Autodesk®
Scaleform®

扩展 ActionScript 参考

本文档描述了 Scaleform 4.1 中的扩展 ActionScript 2.0 的相关内容。

作者： Artem Bolgar
版本： 4.03
最新修订： 2012 年 9 月 17 号

Autodesk®
GAMEWARE

Copyright Notice

Autodesk® Scaleform® 4.2

Autodesk Scaleform 联系方式：

文档	扩展 ActionScript 2.0 参考
地址	Autodesk Scaleform Corporation 6305 Ivy Lane, Suite 310 Greenbelt, MD 20770, USA
网站	www.scaleform.com
邮箱	info@scaleform.com
电话	(301) 446-3200
传真	(301) 446-3199

目录

1 介绍

Autodesk Scaleform SDK 是一个结构简洁、性能高效、功能丰富的 Adobe® Flash 矢量图形引擎，它建立于净室工具，对控制台和 PC 游戏开发者特别有用。Scaleform 整合了应用广泛的可视化创作工具，如 Adobe® Creative Suite®，具备先进的硬件图形加速功能，满足了前沿游戏开发者的需求。

由于 Flash 主要为 Web 应用，而不是游戏或者其他开发应用所设计，其功能在某些方面非常有限，如聚焦手柄、鼠标支持、文本域支持和 IME 输入支持。Scaleform 通过为 ActionScript 类增加“扩展”，来改进这些方面的基本功能。为使 Scaleform Player 中扩展功能有效，必须将`_global.gfxExtensions` 设置为 `true`。

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

在大多数情况下，开发者在 FLA 文件的第一帧中就加入扩展功能有效声明。如果未设置该变量，尽量与 Flash 保持最高等级的兼容，Scaleform Player 将忽略所有扩展功能的引用。性。

开发者应该清楚本文档中所描述的扩展功能在标准的 Flash® Player 中是不能使用的，只在 Scaleform 中有效。每项扩展功能对应一个 Scaleform 版本号，来区分增加扩展的播放器 SDK 发行号。在扩展功能版本号之前的 Scaleform Player 版本中将不支持。

尽管 Scaleform 尽最大力量去维持不同发行版本中扩展 API 函数的支持性和一致性，但是我们保留在将来 Scaleform 发行版本中修改、重命名或者移除扩展 API 函数的权利。如果有较大改动，我们将在主发行版本中体现出来，并尽早发出通告。

2 扩展鼠标类

除了支持完整的标准鼠标方法类，Scaleform 引进了一些其他的扩展功能，可以跟踪多种类型鼠标光标并可识别 RIGHT, MIDDLE 和其他的鼠标事件（至今已有 16 项）。本章首先简单描述基本鼠标功能，然后详细描述了 Scaleform 鼠标扩展功能。如需鼠标对象更详细的资料，建议开发者参考 Flash Mouse Class 文档。

2.1 多类型光标支持

在有些应用平台中，如 Wii™游戏控制台，需要支持多种类型鼠标光标。Scaleform 3.2 中最多支持 4 中鼠标光标。GFx::Movie::SetMouseCursorCount(unsigned n)方法函数用来设置支持的光标数。GFx::Movie::HandleEvent with GFx::MouseEvent 方法作为将鼠标事件赋给 Scaleform 内核的对象参数。GFx::MouseEvent 和 GFx::MouseCursorEvent 构造器具有扩展的参数来指派以零点为基准的鼠标索引。

2.2 扩展按钮事件

鼠标按键事件如 onRollOver, onRollOut, onDragOver, onDragOut, onPress, onRelease, onReleaseOutside 在 Scaleform 扩展有效时接收一个或两个参数。第一个参数为产生事件鼠标基于零点基准的索引，从而，若索引号为 1 的鼠标光标产生 onPress 事件，则该参数值为 1：

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

onPress 和 onRelease 的第二个参数是数字属性，指明事件是由鼠标/光标还是键盘所引起。如果是键盘，则值为 -1，而如果是鼠标/光标，则值为 0。这个属性对于确定事件的起源非常有用：

```
mc.onPress = function(mouseIdx:Number, keyboardOrMouse:Number)
{
      if (keyboardOrMouse == 0)
            . . .
      else
            . . .
```

```
        . . .
}
```

参数 2 为 onRollOver/Out 和 onDragOver/Out 事件选项。这些参数指定在相同文字上的 rollover/dragover 嵌套事件索引。如果这些功能处理参数未定义，则 onRollOver/onRollOut 和 onDragOver/onDragOut 事件对中只能触发一次：onRollOver/onDragOver 只在光标移到字符位置时触发，而 onRollOut/onDragOut 在光标最后离开时触发。如果定义了处理这些事件的参数 2，则可产生嵌套的 onRollOver/onRollOut 和 onDragOver/onDragOut 事件（独立于光标），该参数则描述了基于零点位置的嵌套序号：初始时的 onRollOver/onDragOver 事件参数 2 为 0；若第二个光标滚过相同字符，则第二个 onRollOver/onRollOut 事件触发并且第二个参数设置序号为 1。如果任何一个光标离开该字符，onRollOut/onDragOut 事件将被触发并将参数 2 设置为 1；最后的 onRollOut/onDragOut 事件触发并设置参数 2 为 0。

因此，参数 2 的定义改变了 onRollOver/onRollOut 和 onDragOver/onDragOut 事件的触发方式。在参数 2 被定义时，用户需要关注嵌套事件。

```
mc.onRollOver = function(mouseIdx:Number, nestingIdx:Number)
{
      // 只在 nestingIdx == 0 时做翻滚动作
      if (nestingIdx == 0)
            doOverAnimation();
      . . .
}
mc.onRollOut = function(mouseIdx:Number, nestingIdx:Number)
{
      //只在 nestingIdx == 0 时做翻滚动作
      if (nestingIdx == 0)
            doOutAnimation();
      . . .
}
```

注意，传统事件如 RollOver，RollOut 等不提供扩展参数，因此不能区分被哪个光标所触发。从而不推荐在多鼠标光标中使用此类事件。
为了支持除用于 onPress、onRelease 等的左按钮之外的鼠标按钮，增加了按钮事件的新辅助版本：

- onPressAux
- onReleaseAux
- onReleaseOutsideAux
- onDragOver
- onDragOut

如果在事件目标上定义这些辅助事件处理程序，它们就会被调用。只针对包含 index != 0（0 指数是指鼠标左按钮）的按钮调用它们。始终只针对鼠标左按钮调用一般处理程序，而不管是否存在辅助事件处理

程序。这些处理程序还具有与其标准副本（本节前面已定义）相同的函数签名。不过，辅助事件处理程序为按钮指数提供一个额外参数。

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

2.3 鼠标类事件

在 ActionScript 中可安装鼠标事件探测器来接收鼠标移动、左键按键和鼠标滚轮事件。在 Flash 中这些事件通常通过使用 Mouse.addListener 方法来安装一个鼠标事件探测对象来实现如下方法：

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

安装了探测对象后，当任何时候事件发生时相关的方法就会被调用。在 Flash 中 OnMouseDown 和 onMouseUp 方法只被鼠标左键调用，没有其他的公共接口来接收鼠标事件。

为应对这个限制（也为支持多鼠标光标），Scaleform 扩展了这些方法调用，当 `_global.gfxExtensions` 变量设置为真时，使其能够获得扩展参数。新的函数特性如下：

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

当分配的鼠标侦测函数获得至少一个扩展参数，Scaleform 将调用这个侦测函数对应的方法来 处理 RIGHT, MIDDLE 和以及其他鼠标事件，使得逻辑上能够侦测到此类事件。LEFT，RIGHT， MIDDLE 对应的数值分别为 1，2，3。当前最多支持 16 种事件。为保持兼容，若函数未定义参数，只调用 LEFT 鼠标按钮事件，在 Flash 中也相同。以下为装载此类处理功能的例子：

```
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);
```

新产生的鼠标 down/up 句柄至少包含两个参数：按键和位置。参数一代表按键，描述鼠标按键；与鼠标 Mouse ["LEFT"], Mouse ["RIGHT"]和 Mouse ["MIDDLE"]扩展常数相比较解释其含义（这里用 Mouse["LEFT"] 来代替 Mouse.LEFT 表示方法是为了与 ActionScript 2.0 编译器兼容）。参数二为位置，提供鼠标按下时光标下面对象的顶部位置路径信息；如没有此类路径信息该参数可不作定义。

在 Flash 中只可调用鼠标左键按钮句柄，所有扩展参数都“未定义”。把“未定义”值看成是 LEFT 鼠标按钮，确保与 Flash player 兼容。

为支持多类型鼠标光标需要以下扩展参数：

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

参数 `mouseIdx` 包含了触发事件的光标基于零点基准的索引号。x 和 y 参数则包含了光标位置（_root 空间）。`onMouseDown` 句柄的 `dblClick` 参数表示发生一个双击事件。这种情况下，参数值为 `true`。

2.4 改变鼠标光标

Flash 中有三种类型的鼠标光标：

- 箭头 - 光标位于普通对象之上。
- 手形 - 在 useHandCursor 属性为真时，光标位于按钮和链接上。
- I 形 – 光标位于文本域上。

Flash 中不无法侦测到鼠标光标的变化，Scaleform 提供了 C++ API 函数和 ActionScript 鼠标扩展类来管理自定义光标。在 C++中，可以使用 `GFx::StateBag::SetUserEventHandler` 函数安装事件句柄，当鼠标光标改变时触发。使用这个句柄，如 FxPlayer.cpp 中所示，既可改变游戏引擎所绘制的光标，也可改变系统光标。同时，也可以选择完全采用 ActionScript 实现自定义光标，能体现出其动画和艺术控制优势。SDK 中提供了该方法的一个实例文 *Mouse.fla/swf*。

