableFocusKeys` プロパティと `modalClip` プロパティを併用して、カスタムのフォーカス管理の実装に使用できる場合があります。

パラメータ

`keyToSimmulate:String` – “up”、“down”、“left”、“right”、“tab”、“shifttab” などシミュレートするキーの名前

`startFromMovie:Object` – 文字を指定するオプションのパラメータです。moveFocus は、現在フォーカスが適用されている文字の代わりに、開始点としてこれを使用します。このプロパティは null、または未定義になっている場合があります。つまり、現在フォーカスが適用されている文字は、開始点として使用されるということです。これは 3 つ目のオプションのパラメータを指定する場合に便利なときがあります。

`includeFocusEnabledChars:Boolean` – moveFocus を focusEnabled プロパティだけが設定されている文字と、tabEnabled / tabIndex プロパティが設定されている文字で許可するオプションのフラグです。このフラグが指定されていない、または false に設定されている場合は、tabEnabled / tabIndex プロパティが設定されている文字だけが、フォーカスの移動の対象となります。

`controllerIdx:Number` – オプションで、キャプチャ動作に使われるキーボード／コントローラを示します。デフォルトではコントローラ 0 を使用します。

例:

```
Selection["moveFocus"](“up”);
Selection["moveFocus"](“tab”, _root.mc);
Selection["moveFocus"](“tab”, _root.mc, true);
Selection["moveFocus"](“tab”, null, true);
```

戻り値

新たにフォーカスが適用された文字、またはその文字が見つからない場合は、undefined を返します。

関連項目:

`disableFocusKeys`
`modalClip`

findFocus()スタティックメソッド

```
public function findFocus(keyToSimmulate : String [, parentMovie:Object, loop :
                Boolean, startFromMovie:Object, includeFocusEnabledChars : Boolean,
                controllerIndex : Number]) : Object
```

Scaleform のバージョン: 3.3.84

TAB、Shift＋TAB、またはアロー（カーソル）のいずれかのキーを押下するシミュレーションを行って、次のフォーカス・アイテムを探すスタティックメソッドです。カスタムのフォーカス・マネージメントは、このメソッドと `disableFocusKeys` および `setModalClip/getModalClip` エクステンションとを組み合わせて行います。

パラメータ

`keyToSimulate:String` – シミュレーションするキーの名称：up”、 “down”、“left”、“right”、“tab”、“shifttab

`parentMovie` - モーダル・クリップとして使用するムービークリップです。フォーカスアイテム検索は、このクリップのチャイルドを対象に実行されます。NULL でもかまいません。

`loop` - フォーカスをループする Boolean フラグです。たとえば、現在フォーカスが当たっているアイテムが下にあり、キーが「down」のとき、findFocus は「null」（このフラグが「false」のとき）または最上位のフォーカス可能アイテム（このフラグが「true」のとき）を返します。

`startFromMovie:Object` – `findFocus` がスタートポイントとして使用するキャラクタを、現在フォーカスが当たっているキャラクタの代わりに指定する、オプション・パラメータです。このプロパティは NULL または未定義でも構いません。その場合は現在フォーカスが当たっているキャラクタがスタートポイントとして使われます。

`includeFocusEnabledChars:Boolean` – `focusEnabled` プロパティのみが設定されているキャラクタと `tabEnabled/tabIndex` プロパティが設定されているキャラクタに対して、moveFocus を許可するオプションフラグです。このフラグが設定されていない場合、または false に設定されている場合は、`tabEnabled/tabIndex` プロパティが設定されているキャラクタのみがフォーカス移動の対象になります。

`controllerIndex` - フォーカスを操作するコントローラのインデックスで、ゼロから始まります。このインデックスとフォーカスグループとを組み合わせて複数コントローラのフォーカスをサポートします。

例

```
var a = Selection["findFocus"]("up");
```

戻り値

フォーカス対象の次のキャラクタ、またはキャラクタが見つからない場合は null。

参照

```
disableFocusKeys
setModalClip
getModalClip
```

setModalClip()スタティックメソッド

```
public function setModalClip(modalClip : Object, controllerIndex : Number)
```

Scaleform のバージョン: 3.3.84

指定したムービークリップをフォーカス・マネージメントの「モーダル」クリップとして設定するスタティックメソッドです。TAB、Shift+TAB、アローキーを使って、「タブ可能」なチャイルドにまたがって、指定したムービークリップ内でのみフォーカスを移動できることを意味します。

パラメータ

`modalClip`	– モーダル・クリップ。
`controllerIndex`	- ゼロから始まるコントローラのインデックス。

getModalClip()スタティックメソッド

```
public function getModalClip(controllerIndex : Number) : Object
```

Scaleform のバージョン: 3.3.84

指定したコントローラのモーダル・クリップを返すスタティックメソッドです。

パラメータ

`controllerIndex`	- ゼロから始まるコントローラのインデックス。

戻り値

モーダル・クリップ、または見つからない場合は不定。

setControllerFocusGroup()スタティックメソッド

```
public function setControllerFocusGroup (controllerIndex : Number, focusGroupIdx :
                                                                    Number) : Boolean
```

Scaleform のバージョン: 3.3.84

フォーカスグループと `controllerIndex` で表されるコントローラとを関連付けるスタティックメソッドです。デフォルトで、すべてのコントローラはフォーカスグループ `0` に関連付けられており、同じフォーカスを使用することを意味します。ただし、各コントローラにそれぞれ個別のフォーカスを割り当てることが可能です。たとえば、2 個のコントローラに別々のフォーカス（分割画面を使った場

合）を持たせたい場合、setControllerFocusGroup (1,1)をコールすると、コントローラ１に異なるフォーカスグループが生成されます。setControllerFocusGroup (1,0)をコールすると、コントローラ 0 とコントローラ１は再び同じフォーカスを共有するようになります。

パラメータ

controllerIndex – ゼロから始まるコントローラのインデックス。
focusGroupIdx – ゼロから始まるフォーカスグループ。

例

```
Selection["setControllerFocusGroup"](0,0);
Selection["setControllerFocusGroup"](1,1);
Selection["setControllerFocusGroup"](2,1);
```

戻り値

正常終了の場合 true を返す。

getControllerFocusGroup()スタティックメソッド

```
public function getControllerFocusGroup (controllerIndex : Number) : Number
```

Scaleform のバージョン: 3.3.84

指定したコントローラに関連付けられているフォーカスグループを返すスタティックメソッドです。

パラメータ

controllerIndex – ゼロから始まる物理コントローラのインデックス。

戻り値

ゼロから始まるフォーカスグループ

getFocusArray()スタティックメソッド

```
public function getFocusArray(mc : Object) : Array
```

Scaleform のバージョン: 3.3.84

指定したムービークリップ／ボタン／テキストフィールドを現時点でフォーカスしているコントローラ・インデックスのアレイを返すスタティックメソッドです。

パラメータ

mc - ムービークリップ、ボタン、またはテキストフィールド。

戻り値

ゼロから始まるインデックスのアレイ（個数）。

getFocusBitmask()スタティックメソッド

```
public function getFocusBitmask(mc : Object) : Number
```

Scaleform のバージョン: 3.3.84

指定したムービークリップ／ボタン／テキストフィールドを現時点でフォーカスしているコントローラをそれぞれのビットで表したビットマスクとして返すスタティックメソッドです。

パラメータ

`mc` - ムービークリップ、ボタン、またはテキストフィールド。

戻り値

コントローラのビットマスク。

getControllerMaskByFocusGroup()スタティックメソッド

```
public function getControllerMaskByFocusGroup (focusGroupIdx : Number) : Number
```

Scaleform のバージョン: 3.3.84

指定したフォーカスグループに関連付けられているコントローラをそれぞれのビットで表したビットマスクとして返すスタティックメソッドです。

パラメータ

`focusGroupIdx` - フォーカスグループのインデックス。

戻り値

コントローラのビットマスク。

numFocusGroups プロパティ

```
numFocusGroups : Number
```

Scaleform のバージョン: 3.3.84

`setControllerFocusGroup` 関数のコールによってセットアップされたフォーカスグループの個数を返します。たとえば、フォーカスグループ `0` と `3` がアクティブの場合、`numFocusGroups` は `2` を返します。

7 TextField クラス エクステンション

Scaleform は多くのエクステンションを TextField クラスに導入して、Flash 8 に内蔵されている TextField クラスよりも、さらに柔軟で機能的にしています。提供されたエクステンションの中には、Flash 9 の TextField と同じ、または類似した動作を再現するものもあります。追加されたエクステンションは、IME、位置合わせ、テキスト サイズとテキストの表示、テキスト フィルタ効果 (ドロップシャドウ、ぼかしなど)、埋め込まれたイメージ、選択やクリップボードの操作などの管理を向上する機能を提供します。

