

ACTIONSCRIPT® 3.0 コンポーネントガイド

法律上の注意

法律上の注意については、http://help.adobe.com/ja_JP/legalnotices/index.htmlを参照してください。

目次

第 1 章：はじめに

対象読者	1
システム要件	1
本マニュアルについて	1
表記規則	2
用語	2
その他のリソース	2

第 2 章：ActionScript 3.0 コンポーネントについて

コンポーネントを使用するメリット	3
コンポーネントタイプ	4
ドキュメントに対する追加と削除	6
コンポーネントのバージョンの確認	8
ActionScript 3.0 のイベント処理モデル	9
単純なアプリケーション	9

第 3 章：コンポーネントの利用

コンポーネントアーキテクチャ	16
コンポーネントファイルの操作	18
コンポーネントアプリケーションのデバッグ	20
パラメーターとプロパティの設定	20
ライブラリ	22
コンポーネントのサイズ変更	22
ライブプレビュー	22
イベント処理	23
表示リストの操作	24
FocusManager の操作	26
リストベースのコンポーネントの操作	28
DataProvider の使用	28
CellRenderer の操作	35
コンポーネントをアクセス可能にする	42

第 4 章：UI コンポーネントの使用

Button コンポーネントの使用	43
CheckBox コンポーネントの使用	45
ColorPicker コンポーネントの使用	48
ComboBox コンポーネントの使用	50
DataGrid コンポーネントの使用	53
Label コンポーネントの使用	58

目次

List コンポーネントの使用	61
NumericStepper コンポーネントの使用	64
ProgressBar コンポーネントの使用	67
RadioButton コンポーネントの使用	72
ScrollPane コンポーネントの使用	75
Slider コンポーネントの使用	78
TextArea コンポーネントの使用	81
TextInput コンポーネントの使用	84
TileList コンポーネントの使用	87
UILoader コンポーネントの使用	89
UIScrollBar コンポーネントの使用	90
第 5 章：UI コンポーネントのカスタマイズ	
UI コンポーネントのカスタマイズについて	93
スタイルの設定	93
スキンについて	96
Button コンポーネントのカスタマイズ	98
CheckBox コンポーネントのカスタマイズ	100
ColorPicker コンポーネントのカスタマイズ	102
ComboBox コンポーネントのカスタマイズ	103
DataGrid コンポーネントのカスタマイズ	105
Label コンポーネントのカスタマイズ	109
List コンポーネントのカスタマイズ	110
NumericStepper コンポーネントのカスタマイズ	113
ProgressBar コンポーネントのカスタマイズ	114
RadioButton コンポーネントのカスタマイズ	116
ScrollPane コンポーネントのカスタマイズ	118
Slider コンポーネントのカスタマイズ	118
TextArea コンポーネントのカスタマイズ	120
TextInput コンポーネントのカスタマイズ	122
TileList コンポーネントのカスタマイズ	123
UILoader コンポーネントのカスタマイズ	125
UIScrollBar コンポーネントのカスタマイズ	125
第 6 章：FLVPlayback コンポーネントの使用	
FLVPlayback コンポーネントの使用	128
FLVPlayback コンポーネントのカスタマイズ	146
SMIL ファイルの使用	156
第 7 章：FLVPlayback キャプションコンポーネントの使用	
FLVPlaybackCaptioning コンポーネントの使用	164
Timed Text キャプションの使用	166
キャプションでのキューポイントの使用	171

キャプションを使用した複数の FLV ファイルの再生	173
FLVPlaybackCaptioning コンポーネントのカスタマイズ	173

第1章：はじめに

Adobe® Flash® CS5 Professional は、インパクトの強い Web 体験を生み出すための本格的な標準オーサリングツールです。コンポーネントとは、そのような体験を生むリッチなインターネットアプリケーションを構築する構成要素です。コンポーネントは、Flash でのオーサリング中、または実行時に Adobe® ActionScript® メソッド、プロパティおよびイベントを使用してコンポーネントをカスタマイズできるパラメーターを持つムービークリップです。開発者はコンポーネントを通じてコードを再利用および共有し、複雑な機能をカプセル化することができます。デザイナーは ActionScript を使わずに、それらを利用してカスタマイズすることができます。

コンポーネントを使用することにより、一貫した外観と操作性を備えた堅牢なアプリケーションをすばやく簡単に作成できます。本マニュアルでは、Adobe ActionScript 3.0 コンポーネントを使用してアプリケーションを作成する方法について説明します。『Adobe® ActionScript® 3.0 言語およびコンポーネントリファレンス』では、各コンポーネントのアプリケーションプログラミングインターフェイス (API) について説明します。

Adobe® が作成したコンポーネントを使用することも、他の開発者が作成したコンポーネントをダウンロードすることも、独自のコンポーネントを作成することもできます。

対象読者

本マニュアルは、Flash アプリケーションを作成しており、コンポーネントを使って開発を効率化したいと考えている開発者向けに書かれています。このため、Flash でのアプリケーションの開発と ActionScript の記述について既に知識があることが望まれます。

ActionScript を作成した経験が少ない方でも、コンポーネントをドキュメントに追加し、プロパティインスペクターやコンポーネントインスペクターパネルでパラメーターを設定し、ビヘイビアパネルでイベントを操作することができます。例えば、「Web ページへ移動」ビヘイビアを Button コンポーネントに追加すると、ActionScript コードをいっさい記述しなくても、ボタンがクリックされると Web ブラウザーの中で URL を開くことができます。

作成するアプリケーションの堅牢性を高めるには、コンポーネントを動的に作成します。ActionScript を使用して、実行時にプロパティを設定し、メソッドを呼び出します。また、イベントリスナーモデルを使用して、イベントを処理します。

詳しくは、16 ページの「[コンポーネントの利用](#)」を参照してください。

システム要件

Flash コンポーネントには、Flash のシステム要件を超えるシステム要件はありません。

Flash CS3 以降のコンポーネントを使用する SWF ファイルは、Adobe® Flash® Player 9.0.28.0 以降で表示する必要があります。さらに、ActionScript 3.0 を使用してパブリッシュする必要があります (ファイル/パブリッシュ設定を選択して、「Flash」タブで設定できます)。

本マニュアルについて

本マニュアルでは、コンポーネントを使った Flash アプリケーションの開発について詳しく説明します。本マニュアルは、Flash および ActionScript 3.0 の全般的知識がある読者を対象とします。Flash および関連製品についての特定のマニュアルは個別に提供されています。

本マニュアルは、PDF ファイルとオンラインヘルプの形式で提供されています。オンラインヘルプを表示するには、Flash を起動し、ヘルプ / Flash ヘルプ / Adobe ActionScript 3.0 コンポーネントガイドを選択します。

Flash の詳細については、次のマニュアルを参照してください。

- **Flash** ユーザーガイド
- **ActionScript 3.0** 開発ガイド
- **Adobe Flash Professional CS5** 用 **ActionScript 3.0** リファレンスガイド

表記規則

本マニュアルでは、次の表記規則を使用しています。

- **Italic font** (イタリック体のフォント) は、置き換える必要がある値 (フォルダーパス内の値など) を示します。
- **Code font** (コードフォント) は、メソッドやプロパティ名など、ActionScript コードを示します。
- **Code font italic** (イタリック体のコードフォント) は、置き換える必要があるコードアイテム (ActionScript パラメーターなど) を示します。
- **Bold font** (ボールドフォント) は、ユーザーが入力する値を示します。