鼠标实例实现自定义光标的方法为将其当作在场景中的一个简单视频剪辑，称为‘Cursor _M’。光标包括了不同的帧，不同帧中分别为“手形”，“箭头”和“I-形”；可以通过移动时间线来改变光标图标。当侦测到鼠标移动时，视频剪辑的_x 和_y 坐标随着鼠标而改变。

为了改变位于对象上面的光标类型，鼠标实例采用了 Scaleform 扩展函数 `Mouse.setCursorType`。该函数当光标需要改变时调用 cursorType 和 mouseIndex 参数，允许光标改变动作被侦测到。通过比较鼠标参数(Mouse["ARROW"]，Mouse["HAND"] and Mouse["IBEAM"])的 cursorType 类型并适当得改变光标帧来展示鼠标实例。

下面为处理鼠标实例中多类型自定义光标的源代码。建议查看实例文件获得详细的信息。

```
_global.gfxExtensions = 1;
// 隐藏系统光标。
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;
    }
}

// 运用 Mouse.show 和 Mouse.hide 函数
// 将对光标产生影响。
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 ActionScript 扩展鼠标类

Scaleform 中 ActionScript 扩展鼠标类中增加了一些静态方法和属性。ActionScript 1.0 中可以直接引用扩展函数，如 `Mouse.setCursorType(...)`。而在 ActionScript 2.0 没有如此方便，直接引用将导致异常，为减轻 ActionScript 2.0 编译器负担，需使用不用的语法，如:`Mouse["setCursorType"](..)`。

getButtonsState() 静态方法

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

Scaleform 版本： 2.2

返回鼠标按钮状态。返回的值为一个位掩码值，每个位代表一个鼠标按钮序号。但对应的位为 1 则代表该按钮已被按下。

参数

`mouseIndex : Number` – 基于零点的鼠标索引

getTopMostEntity()静态方法

```
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 版本： 1.2.34, 从 2.2 版本开始支持多类型鼠标光标

静态方法返回鼠标光标下的目标字符或者指定对象。此方法在有按钮句柄设置（如 onPress, onMouseDown, onRollOver 等）和无按钮句柄设置的字符并不相同。这样可以区分出哪些字符无鼠标事件句柄需要处理。

此方法在 `MovieClip.hitTest` 正好起到反向作用， `hitTest` 检查 x 和 y 是否对应在目标对象范围之内，而 `getTopMostEntity` 返回对应于 x 和 y 坐标的实际对象。从而，`Mouse.getTopMostEntity(x, y).hitTest(x, y, true) == true`。

参数

`mouseIndex : Number` –基于零点的鼠标索引

`testAll : Boolean` – 指示按钮句柄（false）字符或任何字符（false）。如该参数并没指定，`getTopMostEntity` 默认其为‘true’。

`x : Number, y : Number` – 字符查找位置

示例:

```
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 静态方法

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

Scaleform 版本： 2.2

该方法返回对应的鼠标光标位置，基于_root 位置空间。返回的值为一个 flash.geom.Point 实例。

参数

`mouseIndex : Number` – 基于零点的鼠标索引

setCursorType() 静态方法

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

Scaleform 版本： 1.2.32

本静态方法根据参数 cursorType 改变鼠标光标。参阅“改变鼠标光标”获得详细信息。

参数

`cursorType : Number` – 显示光标类型， Mouse.ARROW, Mouse.HAND, Mouse.IBEAM.
`mouseIndex : Number` –基于零点的鼠标索引

3 扩展按钮类

hitTestDisable 属性

`hitTestDisable:Boolean  [读-写]`

Scaleform 版本： 2.1.51

当本属性设置为 `true` 时，`MovieClip.hitTest` 函数在按钮点击时忽略该按钮。并且对应于该按钮的所有鼠标事件将无效。

默认值为 `false`

可参考：

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

topmostLevel 属性

`topmostLevel:Boolean  [读-写]`

Scaleform 版本： 2.1.50

如果本属性设置为 `true`，则此字符将在其他字符顶层显示，不管其层叠深度如何。这在自定义鼠标光标中当光标需要在所有对象顶层显示时候有用。默认值为 false

在将几个特性标识为“topmostLevel”时，绘图命令如下所示：

- 对于 Scaleform 3.0.71 而言，将特性标识为“topmostLevel”的绘图命令取决于将这一属性设置为“真”的命令（因此，首先被标识为“置顶”的特性将首先绘出）；
- 从 Scaleform 3.0.72 开始，绘图命令都是相同的，因为它不再标识“置顶”特性，即将对象 A 绘制在对象 B 下面，置顶后对象 A 仍然位于对象 B 下方，而无论使用何种命令将“topmostLevel”的属性设置为“真”。

注释：一旦某一属性被标识为“topmostLevel”，swapDepth ActionScript 功能将不会对这一特性产生任何影响。

可参考：

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

focusGroupMask 属性

```
focusGroupMask : Number
```

Scaleform 版本： 3.3.84

此属性将一个位掩码设置为一个舞台人物及其全部 (**ALL**) 子项。此位掩码将焦点组所有权指定给该角色，意思是只有位掩码中表明的控制器才能将焦点移动到角色内并在其中移动焦点。可以使用 `setControllerFocusGroup` 扩展方法将焦点组与控制器相关联。

例如，我们假设“按钮 1”(button1) 只能作为控制器 0 的焦点，而“电影剪辑 2”(movieclip2) 只能作为控制器 0 和 1 的焦点。要实现这一行为，请将焦点组与控制器相关联：

```
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`”位掩码可以设置为父电影剪辑 (parent movieclip)。这将把掩码值传播到所有其子项。

4 MovieClip 扩展类

hitTest ()方法

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

Scaleform 版本： 2.1.51

标准的 3 参数 hitTest 还拥有一个布尔参数扩展项，指示是否忽略或不显示其子剪辑。Flash 中 hitTest 的默认为返回“true”，甚至在 x，y 位置处于可视的子剪辑也同样如此（如_visible 属性设为 false；子剪辑_alpha 设为 0 时当作不可视处理）。

参数

`ignoreInvisibleChildren:Boolean` – 应该设置为 true 忽略所有可视子剪辑。

hitTestDisable 属性

```
hitTestDisable:Boolean  [读-写]
```

Scaleform 版本： 1.2.32

当本属性设置为 `true` 时，`MovieClip.hitTest` 函数在按钮点击时忽略该按钮。并且对应于该按钮的所有鼠标事件将无效。

默认值为 `false`。

可参考：

`TextField.hitTestDisable`
`Button.hitTestDisable`

noAdvance 属性

```
noAdvance : Boolean
```

Scaleform 版本：2.1.52

本属性设置为 true 时，关闭视频剪辑和所有子剪辑的前进动作。这可以改进性能。
注释，Flash 中视频剪辑通常有前进动作（按时间线执行动画，调用帧中的 ActionScript 等）。因此，在 Scaleform 和 Flash 中设置属性为“true”可导致不同的效果。

可参考：
```
_global.noInvisibleAdvance
```

topmostLevel 属性

```
topmostLevel:Boolean  [读-写]
```

Scaleform 版本： 2.1.50

若本属性设置为 true，则此字符将在其他字符顶层显示，不管其层叠深度如何。这在自定义鼠标光标中当光标需要在所有对象顶层显示时候有用。默认值为 false。

在将几个特性标识为“topmostLevel”时，绘图命令如下所示：

- 对于 Scaleform 3.0.71 而言，将特性标识为“topmostLevel”的绘图命令取决于将这一属性设置为“真”的命令（因此，首先被标识为“置顶”的特性将首先绘出）；
- 从 Scaleform 3.0.72 开始，绘图命令都是相同的，因为它不再标识“置顶”特性，即将对象 A 绘制在对象 B 下面，置顶后对象 A 仍然位于对象 B 下方，而无论使用何种命令将“topmostLevel”的属性设置为“真”。

注释：一旦某一属性被标识为“topmostLevel”，swapDepth ActionScript 功能将不会对这一特性产生任何影响。

可参考
```
TextField.topmostLevel
Button.topmostLevel
```

rendererString 属性

```
rendererString:String  [读-写]
```

Scaleform 版本： 2.2.55

本属性允许在任何 MovieClip 实例中通过 ActionScript 发送自定义变量到渲染器。如果本属性置位，字符串值将作为用户数据传递给渲染器。

无默认值。

示例：

```
myMovieInstance.rendererString = "SHADER_Blur"
```

rendererFloat 属性

`rendererFloat:Number  [读-写]`

Scaleform 版本： 2.2.55

本属性允许在任何 MovieClip 实例中通过 ActionScript 发送自定义变量到渲染器。如果本属性置位，浮点值将作为用户数据传递给渲染器。

无默认值。

示例：

```
myMovieInstance.rendererFloat = 4; // 例如 C++ 枚举值
```

disableBatching 属性

`disableBatching:Boolean`

Scaleform 版本: 4.0.17

此属性禁用针对自定义绘图的网格生成批处理。

focusGroupMask 属性

`focusGroupMask : Number`

Scaleform 版本： 3.3.84

此属性将一个位掩码设置为一个舞台人物及其全部 (**ALL**) 子项。此位掩码将焦点组所有权指定给该角色，意思是只有位掩码中表明的控制器才能将焦点移动到角色内并在其中移动焦点。可以使用 `setControllerFocusGroup` 扩展方法将焦点组与控制器相关联。