7.1 一般的な機能

この章では、汎用エクステンション プロパティとメソッドについて説明します。

autoFit プロパティ

```
autoFit:Boolean  [read-write]
```

Scaleform のバージョン: 2.0.39 以降

フォントの自動ヒンティングのオン/オフを設定します。

デフォルト値は `false` です。

appendText () メソッド

```
public function appendText(newText:String):void
```

Scaleform のバージョン: 2.0.37 以降

newText パラメータで指定したストリングを、テキスト フィールドのテキストの末尾に追加します。このメソッドは、テキスト プロパティに付加を指定する (+=) (例、`my_txt.text += appendingText`) よりもはるかに効率的です。特に大量のテキストを含むテキスト フィールドには便利です。

ご注意： このメソッドは、テキストフィールドにスタイルシートが適用されている場合には無効です。

パラメータ

`newText:String` – 既存のテキストに追加されるストリングです。

関連項目:

`appendHtml()`

appendHtml () メソッド

```
public function appendHtml(newHtml:String):void
```

Scaleform のバージョン: 2.0.37 以降

newHtml パラメータで指定した HTML を、テキスト フィールドのテキストの末尾に追加します。このメソッドは、htmlText プロパティに付加 (+=) を指定する (`txt.htmlText += moreHtml` など) よりも効率的です。htmlText プロパティの標準の += は、HTML ストリングを生成して、新規の HTML の部分を追加し、最初から HTML 全体を解析します。この関数は HTML の増分解析を行います。つまり、newHtml ストリング パラメータの HTML のみを解析します (newHtml パラメータの HTML が適格であるべきなのは、このためです。このことは+=演算子には必要ありません)。大量の内容を含んでいるテキスト フィールドには、これは特に重要です。

ご注意：　このメソッドは、テキストフィールドにスタイルシートが適用されている場合には、無効です。

パラメータ

`newHtml:String` – 既存のテキストに追加される HTML 付きのストリングです。

関連項目:

`appendText()`

caretIndex プロパティ

```
caretIndex:Number  [read-only]
```

Scaleform のバージョン: 2.0.37 以降

このプロパティは、挿入ポイント (キャレット、またはカーソル) の位置のインデックスを表します。テキスト フィールドにフォーカスが適用されておらず、キャレットの位置が (カーソルの点滅で) 表示されていない場合、テキスト フィールドがフォーカスを失う直前のキャレットの位置を表します。テキスト フィールドにまったくフォーカスがなかった場合は 0 です。このプロパティには、Selection.getCaretIndex() メソッドが返す値と同じものが含まれます。ただし、caretIndex はテキスト フィールドにフォーカスがない場合でも、アクセス可能です。一方で、Selection.getCaretIndex() はこのような場合に -1 を返します。

キャレット インデックスはゼロベースです (したがって、最初の位置は 0 です)。

関連項目:

`Selection.getCaretIndex()`

getCharBoundaries () メソッド

```
public function getCharBoundaries(charIndex:Number): flash.geom.Rectangle
```

Scaleform のバージョン: 2.0.37 以降

このメソッドは、文字の境界ボックスである矩形を返します。このメソッドは、各グリフのアドバンス値を使って計算した矩形を返します。つまり、戻された矩形は正確ではないということです (下の図では、赤の矩形が 'a'、'w'、'k' の境界を表しています):

パラメータ

`charIndex:Number` – 文字のゼロベースのインデックス値 (たとえば、最初の位置は 0 で、次の位置は 1 と続き、以下同様です)。

戻り値

flash.geom.Rectangle – 文字の境界ボックスを定義する x と y の最小値と最大値を持つ矩形です。座標はテキスト フィールドの座標空間にあります。つまり (0,0) の地点はテキスト フィールドの左上隅に対応します。

関連項目:

`getExactCharBoundaries()`

getExactCharBoundaries () メソッド

```
public function getExactCharBoundaries(charIndex:Number): flash.geom.Rectangle
```

Scaleform のバージョン: 2.0.37 以降

このメソッドは、文字の正確な境界ボックスである矩形を返します。このメソッドは、各グリフの実際の幅を使って計算された (アドバンス値を使用する `getCharBoundaries()` とは異なります) 矩形を返します。この矩形の高さは、その行の高さです。下の図では、赤の矩形が、'a'、'w'、'k' の正確な境界を表しています:

パラメータ

charIndex:Number – 文字のゼロベースのインデックス値 (たとえば、最初の位置は 0 で、次の位置は 1 と続き、以下同様です)。

戻り値

flash.geom.Rectangle – 文字の正確な境界ボックスを定義する x と y の最小値と最大値を持つ矩形です。座標はテキスト フィールドの座標空間にあります。つまり (0,0) の地点はテキスト フィールドの左上隅に対応します。

関連項目:

getCharBoundaries()

getCharIndexAtPoint () メソッド

```
public function getCharIndexAtPoint(x:Number, y:Number): Number
```

Scaleform のバージョン: 2.0.37 以降

このメソッドは、x と y のパラメータで指定された地点にある文字の、ゼロベースのインデックス値を返します。

パラメータ

x:Number – その文字の x 座標
y:Number – その文字の y 座標

戻り値

Number – 文字のゼロベースのインデックス値 (たとえば、最初の位置は 0 で、次の位置は 1 と続き、以下同様です)。その位置に文字がない場合は -1 を返します。

getFirstCharInParagraph () メソッド

```
public function getFirstCharInParagraph(charIndex:Number): Number
```

Scaleform のバージョン: 2.0.37 以降

このメソッドは、charIndex インデックスの文字を含む段落の、最初の文字のインデックスを返します。

パラメータ

charIndex:Number – 文字のゼロベースのインデックス値 (たとえば、最初の位置は 0 で、次の位置は 1 と続き、以下同様です)。

戻り値

Number – 同じ段落の最初の文字のゼロベースのインデックス値です。

getLineIndexAtPoint () メソッド

```
public function getLineIndexAtPoint(x:Number, y:Number): Number
```

Scaleform のバージョン: 2.0.37 以降

このメソッドは、x と y のパラメータで指定された地点にある行の、ゼロベースのインデックス値を返します。

パラメータ

`x:Number` – その行の x 座標
`y:Number` – その行の y 座標

戻り値

Number – 行のゼロベースのインデックス値 (たとえば、最初の行は 0 で、次の行は 1 と続き、以下同様です)。その位置に行がない場合は -1 を戻します。

getLineLength() メソッド

```
public function getLineLength(lineIndex:Number): Number
```

Scaleform のバージョン: 2.0.37 以降

このメソッドは、特定のテキスト行の文字数を返します。

パラメータ

`lineIndex:Number` – 行のゼロベースのインデックス値 (たとえば、最初の行は 0 で、次の行は 1 と続き、以下同様です)。

戻り値

Number – その行の文字数

getLineMetrics() メソッド

```
public function getLineMetrics(lineIndex:Number): Object
```

Scaleform のバージョン: 2.0.43 以降

このメソッドは、指定されたテキスト行についてのメトリック情報を返します。返されるオブジェクトは以下のメンバーを含んでいます:

`ascent : Number`

テキストのアセント値は、ベースラインから、その行の上辺までのピクセル数による長さです。

`descent : Number`

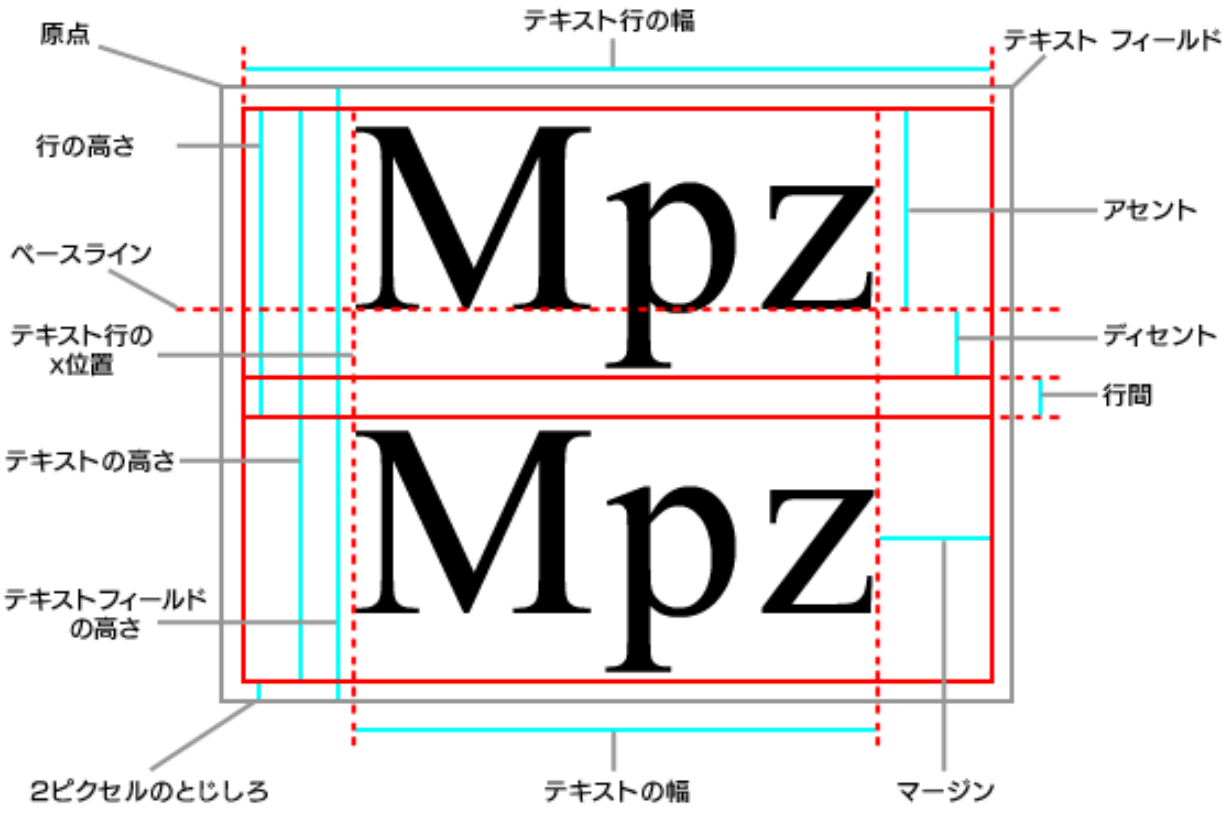
テキストのディセント値は、ベースラインから、その行の下辺までのピクセル数による長さです。