用語

本マニュアルでは、次の用語を使用しています。

実行時 Flash Player でコードを実行しているときを指します。

オーサリング時 Flash オーサリング環境で作業しているときを指します。

その他のリソース

これらのマニュアルの内容に加えて、記事、デザイン案、サンプルを定期的に更新し、アドビデベロッパーセンターとアドビデザインセンターで提供しています。

その他のコンポーネントサンプルは www.adobe.com/go/learn_fl_samples_jp にあります。

アドビデベロッパーセンター

アドビデベロッパーセンターでは、ActionScript の最新情報、実際のアプリケーション開発に関する記事、新たに発生した重要な問題に関する情報を提供しています。デベロッパーセンターは www.adobe.com/go/flash_devcenter_jp で閲覧できます。

アドビデザインセンター

最新のデジタルデザインやモーショングラフィックを紹介しています。このサイトでは、最先端のアーティストによる作品を参照し、新しいデザインのトレンドを見つけることができます。また、チュートリアル、主要なワークフロー、高度な手法を参考にしてスキルを磨くこともできます。チュートリアルと記事、またインスピレーションが得られるギャラリーの作品を月に 2 回更新しています。デザインセンターは、www.adobe.com/go/fl_designcenter_jp で閲覧できます。

第 2 章: ActionScript 3.0 コンポーネントについて

Adobe® Flash® Professional CS5 コンポーネントとは、外観やビヘイビアを変更するためのパラメーターを持つムービークリップです。コンポーネントには、RadioButton や CheckBox などのシンプルなユーザーインターフェイスコントロールから、List や DataGrid のように内容が含まれるものまであります。

コンポーネントを使用することにより、一貫したビヘイビアと外観を備えた堅牢な Flash アプリケーションをすばやく簡単に作成できます。カスタムボタン、コンボボックス、リストを作成するのではなく、これらのコントロールを実装する Flash コンポーネントを利用できます。コンポーネントパネルからこれらのコンポーネントをアプリケーションドキュメントにドラッグするだけです。これらのコンポーネントの外観は、アプリケーションのデザインに合わせて簡単にカスタマイズできます。

このようなすべての操作は ActionScript に精通していない場合でも実行できますが、ActionScript 3.0 を使用すると、コンポーネントのビヘイビアの変更や、新しいビヘイビアの実装もできます。各コンポーネントには、そのアプリケーションプログラミングインターフェイス (API) を構成する ActionScript メソッド、プロパティおよびイベントの一意のセットがあります。API を使用すると、アプリケーションの実行中にコンポーネントを作成し、操作できます。

API を使用して、独自の新しいカスタムコンポーネントを作成することもできます。また、Flash コミュニティのメンバーが作成したコンポーネントを、Adobe Exchange (www.adobe.com/go/flash_exchange_jp) からダウンロードすることもできます。コンポーネントの作成について詳しくは、www.adobe.com/go/learn_fl_creating_components_jp を参照してください。

ActionScript 3.0 コンポーネントのアーキテクチャには、すべてのコンポーネントの基礎となるクラス、外観のカスタマイズに使用できるスキンとスタイル、イベント処理モデル、フォーカス管理、アクセシビリティインターフェイスなどが含まれます。

注意: Adobe Flash CS5 には、ActionScript 3.0 コンポーネントに加えて ActionScript 2.0 コンポーネントも含まれています。これらの 2 つのコンポーネントセットを混合することはできません。1 つのアプリケーションにはいずれかのセットを使用する必要があります。Flash CS5 は、開いているファイルが ActionScript 2.0 であるか、ActionScript 3.0 であるかに応じて、ActionScript 2.0 コンポーネントと ActionScript 3.0 コンポーネントのいずれか一方を表示します。新しい Flash ドキュメントを作成する場合は、Flash ファイル (ActionScript 3.0) と Flash ファイル (ActionScript 2.0) のいずれかを指定する必要があります。既存のドキュメントを開くと、Flash は「パブリッシュ設定」を確認し、使用するコンポーネントセットを決定します。ActionScript 2.0 コンポーネントの詳細については、『Adobe® ActionScript® 2.0 コンポーネントガイド』を参照してください。

Flash ActionScript 3.0 コンポーネントの一覧について詳しくは、4 ページの「[コンポーネントタイプ](#)」を参照してください。

コンポーネントを使用するメリット

コンポーネントを使用すると、アプリケーションの設計プロセスとコーディングプロセスを切り離すことができます。開発者は、コンポーネントによって、デザイナーがアプリケーションで使用できる機能を作成できます。開発者は、使用頻度の高い機能をコンポーネントにカプセル化でき、デザイナーは、コンポーネントのパラメーターを変更することにより、コンポーネントのサイズ、位置、およびビヘイビアをカスタマイズできます。グラフィカルエレメントやスキンを編集して、コンポーネントの外観を変更することもできます。

コンポーネントは、スタイル、スキン、フォーカス管理などの中心的機能を共有しています。最初のコンポーネントをアプリケーションに追加すると、この中心的機能が約 20 キロバイトのサイズを占めます。別のコンポーネントを追加すると、追加されたコンポーネントにより最初のメモリ割り当てが共有されるので、アプリケーションのサイズ増加が抑えられます。

この節では、ActionScript 3.0 コンポーネントのメリットをいくつか説明します。

ActionScript 3.0 の優れた機能 により、Flash Player の機能を向上するための重要な手段であるパワフルなオブジェクト指向プログラミング言語を利用できます。この言語は、再利用可能なコードに基づいてリッチなインターネットアプリケーションを作成できるように設計されています。ActionScript 3.0 は、世界標準のスクリプト言語である ECMAScript に基づいており、ECMAScript (ECMA-262) edition 3 言語仕様に準拠しています。ActionScript 3.0 について詳しくは、『[ActionScript 3.0 開発ガイド](#)』を参照してください。この言語に関するリファレンスについては、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」を参照してください。

FLA ベースのユーザーインターフェイスコンポーネント を使用すると、スキンに容易にアクセスでき、オーサリング時に簡単にカスタマイズできます。これらのコンポーネントは、実行時にコンポーネントの外観のさまざまな側面をカスタマイズし、スキンをロードできるスキンスタイルなどのスタイルも提供します。詳しくは、93 ページの「[UI コンポーネントのカスタマイズ](#)」および「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」を参照してください。

新規の FVLPlayback コンポーネントにより、FLVPlaybackCaptioning コンポーネント が、フルスクリーンサポート、ライブプレビューの向上、色やアルファ設定を追加できるスキン、FLV ダウンロードおよびレイアウト機能の向上と共に追加されます。

プロパティインスペクターおよびコンポーネントインスペクター を使用すると、Flash でオーサリング時にコンポーネントパラメーターを変更できます。詳しくは、18 ページの「[コンポーネントファイルの操作](#)」および 20 ページの「[パラメーターとプロパティの設定](#)」を参照してください。

新規コレクションダイアログボックス (ComboBox、List および TileList コンポーネント用) を使用すると、ユーザーインターフェイスを通じてその dataProvider プロパティに値を指定できます。詳しくは、28 ページの「[DataProvider の作成](#)」を参照してください。