例如，我们假设“按钮 1”(button1) 只能作为控制器 0 的焦点，而“电影剪辑 2”(movieclip2) 只能作为控制器 0 和 1 的焦点。要实现这一行为，请将焦点组与控制器相关联：

```
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”位掩码可以设置为父电影剪辑 (parent movieclip)。这将把掩码值传播到所有其子项。

5 扩展 NetStream 类

Scaleform 添加功能函数到 NetStream 类，该类可以添加字幕和音频轨道到视频文件。

onMetaData 事件句柄

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

Scaleform 版本： 3.0.63

事件句柄在视频文件播放时接收信息，包括 Scaleform 中的两个附加对象属性。这些属性用来获得视频文件自带的字幕和音频轨道信息。

对象传递给 onMetaData 事件句柄，包含了两个附加的只读属性。

1. audioTracks – 视频文件音频轨道编码描述对象组
   audioTracks 序列中的每个对象包含以下属性：
   - channelsNumber – 音频轨道通道数 (1–单声道, 2–立体声, 6–5.1 环绕声)。
   - totalSamples – 音频轨道中声音取样数。
   - trackIndex – 轨道序号，本属性用来选择音频轨道，只需要将其赋给 NetStream.audioTrack 属性即可 (参见下面实例)。
   - sampleRate – 声音取样率。

2. subtitleTracksNumber – 视频文件中包含的字幕通道数。

音频通道属性

audioTrack:数量 [读-写]

Scaleform 版本： 3.0.63

本属性用来设置/获取当前视频文件播放音频的轨道。

例子：

```
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;
```

字幕通道属性

```
subtitleTrack:Number  [读-写]
```

Scaleform 版本： 3.0.63

该属性可以设置和获得当前字幕通道。关闭字幕将属性设置为 0。

onSubtitle 事件句柄

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

Scaleform 版本： 3.0.63

该事件句柄当字幕消息准备完毕需要被播放时调用。

例子：

```
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 函数

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

Scaleform 版本： 3.0.68

该函数用来设置内部视频缓存区数量。视频缓存区用来保存视频编码输出内容，被称为“帧池”。当用来编码的 CPU 资源负载不均衡，拥有较多帧池可以让播放画面更加平滑。默认值为 1。

该方法应该在视频开始播放前被调用（例如，在函数 NetStream.play()前被调用）。

setReloadThresholdTime 函数

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

Scaleform 版本： 3.0.68

该函数设置重载时间，以秒为单位。当输入缓存区的数据量小于重载限度，则 Scaleform 视频库请求读取下一个文件。时间限度根据视频文件比特率和重载时间设置自动判定。重载时间默认值为 0.8 秒。

在视频播放时读取游戏数据，使用本方法可以减少搜索请求。例如，当设置 Netstream 缓存时间（NetStream.setBufferTime() 方法）为 2 秒，设置时间限度为 1 秒，游戏数据可以连续读取 1 秒。但是游戏数据读取时间应该不大于重载时间内。当游戏数据过大，可能需要分块读取。如果游戏数据读取未能在重载时间内完成，由于视频数据缓存区为空，则视频播放将“卡片”。

该方法应该在视频开始播放前被调用（例如，在 NetStream.play()函数前调用）。

6 扩展选择类

选择类，在其他函数中允许用于管理文本域，视频剪辑和按钮的输入焦点。

Scaleform 扩展了选择类的焦点功能函数。ActionScript 1.0 中可以直接引用选择类扩展函数，如 `Selection.captureFocus()`。而在 ActionScript 2.0 没有如此方便，直接引用将导致异常，为减轻 ActionScript 2.0 编译器负担，需使用不用的语法，如：`Selection["captureFocus"]()`。

alwaysEnableArrowKeys 静态属性

```
alwaysEnableArrowKeys : Boolean
```

Scaleform 版本： 1.2.34

本静态属性允许箭头键改变焦点，甚至在“`_focusrect`”属性设置为 `false` 时也是如此（应用于焦点捕获时）。默认情况下，在黄色焦点通过设置“`_focusrect = false`”被禁止时 Flash 不允许使用箭头键来改变焦点；要改变这个行为，需将 `alwaysEnableArrowKeys` 属性设置为 “`true`”。

示例：

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

alwaysEnableKeyboardPress 静态属性

```
alwaysEnableKeyboardPress : Boolean
```

Scaleform 版本： 3.0.63

该静态属性允许通过按下空格键或回车键触发 onPress / onRelease 事件，甚至在 `focusrect` 属性设置为 `false` 时也可以（在焦点被捕获时应用）。默认情况下，若黄色焦点框通过 `focusrect = false` 设置禁止，则 `Flash` 不允许通过键盘控制按钮；为改变此特性，设置 `alwaysEnableKeyboardPress` 属性为 `true`。

例如：

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

captureFocus() 静态方法

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

Scaleform 版本： 1.2.34

本静态属性可编程实现捕捉或释放键盘焦点。

`captureFocus` 中 `doCapture` 设置为 true（或者无任何参数）可捕获键盘焦点，并允许方向键来改变焦点，效果如同首次按下 Tab 键。
`captureFocus` 中 `doCapture` 设置为 false 则释放键盘焦点，效果如同焦点有效时鼠标移动。

参数

`doCapture:Boolean` – 可选择，表示键盘焦点需要捕获还是释放。如果这个参数被忽略则 `captureFocus` 与该参数设置为 true 时行为相同。

`controllerIdx:Number` – 可选，表示哪个键盘/控制器用于捕获操作。默认情况下，使用控制器 0。

示例：

```
Selection["captureFocus"](); // 与传递“true”相同
Selection["captureFocus"](false);
```

disableFocusAutoRelease 静态属性

```
disableFocusAutoRelease : Boolean
```

Scaleform 版本： 1.2.34

本静态属性用来控制一个鼠标动作是否需要释放键盘焦点（Flash 中标准行为为需要释放）。默认情况下，若焦点被捕获，黄色矩形显示（允许方向键来改变焦点），则鼠标动作释放焦点。当设置本属性为 true 时将禁止此动作。

示例：

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

disableFocusKeys 静态属性

```
disableFocusKeys : Boolean
```

Scaleform 版本： 2.2.58

本静态属性禁止所有焦点键的处理（TAB, Shift-TAB 和方向键），因此，用户可以实现其自身焦点键管理。

示例：

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

可参考：

```
moveFocus()
```

disableFocusRolloverEvent 静态属性

```
disableFocusRolloverEvent : Boolean
```

Scaleform 版本： 2.0.37

本属性用来禁止焦点被键盘改变时触发 rollover/out 事件。设置本属性为 `true` 可禁止此行为。

示例：

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

modalClip 静态属性

```
modalClip : MovieClip
```

Scaleform 版本： 2.2.58

本静态属性设置焦点管理视图中的视频剪辑为“modal”剪辑。这意味着按键焦点（TAB,Shift-TAB 和方向键）只在特定视频剪辑中才可以移动焦点矩形，如，只限制在视频剪辑的“tabable”子项。

示例：

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

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

可参考：

```
disableFocusKeys
moveFocus()
```

moveFocus() 静态方法

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

Scaleform 版本: 2.2.58

本静态方法用来通过模拟键盘按键动作来移动焦点矩形，这些按键包括 TAB, Shift-TAB 或方向键。本方法与 `disableFocusKeys` 和 `modalClip` 属性配合使用，可以实现自定义焦点管理功能。

参数

`keyToSimmulate:String` – 模拟按键名称：“up”, “down”, “left”, “right”, “tab”, “shifttab”。

`startFromMovie:Object` – 可选参数指定字符；moveFocus 将使用其将当前焦点作为起始点。该属性可能为 null 或未定义，意为当前获得焦点的字符为起始指针；这在制定第三个参数选项时或许有用。

`includeFocusEnabledChars:Boolean` – 可选标记，设置允许 moveFocus 用在只有 `focusEnabled` 可设置的字符上或者只有 `tabEnabled / tabIndex` 属性可设置的字符上。如果该标记被指定或设置为 `false`，则只有具有 `tabEnabled / tabIndex` 属性设置的字符可以用与焦点移动。

`controllerIdx:Number` – 可选，表示哪个键盘/控制器用于捕获操作。默认情况下，使用控制器 0。

示例

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

返回值

返回新聚焦字符，或无法识别字符返回未定义内容。

可参考：

`disableFocusKeys`
`modalClip`

findFocus() 静态方法

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

Scaleform 版本： 3.3.84

此静态方法用来通过以模拟方式按下列各键之一来查找下一个焦点项：TAB、Shift-TAB 或箭头键。此方法可与 `disableFocusKeys` 和 `setModalClip/getModalClip` 扩展一起用来实现自定义焦点管理。

参数

`keyToSimulate:String` – 要模拟的键的名称：“向上”、“向下”、“向左”、“向右”、“tab”、“shifttab”。

`parentMovie` – 用作模态剪辑的电影剪辑。焦点项搜索只在此剪辑的子项内执行。可以是“空”(Null)。

`loop` - 循环焦点的布尔旗标。例如，如果当前焦点项位于底部，而键为“向下”键，则 findFocus 要么返回 “null”（如果此旗标为 “false”（假）的话），要么返回最上面的焦点项（如果旗标为 “true”（真）的话）。

`startFromMovie:Object` – 可选参数，指定 findFocus 将使用的人物，而非作为起点的当前焦点项。此属性可能是 “null”或 “ undefined”（未定义），意思是将当前焦点人物用作起点。

`includeFocusEnabledChars:Boolean` – 可选旗标，允许 moveFocus 移动到只设置 `focusEnabled` 属性的人物以及设置 `tabEnabled/tabIndex` 属性集的人物上。如果没把旗标指定或设置为 `false`（假），则只有设置了 `tabEnabled/tabIndex` 属性的人物将参与焦点移动。

`controllerIndex` - 正在操纵焦点的控制器的一个零基指数。这可与焦点组一起用来提供多控制器焦点支持。

示例：

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

返回

返回要作为焦点的下一个人物，或者返回 `null` --- 如果没有找到该人物。

另请参见：

`disableFocusKeys`
`setMmodalClip`
`getMmodalClip`

setModalClip() 静态方法

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

Scaleform 版本： 3.3.84

此静态方法把指定的电影剪辑设置为用于焦点管理的“模态”剪辑。这意味着 TAB、Shift-TAB 和箭头键将在所有 “tabable” 子项之间的指定电影剪辑内移动焦点。