`height : Number`
選択した行の、ピクセル数によるテキストの高さの値 (テキスト全体である必要はありません)。

`leading : Number`
行間値は、テキスト行の間の垂直の距離の測定値です。

`width : Number`
選択した行の、ピクセル数によるテキストの幅の値 (テキスト全体である必要はありません)。

`x : Number`
x 値は、ピクセル数による最初の文字の左の位置です。



パラメータ
`lineIndex:Number` – 行のゼロベースのインデックス値 (たとえば、最初の行は 0 で、次の行は 1 と続き、以下同様です)。

戻り値
Number – メトリック情報を持つオブジェクト

getLineOffset () メソッド

```
public function getLineOffset(lineIndex:Number): Number
```

Scaleform のバージョン: 2.0.37 以降

このメソッドは lineIndex パラメータが指定した行の、最初の文字の文字インデックスを返します。

パラメータ

lineIndex:Number – 行のゼロベースのインデックス値 (たとえば、最初の行は 0 で、次の行は 1 と続き、以下同様です)。

戻り値

Number – 行の最初の文字の、ゼロベースのインデックス値

getLineText () メソッド

```
public function getLineText(lineIndex:Number): String
```

Scaleform のバージョン: 2.0.43 以降

このメソッドは、lineIndex パラメータで指定された行のテキストを返します。

パラメータ

lineIndex:Number – 行のゼロベースのインデックス値 (たとえば、最初の行は 0 で、次の行は 1 と続き、以下同様です)。

戻り値

String – 指定された行に含まれるテキスト ストリング

hitTestDisable プロパティ

```
hitTestDisable:Boolean  [read-write]
```

Scaleform のバージョン: 1.2.32 以降

true に設定されると、MovieClip.hitTest 関数は、ヒット テストの検出中はこのテキスト フィールドを無視します。さらに、その他のマウス イベントもすべて、そのテキスト フィールドには伝播しません。

デフォルト値は false です。

noTranslate プロパティ

```
noTranslate:Boolean  [read-write]
```

Scaleform のバージョン: 1.2.34 以降

true に設定すると、`GFxTranslator::Translate` コールバックがそのテキスト フィールドに対して呼び出されないようにします。

デフォルト値は `false` です。

numLines プロパティ

```
numLines:Number  [read-only]
```

Scaleform のバージョン: 2.0.43 以降

このプロパティは、複数行のテキスト フィールド内のテキスト行の数を表します。`wordWrap` プロパティが true の場合、テキストを折り返すと行数が増えます。

topmostLevel プロパティ

```
topmostLevel:Boolean  [read-write]
```

Scaleform のバージョン: 2.1.50 以降

このプロパティが true に設定された場合、この文字はその深度に関係なく、他のすべての文字の上に表示されます。このプロパティはカスタムのマウス カーソルと、すべてのレベルのオブジェクト上に描画が必要なポップアップの実装に便利です。

デフォルト値は `false` です。

"topmostLevel”として、いくつかの文字にマークしたい場合には、描画順は以下のとおりになります：

- Scaleform3.0.71 までは、”topmostLevel”とマークした文字の描画順は、このプロパティを true と設定した順に依存します（ということは、最初に topmost とマークされた文字が最初に描画されます。）；
- Scaleform3.0.72 以降では、文字に topmost マーキングを行わなかったと同じ順に描画されます。つまり、オブジェクト A がオブジェクト B の下に描画されていたとすれば、topmost マーキングを行った後でも、オブジェクト A はオブジェクト B の下に描画されます。"topmostLevel"プロパティを true と設定した順序には関わりません。

注意： "topmostLevel"としてマークされると、マークされた文字に対して、SwapDepth ActionScript 関数は、有効性を持たなくなります。

関連項目:

```
MovieClip.topmostLevel
Button.topmostLevel
```

focusGroupMask プロパティ

```
focusGroupMask : Number
```

Scaleform のバージョン: 3.3.84

ビットマスクをステージ・キャラクタと**すべての**チャイルドに設定するプロパティです。このビットマスクによって、フォーカスグループのオーナーシップがキャラクタに割り当てられます。すなわち、ビットマスクで示されるコントローラのみが、キャラクタにフォーカスを移動したりキャラクタ内部でフォーカスを移動できることを意味します。フォーカスグループは `setControllerFocusGroup` エクステンションメソッドを使ってコントローラに対応付けることができます。

たとえば、「button1」がコントローラ 0 からのみフォーカス可能で、「movieclip2」がコントローラ 0 とコントローラ 1 からフォーカス可能と仮定します。このような挙動を実現するには、フォーカスグループとコントローラとを以下のように対応付けます。

```
Selection.setControllerFocusGroup(0, 0);
Selection.setControllerFocusGroup(1, 1);

button1.focusGroupMask = 0x1; // bit 0 – focus group 0
movieclip2.focusGroupMask = 0x1 | 0x2 // bits 0 and 1 – focus groups 0 and 1
```

「`focusGroupMask`」ビットマスクは親ムービークリップに設定される場合があります。その場合、マスク値はすべてのチャイルドに伝搬します。

7.2 選択操作とクリップボード操作

この章では選択操作とクリップボード操作のエクステンションについて説明します。

alwaysShowSelection プロパティ

```
alwaysShowSelection:Boolean  [read-write]
```

Scaleform のバージョン: 2.1.45 以降

デフォルトでは、テキスト フィールドにフォーカスがない場合、Scaleform は選択範囲をハイライト表示しません。このプロパティが true に設定されていて、テキスト フィールドにフォーカスがない場合、Scaleform はテキスト フィールドの選択範囲をグレーでハイライト表示します。アクティブな選択範囲とアクティブではない選択範囲の色は、オーバーライドできる場合があります (「関連項目」を参照してください)。

デフォルト値は `false` です。

関連項目:

selectionBkgColor
selectionTextColor
inactiveSelectionBkgColor
inactiveSelectionTextColor

copyToClipboard () メソッド

```
public function copyToClipboard([richClipboard:Boolean], [startIndex:Number],
[endIndex:Number]):void
```

Scaleform のバージョン: 2.1.45 以降

このメソッドはテキストをクリップボードにコピーします。すべてのパラメータはオプションです。

パラメータ

`richClipboard:Boolean` – `true` の場合、テキストはスタイルと共にクリップボードにコピーされます。デフォルト値は `useRichTextClipboard` プロパティと同じです。

`startIndex:Number` – コピーされるテキスト セグメントの開始インデックスです。デフォルト値は `selectionBeginIndex` と同じです。

`endIndex:Number` – コピーされるテキスト セグメントの終了インデックスです。デフォルト値は `selectionEndIndex` と同じです。

関連項目:

useRichTextClipboard
selectionBeginIndex
selectionEndIndex
cutToClipboard()
pasteFromClipboard()

cutToClipboard () メソッド

```
public function cutToClipboard([richClipboard:Boolean], [startIndex:Number],
[endIndex:Number]):void
```

Scaleform のバージョン: 2.1.45 以降

このメソッドはテキストをクリップボードにコピーして、テキスト フィールドからそのテキストを削除します。すべてのパラメータはオプションです。

パラメータ

`richClipboard:Boolean` – `true` の場合、テキストはスタイルと共にクリップボードにコピーされます。デフォルト値は `useRichTextClipboard` プロパティと同じです。

startIndex:Number – コピーされるテキスト セグメントの開始インデックスです。デフォルト値は `selectionBeginIndex` と同じです。

`endIndex:Number` – コピーされるテキスト セグメントの終了インデックスです。デフォルト値は `selectionEndIndex` と同じです。

関連項目:

`useRichTextClipboard`
`selectionBeginIndex`
`selectionEndIndex`
`copyToClipboard()`
`pasteFromClipboard()`

inactiveSelectionBkgColor プロパティ

```
inactiveSelectionBkgColor:Number  [read-write]
```