ActionScript 3.0 イベントモデル を使用すると、アプリケーションはイベントをリスンし、イベントハンドラーを呼び出して応答できます。詳しくは、9 ページの「[ActionScript 3.0 のイベント処理モデル](#)」および 23 ページの「[イベント処理](#)」を参照してください。

Manager クラス を使用すると、アプリケーションのフォーカスの処理とスタイルの管理が簡単になります。詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」を参照してください。

UIComponent 基本クラス は、このクラスを展開するコンポーネントにコアメソッド、プロパティおよびイベントを提供します。すべての ActionScript 3.0 ユーザーインターフェイスコンポーネントは、UIComponent クラスから継承されます。詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の UIComponent クラスを参照してください。

SWC を UI FLA ベースコンポーネントで使用すると、コンポーネントのタイムライン内に ActionScript 定義がアセットとして提供され、コンパイル時間が短縮されます。

ActionScript 3.0 によってクラス階層の拡張が容易 になり、一意の名前空間の作成や、必要に応じたクラスの読み込みが可能になりました。また、容易にサブクラスを作成してコンポーネントを拡張できます。

詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」を参照してください。

注意：Flash CS5 は、FLA ベースのコンポーネントと SWC ベースのコンポーネントの両方をサポートしています。詳しくは、16 ページの「[コンポーネントアーキテクチャ](#)」を参照してください。

コンポーネントタイプ

Flash コンポーネントは、Flash CS5 のインストール時にインストールします。

ActionScript 3.0 コンポーネントには次のユーザーインターフェイス (UI) コンポーネントが含まれます。

Button	List	TextArea
CheckBox	NumericStepper	TextInput
ColorPicker	RadioButton	TileList
ComboBox	ProgressBar	UILoader
DataGrid	ScrollPane	UIScrollBar
Label	Slider	

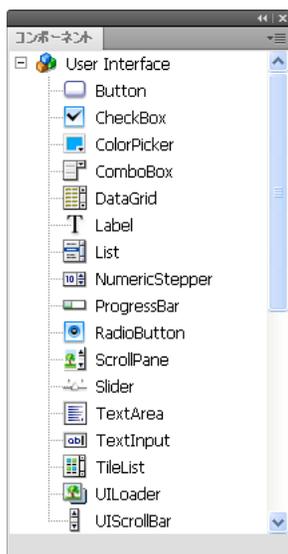
ユーザーインターフェイスコンポーネントに加えて、Flash ActionScript 3.0 コンポーネントには次のコンポーネントとサポートクラスが含まれます。

- FLVPlayback コンポーネント (fl.video.FLVPlayback)。これは SWC ベースのコンポーネントです。
FLVPlayback コンポーネントを使用すると、Flash アプリケーションにビデオプレーヤーを追加して、Adobe® Flash® Video Streaming Service (FVSS) や Adobe の Macromedia® Flash® Media Server (FMS) によりプログレッシブビデオストリーミングを HTTP 上で簡単に再生できます。詳しくは、128 ページの「[FLVPlayback コンポーネントの使用](#)」を参照してください。
- FLVPlayback Custom UI コンポーネント。これは FLA ベースのコンポーネントで、ActionScript 2.0 と ActionScript 3.0 の両方のバージョンの FLVPlayback コンポーネントに使用できます。詳しくは、128 ページの「[FLVPlayback コンポーネントの使用](#)」を参照してください。
- FLVPlayback Captioning コンポーネント。FLVPlayback にクローズドキャプションを提供します。164 ページの「[FLVPlayback キャプションコンポーネントの使用](#)」を参照してください。
ActionScript 3.0 コンポーネントとそのサポートクラスの一覧については、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」を参照してください。

Flash コンポーネントの表示

次の手順に従って Flash ActionScript 3.0 コンポーネントをコンポーネントパネルに表示できます。

- 1 Flash を起動します。
- 2 新しい Flash ファイル (ActionScript 3.0) を作成するか、「パブリッシュ設定」で「ActionScript 3.0」が指定された既存の Flash ドキュメントを開きます。
- 3 コンポーネントパネルが開いていない場合は、ウィンドウ/コンポーネントを選択して開きます。



ユーザーインターフェイスコンポーネントのあるコンポーネントパネル

Adobe Exchange (www.adobe.com/go/flash_exchange_jp) から追加コンポーネントをダウンロードすることもできます。Exchange からダウンロードしたコンポーネントをインストールするには、www.adobe.com/go/exchange_jp から Adobe® Extension Manager をダウンロードしてインストールします。「Adobe Exchange home」リンクをクリックし、「Extension Manager」リンクを探します。

コンポーネントはすべて、Flash のコンポーネントパネルに表示できます。Windows® または Macintosh® コンピューターにコンポーネントをインストールするには、次の手順を実行します。

Windows ベースのコンピューターまたは Macintosh コンピューターへのコンポーネントのインストール

- 1 Flash を終了します。
- 2 コンポーネントが含まれる SWC ファイルまたは FLA ファイルをハードディスクの次のフォルダーに配置します。
 - Windows :
C:\Program Files\Adobe v Adobe Flash CS5\language\configuration\Components
 - Mac OS :
Macintosh HD/ アプリケーション /Adobe Flash CS5/Configuration/Components
- 3 Flash を起動します。
- 4 コンポーネントパネルが開いていない場合は、ウィンドウ/コンポーネントを選択し、コンポーネントを表示します。
コンポーネントファイルについては詳しくは、18 ページの「[コンポーネントファイルの操作](#)」を参照してください。

ドキュメントに対する追加と削除

FLA ベースのコンポーネントをコンポーネントパネルからステージにドラッグすると、編集可能なムービークリップがライブラリに読み込まれます。SWC ベースのコンポーネントをステージにドラッグすると、コンパイルされたクリップがライブラリに読み込まれます。コンポーネントをライブラリに読み込むと、そのインスタンスを、ライブラリパネルまたはコンポーネントパネルからステージにドラッグできます。

オーサリング時のコンポーネントの追加

コンポーネントは、コンポーネントパネルからドラッグすることでドキュメントに追加できます。コンポーネントの各インスタンスのプロパティは、プロパティインスペクターや、コンポーネントインスペクターの「パラメーター」タブで設定できます。

- 1 ウィンドウ／コンポーネントを選択します。
- 2 コンポーネントパネルでコンポーネントをダブルクリックするか、コンポーネントをステージにドラッグします。
- 3 ステージでコンポーネントを選択します。
- 4 プロパティインスペクターが表示されない場合は、ウィンドウ／プロパティ／プロパティを選択します。
- 5 プロパティインスペクターで、コンポーネントインスタンスのインスタンス名を入力します。
- 6 ウィンドウ／コンポーネントインスペクターを選択し、「パラメーター」タブを選択してインスタンスのパラメーターを指定します。