参数

`modalClip` – 一个模态剪辑。
`controllerIndex` - 控制器的一个零基指数。

getModalClip() 静态方法

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

Scaleform 版本： 3.3.84

此静态方法为指定控制器返回模态剪辑。

参数

`controllerIndex` - 控制器的零基指数。

返回

一个模态剪辑，或者未定义 --- 如果没有找到的话。

setControllerFocusGroup() 静态方法

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

Scaleform 版本： 3.3.84

此静态方法将 `controllerIndex` 指明的控制器与一个焦点组关联。默认情况下，所有控制器都与焦点组 0 关联，这意味着它们在用同一焦点。不过，可以使每个控制器都使用其各自的焦点。例如，如果两个控制器都有单独的焦点（在分屏使用情况下），那么 `setControllerFocusGroup (1,1)` 将为控制器 1 创建一个单独的焦点组。调用 `setControllerFocusGroup (1,0)` 将使控制器 0 和 1 再次共享同一焦点。

参数

`controllerIndex` – 控制器的零基指数。
`focusGroupIdx` – 焦点组的零基指数。

示例：

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

返回

返回 true --- 如果成功的话。

getControllerFocusGroup() 静态方法

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

Scaleform 版本： 3.3.84

此静态方法返回与指定控制器关联的焦点组。

参数

`controllerIndex` - 物理控制器的零基指数。

返回

焦点组的基于零的指数。

getFocusArray() 静态方法

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

Scaleform 版本： 3.3.84

此静态方法返回一组控制器指数，这些指数当前在指定电影剪辑/按钮/文本框上拥有焦点。

参数

`mc` – 一个电影剪辑、按钮或文本框。

返回

基于零的指数（数字）组。

getFocusBitmask() 静态方法

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

Scaleform 版本： 3.3.84

此静态方法返回一个位掩码，其中每个位代表一个当前在指定电影剪辑/按钮/文本框上有焦点的控制器。

参数

`mc` - 一个电影剪辑、按钮或文本框。

返回

控制器的一个位掩码。

getControllerMaskByFocusGroup() 静态方法

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

Scaleform 版本： 3.3.84

此静态方法返回一个位掩码，其中每个位代表一个与指定焦点组关联的控制器。返回 `setControllerFocusGroup` 函数设置的状态。

参数

`focusGroupIdx` - 焦点组的一个指数。

返回

控制器的一个位掩码。

numFocusGroups 属性

```
numFocusGroups : Number
```

Scaleform 版本： 3.3.84

返回焦点组的数目，通过调用 `setControllerFocusGroup` 函数来设置。如果焦点组 0 和 3 是活动的，则 numFocusGroups 将返回 2。

7 扩展 TextField 类

Scaleform 引入了众多 TextField 扩展类，使得比 Flash 8 中的内嵌 TextField 类更加易用和强大。 有些为复制了 Flash 9 中的内嵌 TextField 相同或相似的功能。增加的扩展功能是为了更好地管理 IME、对齐、文本大小和外观、文本滤镜效果（如阴影，模糊等）、嵌入图像、选择和剪贴等操作。

7.1 般性功能

本章描介绍了扩展属性和方法的一般性用途。

autoFit 属性

```
autoFit:Boolean  [读-写]
```

Scaleform 版本： 2.0.39

设置 on/off 字体自动。

默认值为 false。

appendText () 方法

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

Scaleform 版本： 2.0.37

用 newText 参量将字符串追加到文本结尾处。这种方法比用（+=）符号来附加字符串更加有效（例如，`my_txt.text += appendingText`），在文本域中包含大量文本时特别有用。

注释：如果将某一样式表格应用到文本字段中，则本方法不适用。

参数

`newText:String` – 附加字符串

可参考

`appendHtml()`

appendHtml () 方法

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

Scaleform 版本： 2.0.37

用 newHtml 参量附加 HTML 文本到文本域结尾处。这种方法比用（+=）符号来附加更加有效（如 `txt.htmlText += moreHtml`）。位于 htmlText 属性的+=追加符将生成 HTML 标识，附加新的 HTML 文本并对其进行解析。该函数增加了 HTML 解析功能，如只能从 newHtml 字符串常量中解析出 HTML 文本（这也是在 newHtml 常量中 HTML 文本应该用特定格式表示的原因，这对于+=操作符并无要求）。此方法在文本域中包含大量文本时特别有用。

注释：如果将某一样式表格应用到文本字段中，则本方法不适用。

参数

`newHtml:String` – 追加到显存文本的 HTML 字符串。

可参考：

`appendText()`

caretIndex 属性

```
caretIndex:Number  [读-写]
```

Scaleform 版本： 2.0.37

该属性描述了插入位置（符号或者指针形式）的索引。如果文本域没有获得焦点，字符位置没有被显示出来（闪烁光标形式），则表示上次获得焦点的字符位置；如果文本域从没获得过焦点，则表示为 0。该属性包含的值与 `Selection.getCaretIndex()`方法函数返回值相同；但是，`caretIndex` 即使在文本域没有获得焦点的时候也可发挥作用，而 `Selection.getCaretIndex()`在此情况下则返回-1。

符号索引以零为基准（因此，首位置为 0）。

可参考：

`Selection.getCaretIndex()`

getCharBoundaries () 方法

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

Scaleform 版本： 2.0.37

该方法返回字符边框的矩形位置。注释，该方法返回的矩形为每个字符轮廓的靠前估略值。这意味着返回的矩形是一个大致的轮廓框（看下图，红色矩形描述了‘a’、‘w’和‘k’的轮廓）：

参数

`charIndex:Number` – 以零为基准的字符位置索引值（如，首位置为 0，第二个位置为 1，依次类推）。

返回值

flash.geom.Rectangle – 矩形的 x，y 坐标起始位置值，定义了字符的边框。与文本域的空间位置相对应，如（0，0）点对应于文本域的左上角。

可参考：

`getExactCharBoundaries()`

getExactCharBoundaries () 方法

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

Scaleform 版本： 2.0.37

该方法返回字符精确边框的矩形位置。注释，本方法返回的矩形框为字符轮廓的准确大小（而getCharBoundaries()返回的是靠前估略的轮廓大小）。矩形的高度为整个字符串的整体高度。查看下图：红色矩形显示了‘a’、‘w’和‘k’的精确边框：

参数

charIndex:Number -以零为基准的字符位置索引值（如，首位置为 0，第二个位置为 1，依次类推）。

返回值

flash.geom.Rectangle - 矩形的 x，y 坐标起始位置值，定义了字符的边框。与文本域的空间位置相对应，如（0，0）点对应于文本域的左上角。

可参考：

getCharBoundaries()