Scaleform のバージョン: 2.1.45 以降

アクティブではない選択範囲の背景色を指定します。これは、alwaysShowSelection プロパティが true に設定されている場合に限って、有効です。この背景色は、0xAARRGGBB というフォーマットで指定されます。この場合、AA はアルファチャネル コンポーネントの 16 進数表現[0...255]、RR は赤のコンポーネントの 16 進数表現 [0...255]、BB は青のコンポーネントの 16 進数表現 [0...255]、GG は緑のコンポーネントの 16 進数表現 [0...255] です。

注意事項として、アルファチャネルは、“0”には設定しないようにしてください。そうすると何も描画されなくなってしまいます（というのは、“0”に設定されたアルファは全くの透明であるからです）。こうしてここでの色は Flash の背景色に使われる標準色とはやや異なることになります。例えば、このプロパティを真っ赤な色に設定するには、0xFFFF0000 (アルファと赤を 0xFF(255)に設定する) という値を使わなければなりません。これに対し標準の“backgroundColor ”プロパティでは、0xFF0000 という値を使えば済むわけです。

関連項目:

`alwaysShowSelection`
`inactiveSelectionTextColor`
`selectionBkgColor`
`selectionTextColor`

inactiveSelectionTextColor プロパティ

```
inactiveSelectionTextColor:Number  [read-write]
```

Scaleform のバージョン: 2.1.45 以降

アクティブではない選択範囲のテキストの色を指定します。これは、alwaysShowSelection プロパティが true に設定されている場合に限って、有効です。この色は、0xAARRGGBB というフォーマットで指定されます。この場合、AA はアルファチャネル コンポーネントの 16 進数表現[0..255]、RR は

赤のコンポーネントの 16 進数表現 [0..255]、BB は青のコンポーネントの 16 進数表現 [0..255]、GG は緑のコンポーネントの 16 進数表現 [0..255] です。

注意事項として、アルファチャネルは“0”には設定しないようにしてください。そうすると何も描画されなくなってしまいます（というのは、”0”に設定されたアルファは全くの透明であるからです）。こうしてここでの色は Flash のテキスト色に使われる標準色とはやや異なることになります。例えば、このプロパティを真っ赤な色に設定するには、0xFFFF0000　(アルファと赤を 0xFF(255)に設定する)という値を使わなければなりません。これに対し標準の“textColor”プロパティでは、0xFF0000 という値を使えば済むわけです。

関連項目:

```
alwaysShowSelection
inactiveSelectionBkgColor
selectionBkgColor
selectionTextColor
```

noAutoSelection プロパティ

```
noAutoSelection:Boolean  [read-write]
```

Scaleform のバージョン: 2.1.45 以降

`true` に設定された場合、フォーカスがテキスト フィールドに移ったときの自動選択を防止します。

デフォルト値は `false` です。

pasteFromClipboard () メソッド

```
public function pasteFromClipboard([richClipboard:Boolean], [startIndex:Number],
[endIndex:Number]):void
```

Scaleform のバージョン: 2.1.45 以降

このメソッドはクリップボードからテキストをペーストします。すべてのパラメータはオプションです。

パラメータ

`richClipboard:Boolean` – `true` の場合、テキストはスタイルと共にクリップボードからペーストされます。デフォルト値は `useRichTextClipboard` プロパティと同じです。

`startIndex:Number` – クリップボードのテキストに置き換えられるセグメントの開始インデックスです。デフォルト値は `selectionBeginIndex` と同じです。

`endIndex:Number` – クリップボードのテキストに置き換えられるセグメントの終了インデックスです。デフォルト値は `selectionEndIndex` と同じです。

関連項目:

```
useRichTextClipboard
selectionBeginIndex
selectionEndIndex
copyToClipboard()
cutToClipboard()
```

selectionBeginIndex プロパティ

```
selectionBeginIndex:Number  [read-only]
```

Scaleform のバージョン: 2.1.45 以降

このプロパティは現在の選択範囲の、最初の文字のゼロベースの文字インデックス値を表します。テキストが選択されていない場合、このプロパティは `caretIndex` の値になります。このプロパティには、`Selection.getBeginIndex()` メソッドが返す値と同じものが含まれます。
`selectionBeginIndex` はテキスト フィールドにフォーカスがない場合でもアクセス可能ですが、`Selection.getBeginIndex()` はこのような場合に -1 を返します。

関連項目:

```
caretIndex
selectionEndIndex
Selection.getBeginIndex()
```

selectionEndIndex プロパティ

```
selectionEndIndex:Number  [read-only]
```

Scaleform のバージョン: 2.1.45 以降

このプロパティは現在の選択範囲の、最後の文字のゼロベースの文字インデックス値を表します。テキストが選択されていない場合、このプロパティは `caretIndex` の値になります。このプロパティには、`Selection.getEndIndex()` メソッドが返す値と同じものが含まれます。ただし、
`selectionEndIndex` はテキスト フィールドにフォーカスがない場合でもアクセス可能です。一方で、`Selection.getEndIndex()` はこのような場合に -1 を返します。

関連項目:

```
caretIndex
selectionBeginIndex
Selection.getEndIndex()
```

selectionBkgColor プロパティ

```
selectionBkgColor:Number  [read-write]
```

Scaleform のバージョン: 2.1.45 以降

アクティブな選択範囲の背景色を指定します。この背景色は、0xAARRGGBB というフォーマットで指定されます。この場合、AA はアルファチャネル コンポーネントの 16 進数表現[0..255]、RR は赤のコンポーネントの 16 進数表現 [0..255]、BB は青のコンポーネントの 16 進数表現 [0..255]、GG は緑のコンポーネントの 16 進数表現 [0..255] です。

注意事項として、アルファチャネルは“0”には設定しないようにしてください。そうすると何も描画されなくなってしまいます（というのは、“0”に設定されたアルファは全くの透明であるからです）。こうしてここでの色は Flash の背景色に使われる標準色とはやや異なることになります。例えば、このプロパティを真っ赤な色に設定するには、0xFFFF0000　(アルファと赤を 0xFF(255)に設定する)という値を使わなければなりません。これに対し標準の“backgroundColor”プロパティでは、0xFF0000 という値を使えば済むわけです。

関連項目:

```
inactiveSelectionTextColor
inactiveSelectionBkgColor
selectionTextColor
```

selectionTextColor プロパティ

```
selectionTextColor:Number  [read-write]
```

Scaleform のバージョン: 2.1.45 以降

アクティブな選択範囲のテキストの色を指定します。この色は、0xAARRGGBB というフォーマットで指定されます。この場合、AA はアルファチャネル コンポーネントの 16 進数表現[0..255]、RR は赤のコンポーネントの 16 進数表現 [0..255]、BB は青のコンポーネントの 16 進数表現 [0..255]、GG は緑のコンポーネントの 16 進数表現 [0..255] です。

注意事項として、アルファチャネルは“0”には設定しないようにしてください。そうすると何も描画されなくなってしまいます（というのは、“0”に設定されたアルファは全くの透明であるからです）。こうしてここでの色は Flash のテキスト色に使われる標準色とはやや異なることになります。例えば、このプロパティを真っ赤な色に設定するには、0xFFFF0000　(アルファと赤を 0xFF(255)に設定する)という値を使わなければなりません。これに対し標準の“textColor”プロパティでは、0xFF0000 という値を使えば済むわけです。

関連項目:

```
inactiveSelectionBkgColor
inactiveSelectionBkgColor
selectionBkgColor
```

useRichTextClipboard プロパティ

```
useRichTextClipboard:Boolean  [read-write]
```

Scaleform のバージョン: 2.1.45 以降

テキストと共に、テキストのフォーマットをコピー＆ペーストするかどうかを指定します。true の場合、Scaleform は、テキスト フィールド間でコピー&ペーストをする場合に、フォーマット (行揃え、ボールド、イタリックなど) もコピー＆ペーストします。コピー＆ペースト手順のコピー元とコピー先のテキスト フィールドの両方で、`useRichTextClipboard` を `true` に設定する必要があります。

デフォルト値は false です。

7.3 テキスト サイズと行揃え

この章では、テキストのサイズと行揃えを制御するエクステンションについて説明します。標準の行揃えのほかに (左揃え、右揃え、中央揃え)、Scaleform は垂直方向の行揃えと、垂直方向の自動サイズ機能を提供します。

fontScaleFactor プロパティ

```
fontScaleFactor:Number  [read-write]
```

Scaleform のバージョン: 2.0.43 以降

このプロパティは、テキスト フィールドのすべてのフォントに対するフォントの倍率を指定します。この `fontScaleFactor` 値をフォント サイズに掛けて、すべてのフォント サイズを増減させる場合があります。