詳しくは、20 ページの「[パラメーターとプロパティの設定](#)」を参照してください。

- 7 幅 (W:) と高さ (H:) の値を編集することにより、必要に応じてコンポーネントのサイズを変更します。

特定のコンポーネントタイプのサイズ変更について詳しくは、93 ページの「[UI コンポーネントのカスタマイズ](#)」を参照してください。

- 8 制御／ムービープレビューを選択するか、または Ctrl+Enter キーを押して、ドキュメントをコンパイルし、設定の結果を表示します。

スタイルプロパティを設定してコンポーネントの色やテキストフォーマットを変更することも、コンポーネントのスキンを編集して外観をカスタマイズすることもできます。これらのトピックについて詳しくは、93 ページの「[UI コンポーネントのカスタマイズ](#)」を参照してください。

オーサリング時にコンポーネントをステージにドラッグする場合は、そのインスタンス名 (fmyButton など) を使用してコンポーネントを参照できます。

ActionScript を使用した実行時のコンポーネントの追加

ActionScript を使用して実行時にコンポーネントをドキュメントに追加するには、SWF ファイルのコンパイル時に、最初にコンポーネントをアプリケーションのライブラリに配置する必要があります (ウィンドウ／ライブラリ)。コンポーネントをライブラリに追加するには、コンポーネントパネルからライブラリパネルにコンポーネントをドラッグします。ライブラリについて詳しくは、22 ページの「[ライブラリ](#)」を参照してください。

また、コンポーネントのクラスファイルを読み込んで、その API をアプリケーションで使用できるようにする必要があります。コンポーネントクラスファイルは、1 つまたは複数のクラスが含まれるパッケージにインストールされます。コンポーネントクラスを読み込むには、import ステートメントを使用し、パッケージ名とクラス名を指定します。例えば、次の import ステートメントを使用して Button クラスを読み込みます。

```
import fl.controls.Button;
```

コンポーネントが含まれているパッケージについて詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」を参照してください。コンポーネントのソースファイルの場所について詳しくは、18 ページの「[コンポーネントファイルの操作](#)」を参照してください。

コンポーネントのインスタンスを作成するには、コンポーネントの ActionScript コンストラクターメソッドを呼び出す必要があります。例えば、次のステートメントによって、aButton という Button のインスタンスが作成されます。

```
var aButton:Button = new Button();
```

最後の手順は、静的な addChild() メソッドを呼び出して、コンポーネントインスタンスをステージまたはアプリケーションコンテナに追加します。例えば、次のステートメントは aButton インスタンスを追加します。

```
addChild(aButton);
```

この時点でコンポーネントの API を使用して、コンポーネントのサイズとステージでの位置を動的に指定し、イベントをリスンし、そのビヘイビアを変更するプロパティを設定できます。特定のコンポーネントの API について詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」を参照してください。

addChild() メソッドについて詳しくは、24 ページの「[表示リストの操作](#)」を参照してください。

コンポーネントの削除

オーサリング時にコンポーネントインスタンスをステージから削除するには、そのインスタンスを選択し、Delete キーを押すだけです。その結果、インスタンスはステージから削除されますが、コンポーネントがアプリケーションから削除されることはありません。

ステージまたはライブラリに配置した後にコンポーネントを Flash ドキュメントから削除するには、そのコンポーネントと、関連付けられたアセットをライブラリから削除する必要があります。ステージからコンポーネントを削除するだけでは十分ではありません。ライブラリから削除しないと、コンパイルしたときにそのコンポーネントはアプリケーションに含まれます。

- 1 ライブラリパネルで、そのコンポーネントのシンボルを選択します。
- 2 ライブラリパネルの下部にある「削除」ボタンをクリックするか、ライブラリパネルのメニューから「削除」を選択します。

この手順を繰り返して、そのコンポーネントに関連付けられたすべてのアセットを削除します。

アプリケーションの実行時にコンテナからコンポーネントを削除する方法について詳しくは、26 ページの「[表示リストからのコンポーネントの削除](#)」を参照してください。

コンポーネントのバージョンの確認

Flash ActionScript 3.0 コンポーネントには、Adobe テクニカルサポートに提供する必要がある場合、または使用しているコンポーネントのバージョンを確認する必要がある場合に表示できるバージョンプロパティがあります。

ユーザーインターフェイスコンポーネントのバージョン番号の表示

- 1 新しい Flash ファイル (ActionScript 3.0) ドキュメントを作成します。
- 2 コンポーネントをステージにドラッグし、インスタンス名を指定します。例えば、ComboBox をステージにドラッグし、名前を aCb と指定します。
- 3 **F9** キーを押すか、ウィンドウ/アクションを選択してアクションパネルを開きます。
- 4 メインタイムラインのフレーム 1 をクリックし、アクションパネルに次のコードを追加します。

```
trace(aCb.version);
```

次の図のようなバージョン番号が出力パネルに表示されます。

FLVPlayback および FLVPlaybackCaptioning コンポーネントでは、インスタンス名ではなくクラス名を参照する必要があります。これは、バージョン番号がクラス定数に保存されているからです。

FLVPlayback および FLVPlaybackCaptioning コンポーネントのバージョン番号の表示

- 1 新しい Flash ファイル (ActionScript 3.0) ドキュメントを作成します。
- 2 FLVPlayback および FLVPlaybackCaptioning コンポーネントをライブラリパネルにドラッグします。
- 3 **F9** キーを押すか、ウィンドウ/アクションを選択してアクションパネルを開きます。

4 メインタイムラインのフレーム 1 をクリックし、アクションパネルに次のコードを追加します。

```
import fl.video.*;
trace("FLVPlayback.VERSION: " + FLVPlayback.VERSION);
trace("FLVPlaybackCaptioning.VERSION: " + FLVPlaybackCaptioning.VERSION);
```

バージョン番号が出力パネルに表示されます。

ActionScript 3.0 のイベント処理モデル

ActionScript 3.0 では、以前のバージョンの ActionScript で使用されていたさまざまなイベント処理メカニズムに代わる 1 つのイベント処理モデルが導入されました。新しいイベントモデルは、ドキュメントオブジェクトモデル (DOM) レベル 3 のイベント仕様に基づいています。

ActionScript 2.0 の `addListener()` メソッドを使用した経験があれば、ActionScript 2.0 のイベントリスナーモデルと ActionScript 3.0 のイベントモデルとの違いがよくわかります。次のリストで、この 2 つのイベントモデルの主な違いをいくつか説明します。

- ActionScript 2.0 でイベントリスナーを追加するには、`addListener()` を使用する場合と `addEventListener()` を使用する場合がありますが、ActionScript 3.0 では、すべての場合に `addEventListener()` を使用します。
- ActionScript 2.0 にはイベントフローの仕組みがないので、`addListener()` メソッドを呼び出すことができるのはイベントをブロードキャストするオブジェクトに対してのみでした。ActionScript 3.0 では、イベントフローに組み込まれているすべてのオブジェクトに対して `addEventListener()` メソッドを呼び出すことができます。
- ActionScript 2.0 では、イベントリスナーは関数、メソッド、オブジェクトのいずれかですが、ActionScript 3.0 では、関数とメソッドだけをイベントリスナーにできます。
- `on(event)` シンタックスが ActionScript 3.0 ではサポートされないので、ActionScript イベントコードをムービークリップに追加できません。イベントリスナーを追加するには、`addEventListener()` を使用します。

次の例では、`aButton` という名前の `Button` コンポーネントの `MouseEvent.CLICK` イベントをリスンします。これは、ActionScript 3.0 の基本的なイベント処理モデルを示しています。

```
aButton.addEventListener(MouseEvent.CLICK, clickHandler);
function clickHandler(event:MouseEvent):void {
    trace("clickHandler detected an event of type: " + event.type);
    trace("the event occurred on: " + event.target.name);
}
```