getCharIndexAtPoint () 方法

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

Scaleform 版本： 2.0.37

本方法返回用 x 和 y 参数表示的基于零点的字符位置索引值。

参数

x:Number – 字符的 x 坐标。

y:Number – 字符的 y 坐标。

返回值

Number -以零为基准的字符位置索引值（如，首位置为 0，第二个位置为 1，依次类推）。坐标不位于任何字符上返回-1。

getFirstCharInParagraph () 方法

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

Scaleform 版本： 2.0.37

该方法用 charIndex 参量返回段落中第一个字符的索引。

参数

charIndex:Number -以零为基准的字符位置索引值（如，首位置为 0，第二个位置为 1，依次类推）。

返回值

Number – 以零点为基准的段落中第一个字符索引。

getLineIndexAtPoint () 方法

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

Scaleform 版本： 2.0.37

本方法返回用 x 和 y 参数表示的基于零点的字符串位置索引值。

参数

x:Number -字符的 x 坐标。
y:Number -字符的 y 坐标。

返回值

Number - 以零为基准的文本串索引值（如，第一行文本串为 0，第二行字符串为 1，依次类推）。坐标不位于任何字符串上返回-1。

getLineLength() 方法

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

Scaleform 版本： 2.0.37

返回特定文本行包含的字符数。

参数

`lineIndex:Number` - 以零为基准的文本行数量（如，第一行文本行为 0，第二行文本行为 1，依次类推）。

返回值

Number – 字符行中包含的字符数量。

getLineMetrics() 方法

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

Scaleform 版本： 2.0.43

返回一特定文本行中的格式信息。返回的对象包括以下部分：

`ascent : Number`
以像素为单位描述文本高度，即从基准文本行到顶行的高度。

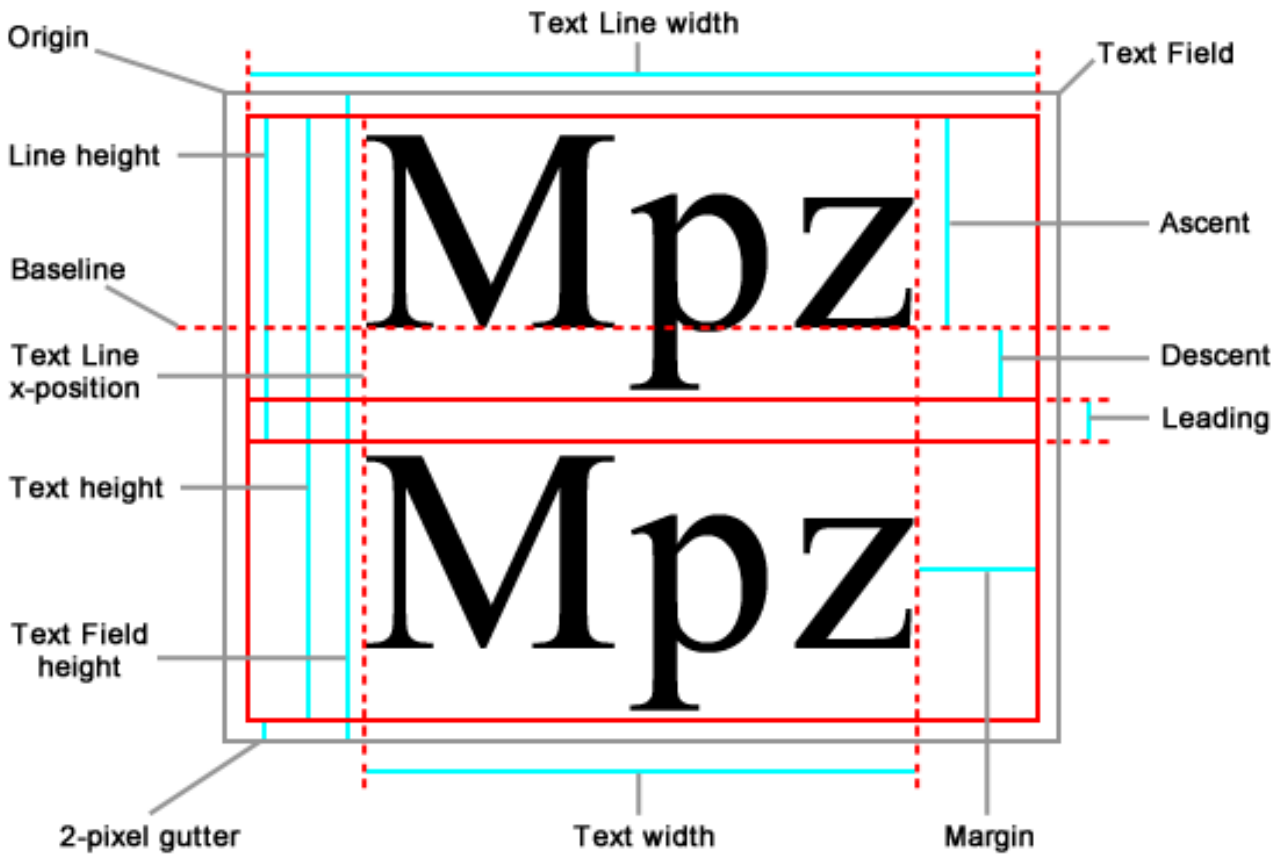
`descent : Number`
以像素为单位描述文本高度，即从基准文本行到底行的高度。

`height : Number`
以像素为单位描述文本高度，选择行高度（不需要全部文本空间）。

`leading : Number`
文本行之间的垂直距离值。

`width : Number`
以像素为单位描述被选择文本行的宽度值（不需要全部文本空间）。

`x : Number`
以像素为单位描述首字符的左侧位置。



参数

`lineIndex:Number` -以零为基准的文本行索引值（如，第一行文本行为 0，第二行文本行为 1，依次类推）。

返回值

Number – 一个对象的格式信息。

getLineOffset () 方法

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

Scaleform 版本： 2.0.37

用 `lineIndex` 参量返回段落中第一个字符的索引。

参数

`lineIndex:Number` - 以零为基准的文本行索引值（如，首行为 0，第二行为 1，依次类推）。

返回值

Number -以零为基准的文本行中第一个字符索引。

getLineText () 方法

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

Scaleform 版本： 2.0.43

用 `lineIndex` 参量返回文本行中的文本索引。

参数

`lineIndex:Number` - 以零为基准的文本行索引值（如，首行为 0，第二行为 1，依次类推）。

返回值

String – 特定行中包含的文本串。

hitTestDisable 属性

```
hitTestDisable:Boolean  [读-写]
```

Scaleform 版本： 1.2.32

设置为 `true` 时，当产生点击动作时 `MovieClip.hitTest` 函数将忽略该文本域。并且，所有鼠标事件对该文本域都无效。

默认值为 false。

noTranslate 属性

```
noTranslate:Boolean  [读-写]
```

Scaleform 版本 1.2.34

设置为 true 时，禁止文本域的 `GFxTranslator::Translate` 函数调用。

默认值为 false。

numLines 属性

numLines:Number [读-写]

Scaleform 版本 2.0.43

描述多行文本域中的文本行数量。如果 wordWrap 属性设置为 true，文本覆盖时文本行数量将递增。

topmostLevel 属性

topmostLevel:Boolean [读-写]

Scaleform 版本： 2.1.50

若属性设置为 true，文字将显示在其他文字的上层，忽略其层叠深度。该属性在实现自定义鼠标和弹出式窗口时需要在对象前面显示时有用。默认值为 false。

在将几个特性标识为“topmostLevel”时，绘图命令如下所示：

- 对于 Scaleform 3.0.71 而言，将特性标识为“topmostLevel”的绘图命令取决于将这一属性设置为“真”的命令（因此，首先被标识为“置顶”的特性将首先绘出）；
- 从 Scaleform 3.0.72 开始，绘图命令都是相同的，因为它不再标识“置顶”特性，即将对象 A 绘制在对象 B 下面，置顶后对象 A 仍然位于对象 B 下方，而无论使用何种命令将“topmostLevel”的属性设置为“真”。

注释：一旦某一属性被标识为“topmostLevel”，swapDepth ActionScript 功能将不会对这一特性产生任何影响。

可参考：

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

focusGroupMask 属性

focusGroupMask : Number

Scaleform 版本： 3.3.84

此属性将一个位掩码设置为一个舞台人物及其全部 (**ALL**) 子项。此位掩码将焦点组所有权指定给该角色，意思是只有位掩码中表明的控制器才能将焦点移动到角色内并在其中移动焦点。可以使用 setControllerFocusGroup 扩展方法将焦点组与控制器相关联。

例如，我们假设“按钮 1”(button1) 只能作为控制器 0 的焦点，而“电影剪辑 2”(movieclip2) 只能作为控制器 0 和 1 的焦点。要实现这一行为，请将焦点组与控制器相关联：

```
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”位掩码可以设置为父电影剪辑 (parent movieclip)。这将把掩码值传播到所有其子项。

7.2 选择和剪贴操作

本节介绍选择和剪贴操作的扩展属性。

alwaysShowSelection 属性

```
alwaysShowSelection:Boolean  [读-写]
```

Scaleform 版本 2.1.45

默认情况下，文本域无焦点获得。Scaleform 不高亮显示被选择内容。当该属性设置为 true 时，即使文本域没有获得焦点，Scaleform 也高亮显示被选择文本为灰色。选择和未选择颜色区分将被覆盖（见“可参考”）。

默认值为 false。

可参考：

```
selectionBkgColor
selectionTextColor
inactiveSelectionBkgColor
inactiveSelectionTextColor
```

copyToClipboard () 方法

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

Scaleform 版本： 2.1.45

复制文本到剪贴板。参数选项如下。

参数

`richClipboard:Boolean` – 若为 true，则文本及其类型信息复制到剪贴板。默认值与 `useRichTextClipboard` 属性一致。

`startIndex:Number` – 复制文本段的起始索引。默认值与 `selectionBeginIndex` 相同。

`endIndex:Number` – 复制文本段的终止索引。默认值与 `selectionEndIndex` 相同。

可参考

`useRichTextClipboard`
`selectionBeginIndex`
`selectionEndIndex`
`cutToClipboard()`
`pasteFromClipboard()`

cutToClipboard ()方法

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

Scaleform 版本 2.1.45

复制和删除文本到剪贴板。参数选项如下。

参数

`richClipboard:Boolean` – 若为 true，则文本及其类型信息复制到剪贴板。默认值与 useRichTextClipboard 属性一致。

`startIndex:Number` – 复制文本段的起始索引。默认值与 `selectionBeginIndex` 相同。

`endIndex:Number` – 复制文本段的终止索引。默认值与 `selectionEndIndex` 相同。

可参考：

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

inactiveSelectionBkgColor 属性

`inactiveSelectionBkgColor:Number  [读-写]`

Scaleform 版本 2.1.45

指定被选择活动文本背景颜色。只在 `alwaysShowSelection` 属性设置为 true 时有效。颜色的描述格式为：0xAARRGGBB ，其中 AA 表示 alpha 透明通道十六进制表示数[0,255]，RR - 红色十六进制表示数[0,255]，BB - 蓝色十六进制表示数[0,255]，GG - 绿色十六进制表示数[0,255]。

注释，确保 alpha 透明通道设置为 0，因为本例中无需绘制任何图画（因为 alpha 设置为 0 表示完全透明）。因此，这种颜色与 Flash 中经常使用的背景色略有不同。例如，设置该属性填充深红色需要使用 0xFFFF0000 值（alpha 和红色设置为 0xFF (255)），但是对于常用的“backgroundColor”属性使用 0xFF0000 值就已足够。

可参考：

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

inactiveSelectionTextColor 属性

`inactiveSelectionTextColor:Number  [读-写]`

Scaleform 版本： 2.1.45

指定被选择活动文本颜色。只在 `alwaysShowSelection` 属性设置为 true 时有效。颜色的表示格式为：0xRRGGBB ，其中 AA 表示 alpha 透明通道十六进制表示数[0,255]，RR - 红色十六进制表示数[0,255]，BB - 蓝色十六进制表示数[0,255]，GG - 绿色十六进制表示数[0,255]。

注释，确保 alpha 透明通道设置为 0，因为本例中无需绘制任何图画（因为 alpha 设置为 0 表示完全透明）。因此，这种颜色与 Flash 中经常使用的背景色略有不同。例如，设置该属性填充深红色需要使用 0xFFFF0000 值（alpha 和红色设置为 0xFF (255)），但是对于常用的“textColor”属性使用 0xFF0000 值就已足够。

可参考：

- `alwaysShowSelection`

inactiveSelectionBkgColor
selectionBkgColor
selectionTextColor

noAutoSelection 属性

noAutoSelection:Boolean [读-写]

Scaleform 版本： 2.1.45

若设置为 true，焦点切换到文本域禁止自动选择。

默认为 false。

pasteFromClipboard () 方法

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

Scaleform 版本： 2.1.45

从剪贴板粘贴文本。参数选项如下。

参数

richClipboard:Boolean –	若为 true，文本类型属性从剪贴板粘贴。默认值与 useRichTextClipboard 属性相同。
startIndex:Number –	从剪贴板粘贴文本段的起始索引。默认值与 selectionBeginIndex 相同。
endIndex:Number –	从剪贴板粘贴文本段的终止索引。默认值与 selectionEndIndex 相同。

可参考：

useRichTextClipboard
selectionBeginIndex
selectionEndIndex
copyToClipboard()
cutToClipboard()

selectionBeginIndex 属性

selectionBeginIndex:Number [读-写]

Scaleform 版本： 2.1.45

表示以零为基准的当前选择字符串中第一个字符索引值。若文本未被选择，本属性值即为 caretIndex 的值。本属性与 Selection.getBeginIndex()函数的返回值相同；不同的是 Selection.getBeginIndex()函数在文本未获得焦点时也可使用，此时，Selection.getBeginIndex()的返回值为-1。

可参考：

- caretIndex
- selectionEndIndex
- Selection.getBeginIndex()

selectionEndIndex 属性

selectionEndIndex:Number [读-写]

Scaleform 版本： 2.1.45

表示以零为基准的当前选择字符串中最后一个字符索引值。若文本未被选择，本属性值即为 caretIndex 的值。本属性与 Selection.getEndIndex()函数的返回值相同；不同的是 Selection.getEndIndex()函数在文本未获得焦点时也可使用，此时，Selection.getEndIndex()的返回值为-1。

可参考：

- caretIndex
- selectionBeginIndex
- Selection.getEndIndex()

selectionBkgColor 属性

selectionBkgColor:Number [读-写]

Scaleform 版本： 2.1.45

描述当前被选择文本的背景色。颜色的表示格式为：0xRRGGBB ，其中 AA 表示 alpha 透明通道十六进制表示数[0,255]，RR - 红色十六进制表示数[0,255]，BB - 蓝色十六进制表示数[0,255]，GG - 绿色十六进制表示数[0,255]。

注释，确保 alpha 透明通道设置为 0，因为本例中无需绘制任何图画（因为 alpha 设置为 0 表示完全透明）。因此，这种颜色与 Flash 中经常使用的背景色略有不同。例如，设置该属性填充深红色需要使用 0xFFFF0000 值（alpha 和红色设置为 0xFF (255)），但是对于常用的 `BkgColor` 属性使用 0xFF0000 值就已足够。

可参考：