デフォルト値は 1.0 です。

textAutoSize プロパティ

```
textAutoSize:String  [read-write]
```

Scaleform のバージョン: 2.0.43 以降

textAutoSize は、テキストのフォント サイズの自動サイズ変更を有効にします。このプロパティの有効な値は、"none"、"shrink"、"fit"です。このモードがオンに設定されていて ("shrink" または "fit")、テキストがテキスト フィールド内に収まらない場合、テキストのサイズは、そのテキスト フィールドにテキスト全体が収まるように均等に縮小されるので、スクロールは必要ありません。テキスト サイズが小さくなりすぎると (約 5 pt など)、デフォルトのスクロール ロジックが適用され、これ以上フォント サイズが縮小されなくなります。

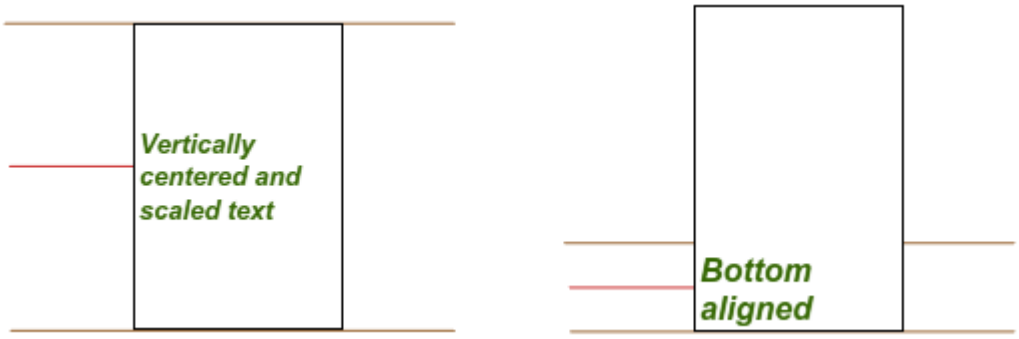
textAutoSize を "fit" モードに設定すると、そのテキスト フィールド全体がテキストで埋まるまで、フォント サイズを拡大することができます。"shrink" モードは、テキストがテキスト フィールドに収まらない場合に限って、フォント サイズを縮小できます。元のフォント サイズは拡大されません。デフォルト値は "none" です。

verticalAlign プロパティ

```
verticalAlign:String  [read-write]
```

Scaleform のバージョン: 2.0.43 以降

テキスト ボックス内のテキストの、垂直方向の行揃えを設定します。このプロパティの有効な値は、"none" ("top" も"none" と同じです)、"bottom"、"center" です。このプロパティが"center" に設定されている場合、テキストはテキスト ボックスの中央に揃えられます。"bottom" の場合は、テキストはテキスト ボックスの下辺に揃えられます (下の図を参照してください)。



デフォルト値は "none" です。

関連項目:

```
verticalAutoSize
TextField.align
```

verticalAutoSize プロパティ

`verticalAutoSize:String [read-write]`

Scaleform のバージョン: 2.0.43 以降

テキスト フィールドの垂直方向の自動サイズ変更と行揃えを制御します。有効な値は"none" (デフォルト)、"top"、 "bottom"、 "center" です。verticalAutoSize が"none" (デフォルト) に設定されている場合、サイズ変更は行われません。

`verticalAutoSize` が "top" に設定されている場合、テキスト フィールドは、`wordWrap` が `true` に設定されているかのように (つまり、テキスト フィールドの下部だけがサイズ変更され、右側は固定されたままです)、通常の `TextField.autoSize` と同様に動作します。

`verticalAutoSize` が `"center"` に設定されている場合、テキスト フィールドはサイズ変更されます。この場合、テキスト フィールドは、上下のマージンが両方とも均一にサイズ変更されます。アンカー ポイントは、元のテキスト フィールドの高さの中間点に位置します (次の図（1）を参照してください)。

`verticalAutoSize` が ″`bottom`″ に設定されている場合、テキスト フィールドは上のマージンに合わせてサイズ変更されます。アンカー ポイントは、元のテキスト フィールド ボックスの下部に位置します (次の図（2）を参照してください)。

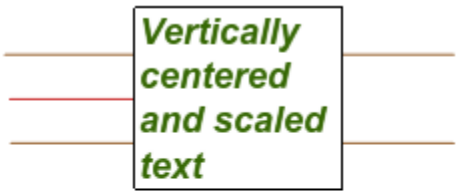


図 (1)

図 (2)

デフォルト値は "none" です。

関連項目:

`verticalAlign`
`TextField.autoSize`

7.4 HTML エクステンション

この章では、Scaleform の HTML に対するエクステンションについて説明しています。

<FONT>タグ、ALPHA 属性

ALPHA="#xx"

テキストのアルファ透明度を、00 から FF (0 から 255) の範囲で制御します。0 とは完全に透明であることを意味し、255 は完全に不透明であることを意味します。COLOR 属性と組み合わせて、この ALPHA 属性は HTML で半透明のテキストを指定することができます。

<IMG> タグ

現時点では、Scaleform の <IMG> タグは、割り当てられたリンケージ識別子と共に、ライブラリにインポートされたイメージだけを参照することができます。Scaleform は、外部ソースからインポートされるイメージはサポートしていません。外部ソースとは、例えば Web やファイルなどです (“http://”、”file://”など、その他のプロトコルはサポートされていません)。詳細については 5.6 章のイメージのインポートとリンケージ識別子の割り当て方法を参照してください。
以下の例は、"myImage" というインポート イメージを参照する <IMG> タグを使った HTML です:

```
t.htmlText = "<p align='right'>abra<img src='myImage' width='20' height='30' align='baseline' vspace='-10'>bed232</p>";
```

IMG タグの現在サポートされている属性:

`src` – ライブラリ内のイメージシンボルに対するリンク識別子を指定します。現時点でイメージのみサポートしています。このアトリビュートは必須であり、残りの各アトリビュートはオプションです。外部ファイル（JPEG、GIF、PNG、SWF ファイル）は現時点でサポートしていません。エンベデッドイメージリンク ID の代わりにユーザープロトコル「img://」が使用可能で、その場合、ユーザー定義の仮想メソッド Scaleform::ImageCreator:: LoadImage がコールされます。

`width` – 挿入されるイメージのピクセル数による幅です。

`height` – 挿入されるイメージのピクセル数による高さです。

`align` – テキスト フィールド内で、埋め込まれたイメージの水平方向の行揃えを指定します。現時点ではサポートされている値は `"baseline"` だけです。

`vspace` - `"baseline"` 行揃えに対し、ベースラインに相対的にイメージのオフセットをピクセル数で指定します。正数値はベースラインより上にイメージを配置し、負数値はイメージを下げます。

サポートされていない属性:

id – 埋め込まれたイメージ ファイル、SWF ファイル、またはムービー クリップを含むムービー クリップ インスタンス (Flash Player で作成したもの) に名前を指定します。

`align` – 「左」と「右」。テキスト フィールド内で、埋め込まれたイメージの水平方向の行揃えを指定します。

`hspace` - 「左」と「右」について、イメージの周囲の、テキストが表示されない水平方向のスペースの量を指定します。

イメージを別の SWF ファイルからインポートした場合、IMG タグで使用可能になるように、このイメージを明確にその Flash ファイルのどこかで使用する必要があります。イメージをインポートしても、一度も使用しない場合、Flash はそのイメージのリファレンスを、作成される SWF ファイルから完全に削除してしまいます。これを防ぐ一つの方法は、ムービー クリップを作成して、IMG タグに必要なインポートされたすべてのイメージを、その上にドラッグ&ドロップすることです。このムービー クリップをエクスポートすることを忘れないでください。これも明確に使用されず、エクスポートされない場合は、Flash は作成される SWF ファイルからそのムービー クリップを削除してしまいます。

7.5 シャドウ効果の制御

この章では、シャドウのさまざまな効果 (ぼかし、ノックアウトなど) を制御するエクステンションについて説明します。

blurX と blurY プロパティ

```
blurX:Number  [read-write]
blurY:Number  [read-write]
```

Scaleform のバージョン: 2.0.41 以降

テキストのぼかしの範囲 (半径) を取得、または設定します。有効な値は 0 から 15.9 (浮動小数点) までです。

デフォルト値は 0 です。

関連項目:
`blurStrength`

blurStrength プロパティ

```
blurStrength:Number  [read-write]
```

Scaleform のバージョン: 2.0.41 以降

テキストのぼかしの程度を取得、または設定します。有効な値は 0 から 15.9 (浮動小数点) までです。

デフォルト値は 0 です。

関連項目:
```
blurX
blurY
```

shadowAlpha プロパティ

```
shadowAlpha:Number  [read-write]
```

Scaleform のバージョン: 2.0.41 以降

シャドウ カラーのアルファ透明度値を制御します。有効な値は 0.00 から 1.00 です。たとえば、25 は 25%の透明度値を設定します。

関連項目:
```
shadowColor
```

shadowAngle プロパティ

```
shadowAngle:Number  [read-write]
```

Scaleform のバージョン: 2.0.41 以降

シャドウの角度を制御します。DropShadowFilter に似ています。有効な値は 0 から 360° (浮動小数点) までです。デフォルト値は 45 です。