ActionScript 3.0 イベント処理について詳しくは、『ActionScript 3.0 のプログラミング』を参照してください。

ActionScript 3.0 のコンポーネントのイベント処理について詳しくは、23 ページの「[イベント処理](#)」を参照してください。

単純なアプリケーション

この節では、Flash コンポーネントと Flash オーサリングツールを使用して単純な ActionScript 3.0 アプリケーションを作成する手順について説明します。タイムラインに ActionScript コードを含む FLA ファイルと、ライブラリにコンポーネントのみを含む FLA ファイルを持つ外部 ActionScript クラスファイルの両方の例が用意されています。一般に、大規模なアプリケーションの開発は、クラスとアプリケーションの間でコードを共有できるように、また、アプリケーションの保守が簡単になるように、外部クラスファイルを使用しています。ActionScript 3.0 を使用したプログラミングについて詳しくは、『ActionScript 3.0 のプログラミング』を参照してください。

アプリケーションの設計

ActionScript コンポーネントアプリケーションの最初の例は標準的な "Hello World" アプリケーションなので、その設計は非常に簡単です。

- このアプリケーションを "Greetings" と呼びます。
- ここでは、`TextArea` を使用して挨拶 (最初は Hello World) を表示します。
- テキストの色を変更できる `ColorPicker` を使用します。
- テキストのサイズを小、大、または最大に設定できる 3 つの `RadioButton` を使用します。
- ドロップダウンリストからさまざまな挨拶を選択できる `ComboBox` を使用します。
- このアプリケーションでは、コンポーネントパネルのコンポーネントを使用し、ActionScript コードを使用してアプリケーションエレメントも作成します。

この定義を適切に設定したら、アプリケーションの作成を開始できます。

Greetings アプリケーションの作成

次の手順では、Flash オーサリングツールを使用して FLA ファイルを作成し、コンポーネントをステージ上に配置し、ActionScript コードをタイムラインに追加して、Greetings アプリケーションを作成します。

FLA ファイルでの Greetings アプリケーションの作成:

- 1 ファイル/新規を選択します。
- 2 新規ドキュメントダイアログボックスで「Flash ファイル (ActionScript 3.0)」を選択し、「OK」をクリックします。
新しい Flash ウィンドウが開きます。
- 3 ファイル/保存を選択し、Flash ファイルの名前を **Greetings.fla** と指定し、「保存」ボタンをクリックします。
- 4 Flash のコンポーネントパネルで `TextArea` コンポーネントを選択し、ステージにドラッグします。
- 5 プロパティウィンドウのステージで `TextArea` を選択し、インスタンス名として **aTa** と入力し、次の情報を入力します。
 - W 値 (幅) に **230** と入力します。
 - H 値 (高さ) に **44** と入力します。
 - X 値 (水平位置) に **165** と入力します。
 - Y 値 (垂直位置) に **57** と入力します。
 - 「パラメーター」タブで、テキストパラメーターとして **Hello World!** と入力します。
- 6 `ColorPicker` コンポーネントをステージ上にドラッグし、`TextArea` の左側に配置して、インスタンス名 **txtCp** を指定します。プロパティインスペクターに次の情報を入力します。
 - X 値として **96** と入力します。
 - Y 値として **72** と入力します。
- 7 `RadioButton` コンポーネントを一度に 1 つずつステージにドラッグし、インスタンス名 **smallRb**、**largerRb** および **largestRb** を指定します。プロパティインスペクターで、これらのコンポーネントに関する次の情報を入力します。
 - それぞれの W 値として **100**、H 値として **22** と入力します。
 - X 値として **155** と入力します。
 - **smallRb** の Y 値として **120**、**largerRb** に **148**、**largestRb** に **175** と入力します。
 - それぞれの `groupName` パラメーターに **fontRbGrp** と入力します。
 - 「パラメーター」タブで、それぞれのラベル **Small**、**Larger** および **Largest** を入力します。

8 ComboBox をステージまでドラッグし、インスタンス名 **msgCb** を指定します。プロパティインスペクターで、このコンポーネントに関する次の情報を入力します。

- W 値として **130** と入力します。
- X 値として **265** と入力します。
- Y 値として **120** と入力します。
- 「パラメーター」タブで、プロンプトパラメーターに **Greetings** と入力します。
- **dataProvider** パラメーターのテキストフィールドをダブルクリックして、値ダイアログボックスを開きます。
- プラス記号をクリックし、ラベルの値を **Hello World!** で置き換えます。
- これまでの手順を繰り返し、ラベル値 **Have a nice day!** と **Top of the Morning!** を追加します。
- 「OK」をクリックして、値ダイアログボックスを閉じます。

9 ファイルを保存します。

10 アクションパネルが開いていない場合は、**F9** キーを押すか、ウィンドウメニューのアクションを選択して開きます。メインタimelineのフレーム 1 をクリックし、アクションパネルに次のコードを入力します。

```
import flash.events.Event;
import fl.events.ComponentEvent;
import fl.events.ColorPickerEvent;
import fl.controls.RadioButtonGroup;

var rbGrp:RadioButtonGroup = RadioButtonGroup.getGroup("fontRbGrp");
rbGrp.addEventListener(MouseEvent.CLICK, rbHandler);
txtCp.addEventListener(ColorPickerEvent.CHANGE, cpHandler);
msgCb.addEventListener(Event.CHANGE, cbHandler);
```

最初の 3 行で、アプリケーションが使用するイベントクラスを読み込みます。ユーザーがコンポーネントの 1 つを操作すると、イベントが発生します。次の 5 行で、アプリケーションがリスンするイベントのイベントハンドラーを登録します。click イベントは、ユーザーが **RadioButton** をクリックすると発生します。change イベントは、ユーザーが **ColorPicker** で別の色を選択すると発生します。ComboBox の change イベントは、ユーザーがドロップダウンリストから別の挨拶を選択すると発生します。

4 行目では、アプリケーションがイベントリスナーを各ボタンに個別に割り当てるのではなく、**RadioButtons** のグループに割り当てることができるように、**RadioButtonGroup** クラスを読み込みます。

11 次のコード行をアクションパネルに追加して、**tf** **TextFormat** オブジェクトを作成します。アプリケーションはこのオブジェクトを使用して、**TextArea** のテキストの **size** および **color** スタイルプロパティを変更します。

```
var tf:TextFormat = new TextFormat();
```

12 次のコードを追加して、**rbHandler** イベント処理関数を作成します。この関数は、ユーザーが **RadioButton** コンポーネントの 1 つをクリックしたときに click イベントを処理します。

```
function rbHandler(event:MouseEvent):void {
    switch(event.target.selection.name) {
        case "smallRb":
            tf.size = 14;
            break;
        case "largerRb":
            tf.size = 18;
            break;
        case "largestRb":
            tf.size = 24;
            break;
    }
    aTa.setStyle("textFormat", tf);
}
```

この関数は、switch ステートメントを使用して、event オブジェクトの target プロパティを確認し、イベントをトリガーした RadioButton を特定します。currentTarget プロパティには、イベントをトリガーしたオブジェクトの名前が含まれます。ユーザーがクリックした RadioButton に応じて、アプリケーションは TextArea のテキストのサイズを 14、18、または 24 ポイントに変更します。