```
inactiveSelectionTextColor
inactiveSelectionBkgColor
selectionTextColor
```

selectionTextColor 属性

`selectionTextColor:Number  [读-写]`

Scaleform 版本： 2.1.45

描述当前被选择文本的颜色。颜色的表示格式为：0xRRGGBB ，其中 AA 表示 alpha 透明通道十六进制表示数[0,255]，RR- 红色十六进制表示数[0,255]，BB - 蓝色十六进制表示数[0,255]，GG - 绿色十六进制表示数[0,255]。

注释，确保 alpha 透明通道设置为 0，因为本例中无需绘制任何图画（因为 alpha 设置为 0 表示完全透明）。因此，这种颜色与 Flash 中经常使用的背景色略有不同。例如，设置该属性填充深红色需要使用 0xFFFF0000 值（alpha 和红色设置为 0xFF (255)），但是对于常用的 `textColor` 属性使用 0xFF0000 值就已足够。

可参考：

```
inactiveSelectionBkgColor
inactiveSelectionBkgColor
selectionBkgColor
```

useRichTextClipboard 属性

`useRichTextClipboard:Boolean  [读-写]`

Scaleform 版本： 2.1.45

指定是否在文本复制和粘贴操作时带上文本格式信息。当设置为 `true` 时，Scaleform 在复制和粘贴文本时同时也复制和粘贴文本格式（如对齐、粗体和斜体等）。要实现此功能，复制和粘贴过程中的原始和目标文本域的属性 `useRichTextClipboard` 必须设置为 `true`。

默认值为 `false`。

7.3 文本大小和对齐

本节介绍文本大小和对齐控制。除了标准的对齐格式（左对齐、右对齐、居中），Scaleform 提供了垂直对齐，也就是垂直大小自适应功能。

fontScaleFactor 属性

`fontScaleFactor:Number [读-写]`

Scaleform 版本： 2.0.43

指定文本域中所有字体的字形缩放。所有字体大小可以通过改变 `fontScaleFactor` 值而改变。

默认值为 1.0。

textAutoSize 属性

`textAutoSize:String [读-写]`

Scaleform 版本： 2.0.43

textAutoSize 能够自动恢复文本字体大小。本属性的有效值为“`none`”、“`shrink`”和“fit”。如果该模式设置有效（“`shrink`”或“`fit`”）同时文本不适应于文本域，则文本大小按比例缩放到与文本域匹配为止，无需滚动操作。如果文本过小（字体大小为 5pt 左右）则默认的操作逻辑将停止字体缩放动作。设置 textAutoSize 为“`fit`”模式时，字体大小增大到填充满整个文本空间位置。设置为“`shrink`”模式时当字体大小超出文本空间范围时将减小字体的大小。

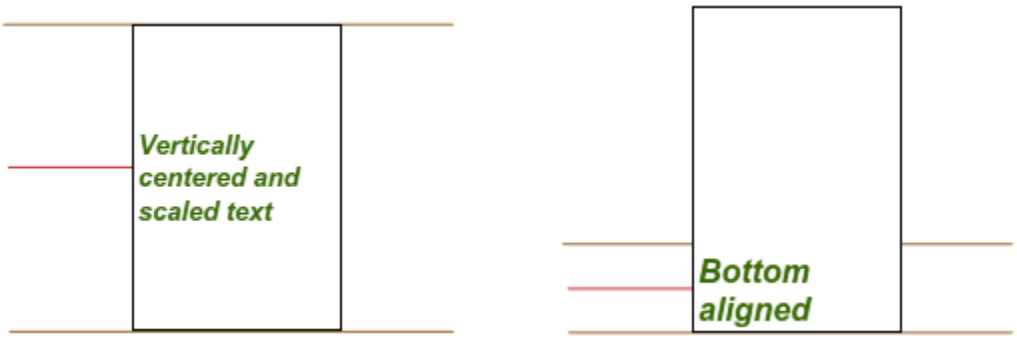
默认值为“none”。

verticalAlign 属性

verticalAlign:String [读-写]

Scaleform 版本： 2.0.43

设置文本框中文字的垂直对齐。该属性的有效值为：“none”（或“top”与“none”相同），“bottom”、“center”。若属性设置为“center”文本在文本域内居中，若设置为“bottom”，则在文本域中底部对齐。



默认值为“none”。

可参考：

verticalAutoSize
TextField.align

verticalAutoSize 属性

verticalAutoSize:String [读-写]

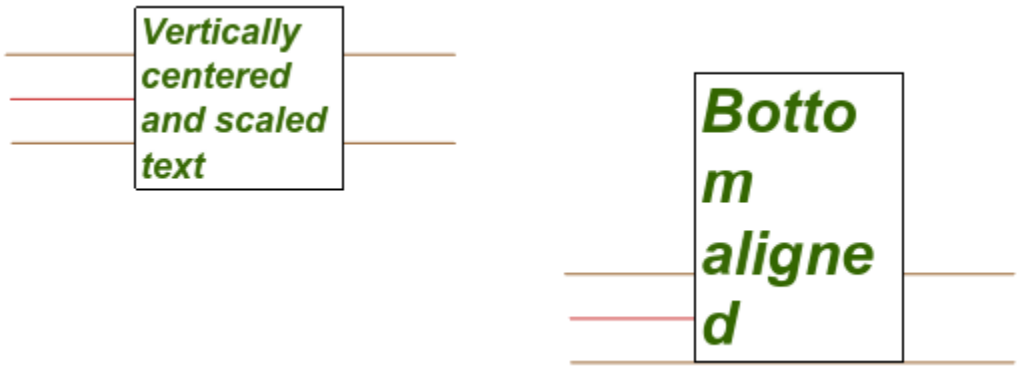
Scaleform 版本： 2.0.43

文本域的自动垂直大小控制和对齐。有效值为“none”（默认）、“top”、“bottom”、“center”。如果 verticalAutoSize 设置为“none”（默认值）将不会改变大小。

若 verticalAutoSize 设置为“top”，文本域中特性与 TextField.autoSize 模式下 wordWrap 设置为 true 时相同（只改变文本域底部边界，不改变右边界）。

若 verticalAutoSize 设置为“center”，文本域将上下均匀改变其大小。定位点位于文本域上下边的中点（参考下图）。

若 verticalAutoSize 设置为“bottom”，文本域将改变顶部边界。定位点位于文本域底部边界（参考下图）。



默认值为“none”。

可参考：