関連項目:
```
DropShadowFilter
```

shadowBlurX プロパティ、shadowBlurY プロパティ

```
:Number  [read-write]
shadowBlurY:Number  [read-write]
```

Scaleform のバージョン: 2.0.41 以降

シャドウのぼかしの範囲 (半径) を制御します。有効な値は 0 から 15.9 (浮動小数点) までです。

デフォルト値は 0 です。

関連項目:

shadowStrength

shadowColor プロパティ

```
shadowColor:Number  [read-write]
```

Scaleform のバージョン: 1.1.31 以降

シャドウ カラーを指定します (詳細は shadowStyle を参照してください)。この色は、0xRRGGBB というフォーマットで指定されます。この場合 RR は赤のコンポーネントの 16 進数表現 [0..255]、BB は青のコンポーネントの 16 進数表現 [0..255]、GG は緑のコンポーネントの 16 進数表現 [0..255] です。

関連項目:
shadowStyle

shadowDistance プロパティ

```
shadowDistance:Number  [read-write]
```

Scaleform のバージョン: 2.0.41 以降

シャドウのピクセル数によるオフセットの距離を制御します。DropShadowFilter に似ています。

関連項目:
DropShadowFilter

shadowHideObject プロパティ

```
shadowHideObject:Boolean  [read-write]
```

Scaleform のバージョン: 2.0.41 以降

テキストが非表示であるかどうかを表します。true はテキスト自体が描画されず、シャドウだけが表示されることを表します。デフォルトは false (テキストを表示する) です。

shadowKnockOut プロパティ

```
shadowKnockOut:Boolean  [read-write]
```

Scaleform のバージョン: 2.0.41 以降

ノックアウト効果 (true) を制御します。この効果はオブジェクトの塗り潰しを効率的に透明にして、ドキュメントの背景色を表示します。デフォルトは false (ノックアウトなし) です。

shadowQuality プロパティ

```
shadowQuality:Number  [read-write]
```

Scaleform のバージョン: 2.0.41 以降

シャドウ フィルタ、またはグロー フィルタの質を制御します。Flash とは異なり、有効な値は 1 と 2 だけです。1 は低品質で、2 は高品質を意味します。

shadowStrength プロパティ

```
shadowStrength:Number  [read-write]
```

Scaleform のバージョン: 2.0.41 以降

シャドウのぼかしの程度を制御します。有効な値は 0 から 15.9 (浮動小数点) までです。

デフォルト値は 0 です。

関連項目:

```
shadowBlurX
shadowBlurY
```

shadowStyle プロパティ

```
shadowStyle:String  [read-write]
```

Scaleform のバージョン: 1.1.31 以降

`shadowColor` と組み合わせて、テキスト フィールドに対するシャドウのレンダリングを制御します。特に、shadowStyle フォーマット ストリングは、適用されるテキスト レイヤーのセットを、その座標と共に記述します。例:

```
field.shadowStyle = "s{0,-1.5}{-1.4,1.2}{1.4,1.2}t{0,0}";
field.shadowColor = 0xff0000;
```

このストリング内では、"s" という文字は、`shadowColor` のカラー値で描画されるシャドウ レイヤーの範囲を定めますが、"t"という文字は標準の `TextField.textColor` カラー値で描画される前面のテキスト レイヤーを指定します。この"s" や "t" のような区切り文字の後に、各テキスト レイヤーがレンダリングされる、オフセットを記述した座標ペアが続きます。テキストに恒等変換が適用される場合、各座標単位は 1 ピクセルにマッピングされます。"t" という区切り文字とその座標ペアがこのスタイル ストリングから省略された場合、テキストは "t{0,0}" と指定されたときとまったく同じように描画されます。

現在のインプリメンテーションでは、シャドウ レイヤーを使用すると、さらに三角形が生成され、描画プリミティブ呼び出しが行われるので注意してください。テキストが複数回レンダリングされるか

らです。結果的に得られるパフォーマンスを計測して、可能な場合はシャドウの使用を制限することを強くお勧めします。

関連項目:

```
shadowColor
TextField.textColor
```

7.6 イメージの置き換え

この章では、イメージの置き換えを制御するエクステンションについて説明します。

setImageSubstitutions () メソッド

```
public setImageSubstitutions(substInfoArr:Array) : void
public setImageSubstitutions(substInfo:Object) : void
public setImageSubstitutions(null) : void
```

Scaleform のバージョン: 2.0.38 以降

サブストリングのイメージへの置き換えを、テキスト フィールドに設定します。
ストリング置き換えは、SWF に埋め込まれたイメージのみで動作します。このようなイメージは、エクスポート名を持つために、リンケージを割り当てておく必要もあります。イメージを SWF に埋め込むには、以下の手順に従います:

1. ビットマップ イメージをライブラリにインポートする。
2. ライブラリでそのイメージを右クリック (Windows)、または control-クリック (Macintosh) して、コンテキスト メニューから [リンケージ] を選択する。
3. [ActionScript に書き出し] と [最初のフレームに書き出し] を選択して、任意の名前 (myImage など) を [識別子] テキスト ボックスに入力する。
4. [OK] をクリックして、リンケージ識別子を設定する。

イメージをインポートして、リンケージ識別子を割り当てた後は、BitmapData インスタンスを作成する必要があります。以下は ActionScript コードの例です:

```
import flash.display.BitmapData;
var imageBmp:BitmapData = BitmapData.loadBitmap("myImage");
```

注意: インポート ステートメント（`import flash.display.BitmapData;` または完全修飾名の `flash.display.BitmapData`）を使用すること忘れないでください。そうでないと、結果が"`undefined`" となります!

複数のイメージを置き換えとして使用する場合、上記の手順をイメージごとに繰り返して、異なるリンケージ ID を設定する必要があります。

単数置き換えの記述子は `Object` で、以下のメンバー セットが続きます:

`subString:String`
　　イメージに置き換えるサブストリングを指定します。このメンバーは必須です。このサブストリングの長さは最大で 15 文字までです。

`image : BitmapData`
　　イメージを指定します。必須。

`width : Number`
　　画面上のイメージの幅を指定します (ピクセル単位)。オプション。

`height : Number`
　　画面上のイメージの高さを指定します (ピクセル単位)。オプション。

`baseLineY : Number`
　　元のイメージのピクセル単位で、イメージのベースラインの Y オフセットを指定します (変形なし)。オプション。デフォルトでは、この値はイメージの高さと同じです。したがって、イメージの下辺はベースライン上に表示されます。

`id : String`
　　`"updateImageSubstitution"` 呼び出しの最初のパラメータとして使用する、置き換えの ID を指定します。オプション。

`"substInfoArr"` は、このようなオブジェクトの配列であり、さらに “`substInfo`” はそのオブジェクトのインスタンスでなければなりません。バージョン `setImageSubstitutions(substInfo:Object)` は、単数の置き換えしか設定できませんが、バージョン `setImageSubstitutions(substInfoArr:Array)` は複数の置き換えを設定します。

`setImageSubstitutions` へのすべての呼び出しは、内部のリストに置き換えを追加するので注意してください。このような呼び出しをすべてクリアするには、`setImageSubstitutions(null)` を呼び出します。

`setImageSubstitutions` の呼び出し後に、ActionScript コードで置き換えを伴う配列、または単数記述子オブジェクトへのリファレンスを維持する必要はありません。ただし、ActionScript コードのどこかでそれを参照する必要がある場合は、維持してください。テキスト フィールドから置き換えの配列を取り戻す方法はないからです。

パラメータ

`substInfoArr:Array` – 置き換え記述子オブジェクトの配列 (上記を参照してください)
`substInfo:Object` – 単数置き換え記述子オブジェクトです (上記を参照してください)。
`null` – すべての置き換えをクリアします。

関連項目:

updateImageSubstitution()

例:

```
var b1 = BitmapData.loadBitmap("smile1");
var b2 = BitmapData.loadBitmap("smile2");
var b3 = BitmapData.loadBitmap("smile3");
var a = new Array;
a[0] = { subString:"=)", image:b1, baseLineY:35, width:20, height:20, id:"sm=)" };
a[1] = { subString:":-)", image:b2, baseLineY:20, id:"sm:-)" };
a[2] = { subString:":\\", image:b3, baseLineY:35, height:100 };
a[3] = { subString:":-\\", image:b1 };
t.setImageSubstitutions(a);
```

テキスト フィールドにサブストリング "=)" が含まれるとすぐに、このサブストリングは、引用符なしで、"smile1" リンケージ識別子を持つイメージに置き換えられます。

updateImageSubstitution () メソッド

```
public updateImageSubstitution(id:String, image:BitmapData) : void
```

Scaleform のバージョン: 2.0.38 以降

`setImageSubstitutions` 関数で前回作成したテキスト置き換え用のイメージを、置き換えまたは削除します。

パラメータ

`id:String` – 置き換えるものの ID で、`setImageSubstitutions` 呼び出しに使用された記述子オブジェクトの id メンバーと同じです。