13 次のコードを追加して cpHandler() 関数を実装します。この関数は、ColorPicker での値の変更を処理します。

```
function cpHandler(event:ColorPickerEvent):void {  
    tf.color = event.target.selectedColor;  
    aTa.setStyle("textFormat", tf);  
}
```

この関数は、tf TextFormat オブジェクトの color プロパティを ColorPicker で選択した色に設定し、次に setStyle() を呼び出して、これを aTa TextArea インスタンスのテキストに適用します。

14 次のコードを追加して cbHandler() 関数を実装します。この関数は、ComboBox での選択内容の変更を処理します。

```
function cbHandler(event:Event):void {  
    aTa.text = event.target.selectedItem.label;  
}
```

この関数は、TextArea のテキストを、ComboBox で選択したテキスト、event.target.selectedItem.label で置き換えます。

15 制御/ムービーをプレビューするを選択するか、Control+Enter を押してコードをコンパイルし、Greetings アプリケーションをテストします。

次の節では、外部 ActionScript クラスと、ライブラリの必須コンポーネントだけが含まれる FLA ファイルを使用して同じアプリケーションを作成する方法について説明します。

外部クラスファイルを使用した Greetings2 アプリケーションの作成:

- 1 ファイル/新規を選択します。
- 2 新規ドキュメントダイアログボックスで「Flash ファイル (ActionScript 3.0)」を選択し、「OK」をクリックします。
新しい Flash ウィンドウが開きます。
- 3 ファイル/保存を選択し、Flash ファイルの名前を **Greetings2.fla** と指定し、「保存」ボタンをクリックします。
- 4 次の各コンポーネントをコンポーネントパネルからライブラリにドラッグします。
 - ColorPicker
 - ComboBox
 - RadioButton
 - TextArea

コンパイルされた SWF ファイルはこれらの各アセットを使用するので、ライブラリに追加する必要があります。コンポーネントをライブラリパネルの下部にドラッグします。これらのコンポーネントをライブラリに追加すると、他のアセット (List、TextInput、UI ScrollBox など) が自動的に追加されます。

5 プロパティウィンドウで、「ドキュメントクラス」に **Greetings2** と入力します。

"このドキュメントクラスの定義が見つかりません" という内容の警告が表示される場合は、無視します。次の手順に従って Greetings2 クラスを定義します。このクラスは、アプリケーションの主要な機能を定義します。

- 6 "Greetings2.fla" ファイルを保存します。
- 7 ファイル/新規を選択します。
- 8 新規ドキュメントダイアログボックスで「ActionScript ファイル」を選択し、「OK」をクリックします。
新しいスクリプトウィンドウが開きます。
- 9 次のコードをスクリプトウィンドウに追加します。

```
package {
    import flash.display.Sprite;
    import flash.events.Event;
    import flash.events.MouseEvent;
    import flash.text.TextFormat;
    import fl.events.ComponentEvent;
    import fl.events.ColorPickerEvent;
    import fl.controls.ColorPicker;
    import fl.controls.ComboBox;
    import fl.controls.RadioButtonGroup;
    import fl.controls.RadioButton;
    import fl.controls.TextArea;
    public class Greetings2 extends Sprite {
        private var aTa:TextArea;
        private var msgCb:ComboBox;
        private var smallRb:RadioButton;
        private var largerRb:RadioButton;
        private var largestRb:RadioButton;
        private var rbGrp:RadioButtonGroup;
        private var txtCp:ColorPicker;
        private var tf:TextFormat = new TextFormat();
        public function Greetings2() {
```

このスクリプトは、Greetings2 という名前の ActionScript 3.0 クラスを定義します。このスクリプトの動作は次のとおりです。

- ファイルで使用するクラスを読み込みます。通常は、これらの import ステートメントは、コードで別のクラスを参照するときに追加しますが、簡潔にするために、この例ではすべてを一度に読み込みます。
- コードに追加するさまざまなタイプのコンポーネントオブジェクトを表す変数を宣言します。別の変数によって tf TextFormat オブジェクトが作成されます。
- クラスのコンストラクター関数 Greetings2() を定義します。次の手順で、この関数に数行を追加し、他のメソッドをクラスに追加します。

10 ファイル/保存を選択し、ファイルの名前を **Greetings2.as** と指定し、「保存」ボタンをクリックします。

11 次のコード行を Greeting2() 関数に追加します。

```
        createUI();
        setUpHandlers();
    }
```

関数は次のようになります。

```
public function Greetings2() {
    createUI();
    setUpHandlers();
}
```

12 次のコード行を Greeting2() メソッドの右括弧の後に追加します。

```
private function createUI() {
    bldTxtArea();
    bldColorPicker();
    bldComboBox();
    bldRadioButtons();
}
private function bldTxtArea() {
    aTa = new TextArea();
    aTa.setSize(230, 44);
    aTa.text = "Hello World!";
    aTa.move(165, 57);
    addChild(aTa);
}
private function bldColorPicker() {
    txtCp = new ColorPicker();
    txtCp.move(96, 72);
    addChild(txtCp);
}
private function bldComboBox() {
    msgCb = new ComboBox();
    msgCb.width = 130;
    msgCb.move(265, 120);
    msgCb.prompt = "Greetings";
    msgCb.addItem({data:"Hello.", label:"English"});
    msgCb.addItem({data:"Bonjour.", label:"Français"});
    msgCb.addItem({data:"¡Hola!", label:"Español"});
    addChild(msgCb);
}
private function bldRadioButtons() {
    rbGrp = new RadioButtonGroup("fontRbGrp");
    smallRb = new RadioButton();
    smallRb.setSize(100, 22);
    smallRb.move(155, 120);
    smallRb.group = rbGrp; //"fontRbGrp";
    smallRb.label = "Small";
    smallRb.name = "smallRb";
    addChild(smallRb);
    largerRb = new RadioButton();
    largerRb.setSize(100, 22);
    largerRb.move(155, 148);
    largerRb.group = rbGrp;
    largerRb.label = "Larger";
    largerRb.name = "largerRb";
    addChild(largerRb);
    largestRb = new RadioButton();
    largestRb.setSize(100, 22);
    largestRb.move(155, 175);
    largestRb.group = rbGrp;
    largestRb.label = "Largest";
    largestRb.name = "largestRb";
    addChild(largestRb);
}
```

これらの行の動作は次のとおりです。

- アプリケーションで使用するコンポーネントをインスタンス化します。
- 各コンポーネントのサイズ、位置およびプロパティを設定します。
- addChild() メソッドを使用して、各コンポーネントをステージに追加します。

13 bldRadioButtons() メソッドの右括弧の後に、setUpHandlers() メソッドの次のコードを追加します。

```
private function setUpHandlers():void {
    rbGrp.addEventListener(MouseEvent.CLICK, rbHandler);
    txtCp.addEventListener(ColorPickerEvent.CHANGE, cpHandler);
    msgCb.addEventListener(Event.CHANGE, cbHandler);
}
private function rbHandler(event:MouseEvent):void {
    switch(event.target.selection.name) {
        case "smallRb":
            tf.size = 14;
            break;
        case "largerRb":
            tf.size = 18;
            break;
        case "largestRb":
            tf.size = 24;
            break;
    }
    aTa.setStyle("textFormat", tf);
}
private function cpHandler(event:ColorPickerEvent):void {
    tf.color = event.target.selectedColor;
    aTa.setStyle("textFormat", tf);
}
private function cbHandler(event:Event):void {
    aTa.text = event.target.selectedItem.data;
}
}
}
```