```
verticalAlign
TextField.autoSize
```

7.4 HTML 扩展

本节介绍 Scaleform 中 HTML 的扩展功能。

<FONT> 标签， ALPHA 特性

ALPHA="#xx"

设置文本 alpha 透明度，范围为 00 到 FF（0-255），0 代表全透明，255 代表全不透明。与 COLOR 属性相对应，ALPHA 属性用来设置 HTML 中的半透明文本。

<IMG> 标签

当前， Scaleform 2.0/2.1 中的<IMG> 标签用相应的链接导入图像到库。Scaleform 不支持从 Web 或者文件等外部数据源导入图像(不支持"http://", "file://" 及其他协议）。 上面描述如何导入到图像并指派链接。

以下为一个 HTML 使用<IMG>标签导入“myImage”图像的例子：

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

当前支持的 IMG 标签属性：

src –为库中的图像符号指定链接标识符。目前仅支持图像。此属性为必选；所有其他属性均为可选。目前还不支持外来文件（JPEG、GIF、PNG 和 SWF 文件）。这里可以使用用户协议“img://”，取代嵌入式图像链接标识符；在此情况下，将调用用户定义的虚拟方法 GFx::ImageCreator:: LoadImage。
width – 插入图像的宽度信息，以像素表示。

height – 插入图像的高度信息，以像素表示。

align – 指定文本域中嵌入图像的水平对齐属性。当前支持的唯一属性为“baseline”。

vspace – “baseline”对齐方式以像素为单位指定相对于基准线的偏移量：正值表示图像位于基线之上，负数则降低图像位置。

不支持属性：

id – 指定包含插入图像文件的视频剪辑实例名字（由 Flash Player 创建），图像文件来自 SWF 文件或者视频剪辑。

align – “left”和“right”。指定文本域中插入图像的垂直对齐方式。

hspace -“left”和“right”。指定围绕图像无文本区域的垂直空间。

注释，如果图像从另一个 SWF 文件中导入，则图像在 Flash 文件中应该用 IMG 标签明确指定。如果图像导入但未在 Flash 中使用，则在最终的 SWF 文件中将被丢弃。为防止图像丢弃，可创建一个视频剪辑并用 IMG 标签指定所有导入图像。不要遗忘导入视频剪辑，如果不明确使用并没有导出则 Flash 同样会在 SWF 文件中丢弃图像。

7.5 阴影效果控制

本节介绍不同阴影效果的控制（blur，knockout 等）。

blurX, blurY 特性

```
blurX:Number  [读-写]
blurY:Number  [读-写]
```

Scaleform 版本： 2.0.41

获得或者设置文本 blur 效果半径。有效值范围为 0 到 15.9(浮点数)。

默认值为 0。

可参考：
```
      blurStrength
```

blurStrength 特性

```
blurStrength:Number  [读-写]
```

Scaleform 版本： 2.0.41

获得或者设置文本 blur 强度。有效值范围为 0 到 15.9（浮点数）。

默认值为 0。

可参考：
```
      blurX
      blurY
```

shadowAlpha 特性

```
shadowAlpha:Number  [读-写]
```

Scaleform 版本： 2.0.41

控制色彩阴影的 alpha 透明值。有效值范围为 0.00 到 1.00，例如 0.25 设置 25%透明度。

可参考：
shadowColor

shadowAngle 特性

```
shadowAngle:Number  [读-写]
```

Scaleform 版本： 2.0.41

控制阴影角度，与 DropShadowFilter 类似。有效值范围为 0 到 360˚ (浮点数)。

默认值为 45。

可参考：
DropShadowFilter

shadowBlurX, shadowBlurY 特性

```
shadowBlurX:Number  [读-写]
shadowBlurY:Number  [读-写]
```

Scaleform 版本： 2.0.41

控制文本 blur 效果半径。有效值范围为 0 到 15.9(浮点数)。

默认值为 0。

可参考：
shadowStrength

shadowColor 特性

```
shadowColor:Number  [读-写]
```

Scaleform 版本： 1.1.31

指定阴影颜色（详情请参考 shadowStyle）。颜色的表示格式为：0xRRGGBB ，其中 RR - 红色十六进制表示数[0..255]，BB - 蓝色十六进制表示数[0..255]，GG - 绿色十六进制表示数[0..255]。

可参考：
shadowStyle

shadowDistance 特性

shadowDistance:Number [读-写]

Scaleform 版本： 2.0.41

以像素为单位控制阴影偏移距离，与 DropShadowFilter 相同。

可参考：
DropShadowFilter

shadowHideObject 特性

shadowHideObject:Boolean [读-写]

Scaleform 版本： 2.0.41

指示文本是否隐藏。值为 true 表示文本本身不现实，只显示其阴影。默认值为 false（显示文本本身）。

shadowKnockOut 特性

shadowKnockOut:Boolean [读-写]

Scaleform 版本： 2.0.41

空心效果控制和（true），该效果令对象透明填充，可以看到其文档背景的颜色。默认值为 false（无空心效果）。

shadowQuality 特性

shadowQuality:Number [读-写]

Scaleform 版本： 2.0.41

控制阴影或者滤光的品质。与 Flash 不同，唯一的设置值为 1 和 2。值为 1 表示低品质，2 或者更大值表示高品质。

shadowStrength 特性

shadowStrength:Number [读-写]

Scaleform 版本： 2.0.41

控制阴影模糊强度。有效值范围为 0-15.9（浮点数）。

默认值为 0。

可参考：
shadowBlurX
shadowBlurY

shadowStyle 特性

shadowStyle:String [读-写]

Scaleform 版本： 1.1.31

与 shadowColor 相结合，控制文本域的阴影渲染。shadowStyle 格式字符串描述了文本层的设置，与应用对象相匹配。例如：

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

字符“s”界定了阴影图层采用 shadowColor 值的颜色表示，而字符“t”界定了文本层用标准的 TextField.textColor 颜色值表示。这些特性在每个文本层将会被使用并渲染；但相同的转换应用到文本，每个单元映射到一个像素。如果“t”界定并且设置为“t（0，0），则忽略其设置效果。

请注意当前的实现当中使用阴影层，由于多次渲染将产生额外的棱角和粗糙感。强烈建议测量性能结果并尽可能得限制阴影的使用。

可参考：

```
shadowColor
TextField.textColor
```

7.6 图像替换

本节介绍了图像替换的控制方法。

setImageSubstitutions () 方法

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

Scaleform 版本： 2.0.38

设置文本域图像替换链接。
只有在图像插入到 SWF 时字符串链接替换才有效；这些图像还需要匹配链接以输出名字。插入图像到 SWF 时需要用到：

1. 导入位图图像到库。
2. 右键点击（Windows 系统）或者控制键点击（Macintosh 系统）位于库中的图像，从菜单目录中选择链接项。
3. 选择 ActionScript 输出，在第一帧中输出并在标识文本框中输入其名字（例如，myImage）。
4. 点击 OK 确定链接符。

在图像导入链接符分配完成后，需要创建一个 BitmapData 实例。下面为该实例的 ActionScript 代码示例：

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

注意：不要遗忘导入声明(导入 `import flash.display.BitmapData`;或使用完整名称 - `flash.display.BitmapData`)，不然导致结果“未定义”！

如果需要使用多个图像做替换，需要为每个图像重复这些步骤，给出不同的链接 ID。

单个替换对象设置如下：

```
subString:String
```

定义替换图像的子链接；这个为必要项。最大子链长度为 15 个字符。

`image : BitmapData`
指定图像；必要项。

`width : Number`
定义屏幕中的图像显示宽度，以像素为单位。可选项。

`height : Number`
定义屏幕中的图像显示高度，以像素为单位。可选项。

`baseLineY : Number`
定义图像基准线的 Y 坐标偏移量，以原始图像的像素为单位（w/o 转换）。可选项。默认情况下，该值与图像高度一致，因此，图像底部正好位于基准线上。

`id : String`
指定替换 ID 作为“updateImageSubstitution ”调用的的第一个参数。可选项。

“`substInfoArr`”应该为这些对象的序列，如同“`substInfo`”应该为对象的实例一样。版本 `setImageSubstitutions(substInfo:Object)`只支持单个替换，而版本 `setImageSubstitutions(substInfoArr:Array)`可设置多个。

注意，每个 `setImageSubstitutions` 的调用增加内部列表的替换。清楚这些元素需要调用 `setImageSubstitutions(null)`。

`setImageSubstitutions` 函数调用后，在 ActionScript 中必须保持一个到替换序列或者单个描述对象的索引；无论如何，需要保持这个索引，由于无法将替换序列恢复到文本域中，在 ActionScript 的其他地方需要引用则需要这个索引。

参数

`substInfoArr:Array` – 描述符对象替换序列（见上）。
`substInfo:Object` – 单个描述符对象替换（见上）。
`null` – 清除所有替换。

可参考：
`updateImageSubstitution()`

示例：

```
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);
```

只要文本域中包含子链“=）”，当并不应用，该子链将被名为“smile1”图像链接符所替换。

updateImageSubstitution () 方法

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

Scaleform 版本： 2.0.38

替换或删除原先由 `setImageSubstitutions` 函数为文本替换创建的图像。

参数

`id:String` –	替换 ID 号，与 `setImageSubstitutions` 函数调用使用的描述符对象 ID 成员相同。
`image:BitmapData` –	指定新图像；若为 null 则完全删除替换。

可参考：
`setImageSubstitutions()`

示例：

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

以下为插入图像的动画示例。在 onEnterFrame 句柄或使用 setInterval 可作更新操作。注意，`updateImageSubstitution` 调用时不会发生文本重新格式化；因此，新图像的大小应该与原先图像保持一致。

```
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 支持

本节介绍扩展 IME 支持。

disableIME 特性

`disableIME:Boolean [读-写]`

Scaleform 版本： 2.1.50

若设为 true，之前 IME 将在当前文本域有效。

默认值为 false。

getIMECompositionStringStyle () 方法

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

Scaleform 版本： 2.1.50

返回 IME 颜色设置种类中的颜色设置描述符对象。参阅 `setIMECompositionStringStyle` 获得 cateforyName 及其描述符对象的详细描述信息。

参数

categoryName:String – IME 颜色设置类别（查看 `setIMECompositionStringStyle` ）。

返回值

对象- 颜色设置目录描述符（查看 `setIMECompositionStringStyle` ）。

可参考：

`setIMECompositionStringStyle`
`disableIME`

setIMECompositionStringStyle () 方法

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

Scaleform 版本： 2.1.50

设置 IME 颜色设置类别类型。categoryName 参数可有如下值：

"compositionSegment" – 设置整个字符的颜色属性；
"clauseSegment" – 设置子段颜色属性；
"convertedSegment" – 设置转换文本段颜色属性；
"phraseLengthAdj" – 设置惯用语长度调整颜色属性；
"lowConfSegment" – 设置有错字符颜色属性。

styleDescriptor 为一个对象实例，有以下选项：

textColor : Number
文本颜色。

backgroundColor : Number
背景色。

underlineColor : Number
下划线颜色。

underlineStyle : String
下划线类型。有效值：
“single”
“thick”
“dotted”
“ditheredSingle”
“ditheredThick”