`image:BitmapData` – 新規のイメージを指定します。`null` の場合は、置き換えるイメージは完全に削除されます。

関連項目: `setImageSubstitutions()`

例:

```
t.updateImageSubstitution("sm=)", b3);
```

以下は、埋め込まれたイメージのアニメーションの例です。更新は onEnterFrame ハンドラ、または setInterval を使って行うことができます。`updateImageSubstitution` を呼び出したときは、テキストの再フォーマットは行われません。したがって新規のイメージのサイズは以前のイメージと同じになるはずです。

```
var phase = 0;
var b1a = BitmapData.loadBitmap("smile1a");
```

```
var b2a = BitmapData.loadBitmap("smile2a");

onEnterFrame = function()
{
   if (phase % 10 == 0)
   {
      if (phase % 20 == 0)
      {
         _root.t.updateImageSubstitution("sm=)", b1);
         _root.t.updateImageSubstitution("sm:-)", b2);
      }
      else
      {
         _root.t.updateImageSubstitution("sm=)", b1a);
         _root.t.updateImageSubstitution("sm:-)",b2a);
      }
   }
   ++phase;
}
```

7.7 IME のサポート

この章では IME のサポートに対するエクステンションについて説明します。

disableIME プロパティ

```
disableIME:Boolean  [read-write]
```

Scaleform のバージョン: 2.1.50 以降

`true` に設定された場合、IME はこのテキスト フィールドではアクティブになりません。

デフォルト値は `false` です。

getIMECompositionStringStyle () メソッド

```
public function getIMECompositionStringStyle(categoryName:String): Object
```

Scaleform のバージョン: 2.1.50 以降

IME 色設定の指定されたカテゴリーに対する、カラー設定記述子オブジェクトを返します。 `categoryName` とこの記述子オブジェクトの詳細については、`setIMECompositionStringStyle` を参照してください。

パラメータ

`categoryName:String` – IME 色設定のカテゴリー (setIMECompositionStringStyle を参照してください) 。

戻り値

Object – そのカテゴリーのカラー設定記述子 (setIMECompositionStringStyle を参照してください)。

関連項目:

`setIMECompositionStringStyle`
`disableIME`

setIMECompositionStringStyle () メソッド

```
public function setIMECompositionStringStyle(categoryName:String,
                                    styleDescriptor:Object): Number
```

Scaleform のバージョン: 2.1.50 以降

IME 色設定の適切なカテゴリーにスタイルを設定します。categoryName パラメータは以下の値を含んでいる場合があります:

`"compositionSegment"` – 入力ストリング全体に対する色設定を設定します。
`"clauseSegment"` – 文節セグメントに対する色設定を設定します。
`"convertedSegment"` – 変換されたテキスト セグメントに対する色設定を設定します。
`"phraseLengthAdj"` – 語句の長さの調整に対する色設定を設定します。
`"lowConfSegment"` – 信頼度の低い文字に対する色設定を設定します。

`styleDescriptor` は、オプションで以下のメンバー セットを持つ Object インスタンスです:

`textColor : Number`
テキストの色

`backgroundColor : Number`
背景色

`underlineColor : Number`
　　下線の色

`underlineStyle : String`
　　下線のスタイル。有効な値は以下のとおりです:

　　`"single"`
　　`"thick"`
　　`"dotted"`
　　`"ditheredSingle"`
　　`"ditheredThick"`

すべての色は、0xRRGGBB というフォーマットで指定されます。この場合 RR は赤のコンポーネントの 16 進数表現 [0..255]、BB は青のコンポーネントの 16 進数表現 [0..255]、GG は緑のコンポーネントの 16 進数表現 [0..255] です。

パラメータ
　　`categoryName:String` – IME 色設定のカテゴリー (説明を参照してください)
　　`styleDescriptor:Object` – カラー設定記述子オブジェクト (説明を参照してください)

関連項目:
　　`getIMECompositionStringStyle`
　　`disableIME`

8 TextFormat クラス エクステンション

TextFormat クラスは、色、フォント サイズ、アンダーラインなどの文字のフォーマット情報を表します。

alpha プロパティ

```
alpha:Number  [read-write]
```

Scaleform のバージョン: 2.0.41 以降

テキストのアルファ透明度値をパーセント単位で制御します。有効な値は 0 から 100 (%) までです。標準の Flash TextFormat.color は、アルファ透明度の設定を許可していないので、このエクステンションを使って、テキストを部分的に、または完全に透明にすることができます。

関連項目:

```
TextFormat
HTML <FONT ALPHA='xx'>
```

9 Stage クラス エクステンション

標準の Stage クラス プロパティのフルセットをサポートする他に、Scaleform はステージのサイズのトラッキングを向上するエクステンションも導入しています。このようなエクステンションには、“visibleRect”、“safeRect”、“originalRect” というプロパティが含まれ、さらに、現在表示されているフレームの四角形を表現する、“onResize” ハンドラに対する追加パラメータが含まれています。詳細は下記を参照してください。

ActionScript 1.0 では、Stage.visibleRect のように、エクステンションを直接参照することができます。ActionScript 2.0 はそれほど寛容ではなく、このような参照には不満を表します。ActionScript 2.0 コンパイラを抑制するために、以下の別のアクセス シンタックスを使用する必要があります: `Stage[“visibleRect”]..`

visibleRect プロパティ

```
visibleRect:flash.geom.Rectangle  [read-only]
```

Scaleform のバージョン: 2.2.56 以降

このプロパティには、現在表示されている四角形が含まれています。この四角形は、縦横比、スケール モード、行揃え、およびビューポートのスケール、またはそのいずれかを変更し、現在どの領域が表示されているかを知りたいときに、変更されます。この四角形は、負数の topLeft 角座標を備えている場合があります。

関連項目:

`safeRect`
`originalRect`

safeRect プロパティ

```
safeRect:flash.geom.Rectangle  [read-only]
```

Scaleform のバージョン: 2.2.56 以降

このプロパティには、現在設定されている安全な四角形が含まれています。この四角形は、ユーザーが `GFx::Movie::SetSafeRect` メソッドを使って設定する必要があります。設定されない場合、“`visibleRect`” プロパティと同じ四角形が含まれます。

関連項目:

`visibleRect`
`originalRect`

originalRect プロパティ

```
originalRect:flash.geom.Rectangle  [read-only]
```

Scaleform のバージョン: 2.2.56 以降

このプロパティは常に、オリジナルの SWF の寸法で、オリジナルの SWF の四角形を含みます。したがって、Flash Studio の [ドキュメント プロパティ] でサイズが (600px×400px) に設定されていた場合、この四角形は { 0, 0, {600, 400} } という値を含みます。

関連項目:

`visibleRect`
`safeRect`

onResize イベント ハンドラ

```
onResize = function([visibleRect:flash.geom.Rectangle]) {}
```

Scaleform のバージョン: 2.2.56 以降

この onResize イベント ハンドラは、エクステンションとして追加パラメータを備えています。このパラメータは、現在表示されている四角形を表します。Stage.visibleRect エクステンション プロパティで返されるものと同じです。

関連項目:

`visibleRect`

translateToScreen()メソッド

```
public function translateToScreen(pt:Object): Point
```

Scaleform のバージョン n: 3.3.84

スクリーン座標系にマッピングされている Stage 座標系内のポイントを返すメソッドです。

パラメータ

`pt:Object` – 変形対象のポイントの座標を表す `X` および `Y` メンバーを持ったオブジェクトです。汎用 `Object` の代わりに `flash.geom.Point` も使用できます。

戻り値

ポイント–スクリーン座標系にマッピングされている入力ポイントです。

10　Array クラス エクステンション

ロケール固有の並べ替え

Flash バージョンの Array.sort メソッドと Array.sortOn メソッドは、Unicode 値に応じた並べ替えを行います。ただし、これは常に正しい順序になるとは限りません。特にフランス語、スペイン語、ドイツ語などではそうです。Unicode の並べ替えの場合、発音区別符号 (ウムラウト記号、アクセント記号、鋭アクセント記号など) を含むすべての文字列は、‘z’ の後に表示されます。したがって、['á', 'z', 'a'] という配列を並べ替える場合、Flash は現在設定されているロケールに関係なく、['a', 'z', 'á'] を返します。ただし、フランス語の場合、ロケール固有の並べ替えの結果は ['a', 'á', 'z'] になるはずです。
この問題を解消するため、Scaleform は Array.sort/sortOn メソッドにもう 1 つ Array.LOCALE というオプションを導入しています:

```
var arr = [‘á’, ‘z’, ‘a’];
var newArr1 = arr.sort(); // returns [‘a’, ‘z’, ‘á’]
var newArrLoc = arr.sort(Array.LOCALE); // returns [‘a’, ‘á’, ‘z’]
var newArrRev = arr.sort(Array.LOCALE | Array.DESCENDING); // [‘z’, ‘á’, ‘a’]
```

大文字と小文字を区別しないロケール固有の並べ替えもサポートされます:

```
var arr = [‘Á’, ‘z’, ‘a’];
var a = arr.sort(Array.LOCALE | Array.CASEINSENSITIVE); // returns [‘a’, ‘Á’, ‘z’]
```