これらの関数は、コンポーネントのイベントリスナーを定義します。

14 ファイル／保存を選択し、ファイルを保存します。

15 制御／ムービーをプレビューするを選択するか、Control+Enter を押してコードをコンパイルし、Greetings2 アプリケーションをテストします。

今後のコード例の作成と実行

Greetings アプリケーションの作成と実行が終了すると、本マニュアルに示した他のコード例の実行に必要な基本的な知識を修得できます。個々の例では関連する ActionScript 3.0 コードを強調表示して説明するので、本マニュアルの各例をカットし、FLA ファイルにペーストした後、コンパイルして実行できます。

第 3 章：コンポーネントの利用

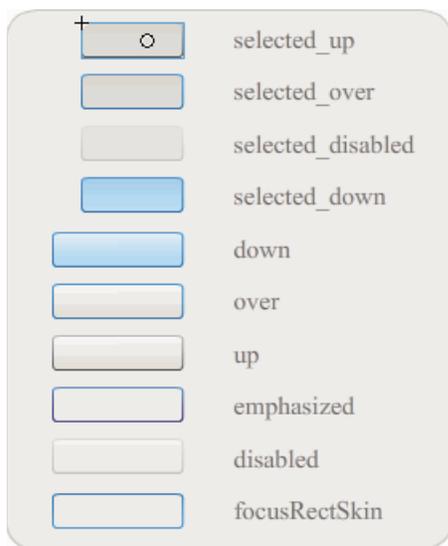
コンポーネントアーキテクチャ

Adobe® ActionScript® 3.0 コンポーネントは、Adobe® Flash Player バージョン 9.0.28.0 以降でサポートされます。Flash CS4 より前に作成されたコンポーネントとは互換性がありません。Adobe® ActionScript® 2.0 コンポーネントの使用について詳しくは、『**Adobe® ActionScript® 2.0** コンポーネントガイド』および『**Adobe® ActionScript® 2.0** 言語およびコンポーネントリファレンス』を参照してください。

Adobe ActionScript 3.0 ユーザーインターフェイス (UI) コンポーネントは FLA ベースのコンポーネントとして実装されますが、Flash CS5 は SWC ベースのコンポーネントと FLA ベースのコンポーネントを両方もサポートします。例えば、FLVPlayback コンポーネントと FLVPlaybackCaptioning コンポーネントは SWC ベースのコンポーネントです。どちらのタイプのコンポーネントも Components フォルダに配置して、コンポーネントパネルに表示することができます。この 2 種類のコンポーネントは異なる方法で構築されるので、ここでは別々に説明しています。

ActionScript 3.0 FLA ベースのコンポーネント

ActionScript 3.0 ユーザーインターフェイス (UI) コンポーネントは、ビルトインスキンを持つ FLA ベース (.fla) のファイルです。ステージ上のコンポーネントをダブルクリックすることによってスキンにアクセスし、編集できます。コンポーネントのスキンおよびその他のアセットは、タイムラインのフレーム 2 に配置されます。コンポーネントをダブルクリックすると、自動的にフレーム 2 に移動し、コンポーネントのスキンのパレットが開きます。次の図は、Button コンポーネントのスキンのパレットを示しています。



Button コンポーネントのスキン

コンポーネントスキンおよびコンポーネントのカスタマイズについて詳しくは、93 ページの「[UI コンポーネントのカスタマイズ](#)」および 146 ページの「[FLVPlayback コンポーネントのカスタマイズ](#)」を参照してください。

アプリケーションのコンパイルを高速化し、ActionScript 3.0 の設定とのコンフリクトを回避するために、Flash CS5 FLA ベースの UI コンポーネントには、コンポーネントのコンパイル済み ActionScript コードを含む SWC も含まれています。ComponentShim SWC は、ユーザーインターフェイスコンポーネントごとにフレーム 2 のステージに配置され、プリコンパイルされた定義を提供します。ActionScript で利用できるようにするには、コンポーネントをステージまたはライブラリ

内に配置し、そのリンケージプロパティの「最初のフレームに書き出し」オプションを選択しておく必要があります。また、ActionScript を使用してコンポーネントを作成するには、import ステートメントでそのクラスを読み込み、アクセスする必要があります。import ステートメントについては、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」を参照してください。

SWC ベースのコンポーネント

SWC ベースのコンポーネントには、FLA ファイルおよび ActionScript クラスファイルが含まれていますが、コンパイルされ、SWC として書き込まれています。SWC ファイルはプリコンパイルされた Flash シンボルと ActionScript コードのパッケージであり、これにより、変更されないシンボルとコードをリコンパイルする必要がなくなります。

FLVPlayback コンポーネントと FLVPlaybackCaptioning コンポーネントは SWC ベースのコンポーネントです。これらのコンポーネントには、ビルトインスキンではなく外部スキンが含まれています。FLVPlayback コンポーネントにはデフォルトのスキンがあり、このスキンは、デザイン済みスキンのコレクションから選択するか、コンポーネントパネルの UI コントロール (BackButton、BufferingBar など) からコントロールをカスタマイズするか、カスタムスキンを作成することによって変更できます。詳しくは、146 ページの「[FLVPlayback コンポーネントのカスタマイズ](#)」を参照してください。

Flash では、次のようにしてムービークリップをコンパイルされたクリップに変換できます。

ムービークリップのコンパイル

- ライブライパネルにあるムービークリップを右クリック (Windows) または Control キーを押しながらクリック (Macintosh) し、「コンパイルされたクリップに変換」を選択します。

コンパイルされたクリップは、コンパイル前の通常のムービークリップと同じように動作しますが、表示やパブリッシュははるかに高速で処理されます。コンパイルされたクリップを編集することはできませんが、そのプロパティはプロパティインスペクターとコンポーネントインスペクターに表示されます。

SWC コンポーネントには、コンパイルされたクリップ、コンポーネントのプリコンパイルされた ActionScript 定義、およびこのコンポーネントについて記述したその他のファイルが含まれています。ユーザー独自のコンポーネントを作成した場合は、それを SWC ファイルとして書き出し、配布できます。

SWC ファイルの書き出し

- ライブライパネルにあるムービークリップを右クリック (Windows) または Control キーを押しながらクリック (Macintosh) し、「SWC ファイル書き出し」を選択します。

注意: Flash CS4 以降の SWC ファイルの形式と Flex SWC 形式には互換性があるので、2 つの製品の間で SWC ファイルを交換可能ですが、変更が必要になることがあります。

SWC ベースのコンポーネントの作成について詳しくは、www.adobe.com/go/learn_fl_creating_components_jp を参照してください。