所有颜色的表示格式为：0xRRGGBB ，其中 RR-红色十六进制表示数[0..255]，BB-蓝色十六进制表示数[0..255]，GG-绿色十六进制表示数[0..255]。

参数

categoryName:String – IME 颜色设置类别（见描述信息）。
styleDescriptor:Object – 颜色设置描述符对象（见描述信息）。

可参考：

getIMECompositionStringStyle
disableIME

8 扩展 TextFormat 类

TextFormat 类表示字符格式信息，如颜色、字体大小、下划线等。

alpha 特性

`alpha:Number  [读-写]`

Scaleform 版本： 2.0.41

按照比例控制文本 alpha 透明度。有效值范围为 0-100%。标准 Flash `TextFormat.color` 不允许设置 alpha 透明度，因此，这个扩展功能不允许用来设置文本为部分或者完全透明。

可参考：

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

9 扩展 Stage 类

除了支持标准 Stage 类特性，Scaleform 引入了扩展功能增强了场景维度跟踪。扩展属性包括“visibleRect”，“safeRect”和“originalRect”，同时还提供了“onResize”额外参数，用来表示当前可视帧形状。详情见下。

注意，在 ActionScript 1.0 中可以直接引用扩展属性，如 Stage.visibleRect。而 ActionScript 2.0 中不能直接引用。为减轻 ActionScript 2.0 编译器负担，应该选用其他访问语法如：Stage["visibleRect"]。

visibleRect 特性

visibleRect:flash.geom.Rectangle [读-写]

Scaleform 版本： 2.2.56

本特性包含了当前可视矩形。该矩形会根据纵横比、缩放模式、对齐和 Viewport 缩放等参数的改变而改变，并且需要知道当前可视对象。该矩形具有负 topLeft 位置值。

可参考：

- safeRect
- originalRect

safeRect 特性

safeRect:flash.geom.Rectangle [读-写]

Scaleform 版本： 2.2.56

本特性包含了当前安全矩形设置信息。应该由用户使用 GFx::Movie::SetSafeRect 来设置。若无设置，则与“visibleRect”包含相同矩形。

可参考：

- visibleRect
- originalRect

originalRect 特性

```
originalRect:flash.geom.Rectangle  [读-写]
```

Scaleform 版本： 2.2.56

该特性总是包含原始 SWF 分辨率的 SWF 原始矩形。因此，若 Flash Studio 中“Document properties”分辨率设置为（600px×400px），则该矩形包含值为{ 0, 0, {600, 400} }。

可参考：

- `visibleRect`
- `safeRect`

onResize 事件句柄

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

Scaleform 版本： 2.2.56

onResize 事件句柄拥有一个扩展参数。该参数描述了当前可视矩形，该矩形与 `Stage.visibleRect` 返回的扩展特性相同。

可参考：

- `visibleRect`

translateToScreen() 方法

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

Scaleform 版本： 3.3.84

此方法返回 Stage 坐标系中的映射到屏幕坐标系的一个点。

参数

`pt:Object` – 具有代表要转换的点的坐标的 x 和 y 成员的一个 Object。可以使用 flash.geom.Point，而非通用 Object。

返回

点 – 映射到屏幕坐标系的输入点。

10 数组类扩展

指定分类排序

Flash 版本中 Array.sort 和 Array.sortOn 根据 Unicode 值来进行分类排序。但是，这样排序结果经常不对，特别是用在法语、西班牙语、德语等。这些语言中使用 Unicode 分类排序，所有的字符串字幕‘z’的后面都跟随音标符号（例如，元音符、重音符、强调）。因此，在 Flash 中排序数组[‘á’, ‘z’, ‘a’] 返回 [‘a’, ‘z’, ‘á’] 。但是，对于法语则返回为[‘a’, ‘á’, ‘z’]。

为解决这个问题，Scaleform 引入了另外一个选项 Array.sort/sortOn 方法：

Array.LOCALE:

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

也支持大小写敏感局部指定分类排序：

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

注释，本函数可能在特定平台上不能正确工作，有些平台内核不支持该函数。这种情况下 Array.LOCALE 分类即被当作按照普通 Unicode 值分类。

Scaleform 版本： 3.0.65

参考：

String.localeCompare

11 String 类扩展

localeCompare()方法

本函数比较两个或者更多字符串类型并以数字值的形式返回比较结果。本方法以局部指定方式来作比较。如果字符串相同，返回值为 0，如果原字符串中部分和参数中指定的比较字符串相同则返回一个负值，如果原字符串与参数中指定的比较字符串中部分内容相同，则返回一个正值。

见“数组类扩展”小节获得更多关于局部指定字符串操作信息。
注释，本函数可能在特定平台上不能正确工作，可能有些平台内核不支持该函数。这种情况下，本函数即被当作 Unicode 值的常规字符串比较函数。

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

Scaleform 版本： 3.0.65

参数

`other:String` –	比较字符串 。
`caseInsensitive:Boolean` –	一个可选参数可被用来做忽略大小字母比较。如果参数未被指定，则默认为大小写敏感比较。

参考:

`Array.sort`

12 视频类扩展

在标准的 Flash 中只有一个视频对象可以每次从 NetStream 对象收到数据。Scaleform 取消了这项限制并允许多个视频对象附加到 NetStream 对象，可以接受相同视频数据。

clipRect 属性

`clipRect: flash.geom.Rectangle  [读-写]`

Scaleform 版本： 3.0.70

该属性允许视频对象可以只显示原始视频帧的一部分（区域）。

例如：

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

13 3Di Extensions

Scaleform 3Di 是使用一组内置到 Scaleform 中的基本 AS2 扩展实现的。为 3Di 增加了下列 ActionScript 扩展：

Scaleform 版本： 3.2.82

_z

设置对象的 Z 坐标（深度），默认值为 0。

_zscale

将 Z 向的对象伸缩设置一个百分比，默认值为 100。

_xrotation

设置对象围绕 X（横向）轴旋转，默认值为 0。

_yrotation

设置对象围绕 Y（纵向）轴旋转，默认值为 0。

_matrix3d

使用一组 16 个浮点数（4 x 4 矩阵）设置对象的完整 3D 转换。如果此值设置为 NULL，就会删除所有 3D 转换，而且对象将会变为 2D。

_perspfov

在对象上设置透视视野（Field Of View) 角度，有效角度必须 > 0 度和 < 180 度。如果不设置，对象就继承根 FOV 角，该角默认为 55 度。

使用这些新扩展需要在 ActionScript 中把全局变量 `gfxExtensions` 设置为 True（真）。

14 全局扩展

noInvisibleAdvance 特性

```
noInvisibleAdvance : Boolean
```

Scaleform 版本：2.1.52

若设置为 `true`，本特性关闭所有隐藏视频剪辑的前进动作。这将提高拥有大量隐藏视频剪辑的 SWF 性能。注意，Flash 中隐藏视频剪辑也有前进动作（将仍然执行时间线动画，调用各帧的 ActionScript 脚本等）。因此，将此特性设置为“`true`”，可以在 Scaleform 和 Flash 中产生不同的效果。

可参考：

`MovieClip.noAdvance`

Imecommand() 函数

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

Scaleform 版本： 2.1.50

本函数类似于 fscommand，但只用来与 Scaleform IME 执行器交互数据。在 Scaleform 内部使用。

getIMECandidateListStyle() 函数

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

Scaleform 版本： 2.1.50

返回 IME 选择列表的颜色设置描述符。查看 `setIMECandidateListStyle` 获得更多的描述符对象信息。

返回

Object – 文字候选列表颜色设置描述器。

可参考：

`setIMECandidateListStyle`
`TextField.disableIME`

setIMECandidateListStyle() 函数

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

Scaleform 版本： 2.1.50

设置候选列表类型作为对象传递参数。

styleDescriptor 为一个对象实例，有以下选项：

```
      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
```
      文本颜色。

```
selectedTextColor : Number
```
      已选择文本颜色。

```
fontSize : Number
```
      行中字体大小。

```
backgroundColor : Number
```
      未选择行中文本的背景颜色。

```
seletedTextBackgroundColor : Number
```
      选择行中文本颜色。

```
indexBackgroundColor : Number
```
      未选择行中指针背景颜色。

```
selectedIndexBackgroundColor : Number
```

选择行中指针背景颜色。

`readingWindowTextColor : Number`
阅览窗中文本颜色。

`readingWindowBackgroundColor : Number`
阅览窗中背景颜色。

`readingWindowFontSize : Number`
阅览窗中文本字体大小。
所有颜色的描述格式为：0xRRGGBB ， 其中 RR - 红色十六进制表示数[0,255]，BB - 蓝色十六进制表示数[0,255]，GG - 绿色十六进制表示数[0,255]。

可参考：
`getIMECandidateListStyle`
`TextField.disableIME`

15 标准方法和事件处理程序扩展

Key 类

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

Key 类方法取一个用于键盘/控制器的可选参数。如果没有指定，则 `controllerIdx` 将默认为 0。

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

Key 侦听器方法 `onKeyDown` 可接收一个用于生成该事件的键盘/控制器的额外参数。

Selection 类

```
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)
```

Selection 类方法取一个用于键盘/控制器的可选参数。如果没有指定，则 `controllerIdx` 将默认为 0。

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

Selection 侦听器方法 `onSetFocus` 可接收用于生成该事件的键盘/控制器的一个额外的参数。

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)
```

这些 MovieClip/Button/TextField 侦听器方法可接收用于生成该事件的键盘/控制器的一个额外的参数。

System.capabilities

`System.capabilities.numControllers` – 返回系统中检测到的控制器的数量。