カーネルで該当する機能をサポートしないプラットフォームでは、この機能は正しく動作しない場合があるので注意してください。この場合、Array.LOCALE の並べ替えは、Unicode 値を使った通常の並べ替えとして行われます。

Scaleform のバージョン: 3.0.65 以降

関連項目

`String.localeCompare`

11 String クラス エクステンション

localeCompare() メソッド

この関数は 2 つ以上の文字列の並べ替え順序を比較して、数値でその結果を返します。このメソッドはロケール固有の方法で比較します。文字列が同じであれば戻り値は 0 です。元のストリング値がパラメータで指定されたストリング値よりも前に来る場合、戻り値は負数の値になります。元のストリング値がパラメータで指定された値の後に来る場合、戻り値は正数の値になります。
ロケール固有の文字列操作の詳細については、「Array クラス エクステンション」を参照してください。
カーネルで該当する機能をサポートしないプラットフォームでは、この機能は正しく動作しない場合があるので注意してください。その場合、この関数は通常の文字列と Unicode 値との比較を行います。

```
public localeCompare(other:String [, caseInsensitive:Boolean]) : Number
```

Scaleform のバージョン: 3.0.65 以降

パラメータ

`other:String` – 比較する文字列です。

`caseInsensitive:Boolean` – 大文字と小文字を区別しない比較に使用されるオプションのパラメータです。このパラメータが指定されていない場合、比較は大文字と小文字を区別して行われます。

関連項目

`Array.sort`

12　Video クラス エクステンション

標準の Flash では、一つの Video オブジェクトだけが、同時に、一つの NetStream オブジェクトからデータを受け取ることができます。Scaleform では、この制約がなくなり、複数の Video オブジェクトが、同じ一つの NetStream オブジェクトにアタッチできますし、同一のビデオ データを受け取ることができます。

clipRect プロパティ

```
clipRect: flash.geom.Rectangle  [read-write]
```

Scaleform のバージョン: 3.0.70 以降

このプロパティは、ビデオ オブジェクトがオリジナルのビデオ フレームの一部（領域）だけを表示することを可能にします。

例:

```
video.clipRect = new flash.geom.Rectangle(10, 20, 100, 100);
```

13　3Di クラス エクステンション

3Di は Scaleform 内に組み込まれている AS2 エクステンションの基本セットを使って実装されています。
3Di では次の ActionScript エクステンションが追加されました。

Scaleform のバージョン: 3.2.82 以降

_z

オブジェクトの Z 座標（デプス）を設定します。デフォルト値は 0 です。

_zscale

Z 方向のオブジェクトのスケール（倍率）をパーセントを単位に設定します。デフォルト値は 100 です。

_xrotation

X 軸（水平）を中心としたオブジェクトの回転を設定します。デフォルト値は 0 です。

_yrotation

Y 軸（垂直）を中心としたオブジェクトの回転を設定します。デフォルト値は 0 です。

_matrix3d

16 個の浮動小数点のアレイ（4×4 のマトリックス）を使ってオブジェクトの完全 3D 変換を設定します。この値を NULL にセットするとすべての 3D 変換は削除されオブジェクトは 2D に戻ります。

_perspfov

オブジェクト上の Field Of View パースペクティブ角を設定します。有効な角度は 0 より大きく 180 未満です。設定しない場合、オブジェクトは 55°のデフォルト・ルート FOV 角を継承します。

これら新しい拡張機能を使用するには、ActionScript にてグローバル変数 gfxExtensions を true に設定する必要があります。

14　グローバル エクステンション

noInvisibleAdvance プロパティ

```
noInvisibleAdvance : Boolean
```

Scaleform のバージョン: 2.1.52 以降

true に設定されている場合、このプロパティは、非表示のムービー クリップすべての再生をオフにします。これは、多くの非表示のムービー クリップを持つ SWF のパフォーマンスの向上に使用できる場合があります。Flash は常に非表示のムービー クリップを進めます (タイムライン アニメーションを実行し、フレームの ActionScript コードを呼び出します)。したがって、このプロパティを “true” に設定すると、Scaleform と Flash の動作に違いが出る場合があります。

関連項目:

```
MovieClip.noAdvance
```

imecommand 関数

```
imecommand(command:String, parameters:String) : Void
```

Scaleform のバージョン: 2.1.50 以降

この関数は fscommand に似ていますが、Scaleform の IME インプリメンテーションとの通信に限って使用されます。これは Scaleform が内部で使用します。

getIMECandidateListStyle () 関数

```
public function getIMECompositionStringStyle(categoryName:String): Object
```

Scaleform のバージョン: 2.1.50 以降

IME の候補リストに対する、カラー設定記述子オブジェクトを返します。この記述子オブジェクトの詳細については、setIMECandidateListStyle を参照してください。

戻り値

Object – 候補リストのカラー設定記述子

関連項目:

```
setIMECandidateListStyle
TextField.disableIME
```

setIMECandidateListStyle () 関数

```
public function setIMECandidateListStyle(styleDescriptor:Object)
```

Scaleform のバージョン: 2.1.50 以降

引数として渡されたオブジェクトの候補リストに、候補リストのスタイルを設定します。

styleDescriptor は、オプションで以下のメンバー セットを持つ Object インスタンスです:

```
      styleObject.textColor                     = 0x5EADFF;
      styleObject.selectedTextColor       = 0xFFFFFF;
      styleObject.fontSize                      = 20;
      styleObject.backgroundColor               = 0x001430;
      styleObject.selectedTextBackgroundColor   = 0x2C5A99;
      styleObject.indexBackgroundColor          = 0x12325A;
      styleObject.selectedIndexBackgroundColor  = 0x2C5A99;
      styleObject.readingWindowTextColor        = 0xFFFFFF;
      styleObject.readingWindowBackgroundColor = 0x001430;
      styleObject.readingWindowFontSize         = 20;
```

```
textColor : Number
```
テキストの色

```
selectedTextColor : Number
```
選択したテキストの色

```
fontSize : Number
```
行と行インデックスのテキストのフォント サイズ

```
backgroundColor : Number
```
選択していない行のテキスト部分の背景色

```
seletedTextBackgroundColor : Number
```
選択した行のテキスト部分の色

```
indexBackgroundColor : Number
```
選択していない行のインデックス部分の背景色

```
selectedIndexBackgroundColor : Number
```
選択した行のインデックス部分の背景色

```
readingWindowTextColor : Number
```
読み入力ウィンドウのテキストの色

`readingWindowBackgroundColor : Number`
　　読み入力ウィンドウの背景色

`readingWindowFontSize : Number`
　　読み入力ウィンドウのテキストのフォント サイズ
すべての色は、0xRRGGBB というフォーマットで指定されます。この場合 RR は赤のコンポーネントの 16 進数表現 [0..255]、BB は青のコンポーネントの 16 進数表現 [0..255]、GG は緑のコンポーネントの 16 進数表現 [0..255] です。

関連項目:
　　`getIMECandidateListStyle`
　　`TextField.disableIME`

15　スタンダード・メソッドとイベントハンドラー・エクステンション

Key クラス

```
Key.getCode(controllerIdx:Number)
Key.getAscii(controllerIdx:Number)
Key.isDown(controllerIdx:Number)
Key.isToggled(controllerIdx:Number)
```

キーボード／コントローラのオプションパラメータを取得する Key クラス・メソッドです。指定しない場合、controllerIdx はデフォルトの 0 になります。

```
Key.onKeyDown(controllerIdx:Number)
```

onKeyDown はイベントを生成するキーボード／コントローラの追加パラメータを受信する Key リスナー・メソッドです。

Selection クラス

```
Selection.setFocus(ch:Object, controllerIdx:Number)
Selection.getFocus(controllerIdx:Number)
Selection.getBeginIndex(controllerIdx:Number)
Selection.getEndIndex(controllerIdx:Number)
Selection.setSelection(start:Number, end:Number, controllerIdx:Number)
Selection.getCaretIndex(controllerIdx:Number)
```

キーボード／コントローラのオプションパラメータを取得する Selection クラス・メソッドです。指定しない場合、controllerIdx はデフォルトの 0 になります。

```
Selection.onSetFocus(old:Object, new:Object, controllerIdx:Number)
```

onSetFocus はイベントを生成するキーボード／コントローラの追加パラメータを受信する Selection リスナー・メソッドです。

MovieClip/Button/TextField

```
MovieClip.onSetFocus(old:Object, controllerIdx:Number)
MovieClip.onKillFocus(new:Object, controllerIdx:Number)
Button.onSetFocus(old:Object, controllerIdx:Number)
Button.onKillFocus(new:Object, controllerIdx:Number)
TextField.onSetFocus(old:Object, controllerIdx:Number)
TextField.onKillFocus(new:Object, controllerIdx:Number)
TextField.onChanged(controllerIdx:Number)
```

MovieClip/Button/TextField は、イベントを生成するキーボード／コントローラの追加パラメータを受信するリスナー・メソッドです。

System.capabilities

`System.capabilities.numControllers` – システムで検出されたコントローラの個数を返します。