ActionScript 3.0 コンポーネント API

各 ActionScript 3.0 コンポーネントは、パッケージフォルダーに置かれた ActionScript 3.0 クラスに基づいて構築され、名前の形式は `fl.<パッケージ名>.<クラス名>` になります。例えば、Button コンポーネントは Button クラスのインスタンスであり、そのパッケージ名は `fl.controls.Button` です。コンポーネントクラスをアプリケーションに読み込むときには、このパッケージ名を参照する必要があります。次のステートメントで Button クラスを読み込みます。

```
import fl.controls.Button;
```

コンポーネントクラスファイルの場所について詳しくは、18 ページの「[コンポーネントファイルの操作](#)」を参照してください。

コンポーネントのクラスは、メソッド、プロパティ、イベントおよびスタイルを定義します。アプリケーションの中ではこれらを使用してクラスとやり取りできます。ActionScript 3.0 UI コンポーネントは、Sprite クラスと UIComponent クラスのサブクラスであり、それらのクラスからプロパティ、メソッドおよびイベントを継承します。Sprite クラスは、基本的な表示リスト構成要素であり、MovieClip に似ていますが、タイムラインはありません。UIComponent クラスは、インタラクティブ、非インタラクティブを問わず、すべてのビジュアルコンポーネントの基本クラスです。各コンポーネントの継承パスおよびプロパティ、メソッド、イベント、スタイルの説明は、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」にあります。

すべての ActionScript 3.0 コンポーネントが ActionScript 3.0 イベント処理モデルを使用します。イベント処理について詳しくは、23 ページの「[イベント処理](#)」および『ActionScript 3.0 のプログラミング』を参照してください。

コンポーネントファイルの操作

この節では、コンポーネントファイルの格納場所、ActionScript ソースファイルの検索先、コンポーネントパネルからコンポーネントを追加および削除する方法について説明します。

コンポーネントファイルの格納場所

Flash コンポーネントは、アプリケーションレベルの設定フォルダー内にあります。

注意：これらのフォルダーについて詳しくは、『Flash ユーザーガイド』の「Flash と共にインストールされる設定フォルダー」を参照してください。

コンポーネントは、次の場所にインストールされます。

- Windows 2000 または Windows XP の場合：C:\Program Files\Adobe\Adobe Flash CS5\language\Configuration\Components
- Mac OS X の場合：Macintosh HD/ アプリケーション /Adobe Flash CS5/Configuration/Components
コンポーネントフォルダー内で、ユーザーインターフェイス (UI) コンポーネントは User Interface.fla ファイルに、FLVPlayback (FLVPlaybackAS3.swc) コンポーネントと FLVPlaybackCaptioning コンポーネントはビデオフォルダーにあります。
コンポーネントは次のユーザー別の場所に格納することもできます。
- Windows 2000 または Windows XP の場合：C:\Documents and Settings\%username%\Local Settings\Application Data\Adobe\Adobe Flash CS5\en\Configuration\Components
- Windows Vista の場合：C:\Users\%username%\Local Settings\Application Data\Adobe\Adobe Flash CS5\jp\Configuration\Components

注意：Windows では、Application Data フォルダーはデフォルトで非表示になっています。非表示のフォルダーとファイルを表示するには、マイコンピューターを選択し、Windows エクスプローラーを開きます。次にツール/フォルダーオプションを選択し「表示」タブを選択します。「表示」タブで「すべてのファイルとフォルダーを表示する」ラジオボタンを選択します。

- Mac OS X の場合：Macintosh HD/ ユーザー /<username>/ライブラリ /Application Support/Adobe Flash CS5/Configuration/Components

コンポーネントソースファイルの格納場所

Windows 2000 または Windows XP の場合、コンポーネントの ActionScript (.as) クラスファイル（またはソースファイル）は、次のアプリケーションフォルダーにインストールされます。

ユーザーインターフェイスコンポーネント C:\Program Files\Adobe\Adobe Flash CS5\jp\Configuration\Component Source\ActionScript 3.0\User Interface\fl

FLVPlayback C:\Program Files\Adobe\Adobe Flash CS5\jp\Configuration\Component Source\ActionScript 3.0\FLVPlayback\fl\video

FLVPlaybackCaptioning C:\Program Files\Adobe\Adobe Flash CS5\jp\Configuration\Component Source\ActionScript 3.0\FLVPlaybackCaptioning\fl\video

Mac OS X の場合、コンポーネントソースファイルは次の場所にあります。

ユーザーインターフェイスコンポーネント Macintosh HD/ アプリケーション /Adobe Flash CS5/Configuration/Component Source/ActionScript 3.0/User Interface/fl

FLVPlayback Macintosh HD/ アプリケーション /Adobe Flash CS5/Configuration/Component Source/ActionScript 3.0/FLVPlayback/fl/video

FLVPlaybackCaptioning Macintosh HD/ アプリケーション /Adobe Flash CS5/Configuration/Component Source/ActionScript 3.0/FLVPlaybackCaptioning/fl/video

コンポーネントソースファイルおよびクラスパス

ActionScript 3.0 コンポーネントはコードをコンパイルインしているため、ActionScript クラスファイルの場所をクラスパス変数に指定しないでください。場所をクラスパスに指定すると、アプリケーションのコンパイルに要する時間が長くなります。ただし、Flash により、クラスパス設定でコンポーネントクラスファイルが検出されると、そのクラスファイルは常にコンポーネントのコンパイルインコードよりも優先されます。

コンポーネントを含むアプリケーションをデバッグする場合も、コンポーネントソースファイルの場所をクラスパス設定に追加することをお勧めします。詳しくは、20 ページの「[コンポーネントアプリケーションのデバッグ](#)」を参照してください。

コンポーネントファイルの編集

SWC ベースのコンポーネントを更新、追加、削除する場合、または FLA ベースの新しいコンポーネントを Flash に追加する場合は、それらをコンポーネントパネルにリロードし、利用可能にする必要があります。コンポーネントをリロードするには、Flash を再起動するか、コンポーネントパネルメニューから「リロード」を選択します。これにより、Flash はコンポーネントフォルダーに追加されたコンポーネントを認識します。

Flash の実行中のコンポーネントパネルへのコンポーネントのリロード：

- コンポーネントパネルメニューから「リロード」を選択します。

コンポーネントパネルからのコンポーネントの削除：

- コンポーネントフォルダーから FLA、SWC、または MXP ファイルを削除し、Flash を再起動するか、コンポーネントパネルメニューから「リロード」を選択します。MXP ファイルは、Adobe Exchange からダウンロードされたコンポーネントファイルです。

Flash の実行中に SWC ベースのコンポーネントを削除および置換し、リロードによってその変更を反映できますが、FLA ベースのコンポーネントを変更または削除した場合、その変更は Flash を終了し、再起動するまで反映されません。ただし、FLA ベースのコンポーネントを追加して、「リロード」コマンドでロードすることは可能です。

 はじめに、変更する Flash コンポーネントファイル (.fla または .as) のコピーを作成してください。その後、必要に応じてそれを復元できます。

コンポーネントアプリケーションのデバッグ

ActionScript 3.0 コンポーネントには、アプリケーションのコンパイル時間を短縮するために、すべてのソースコードが組み込まれています。ただし、Flash デバッガーはコンパイルされたクリップ内のコードを検査できません。そのため、コンポーネントのソースコードのレベルまでアプリケーションをデバッグする場合には、コンポーネントソースファイルをクラスパス設定に追加する必要があります。

コンポーネントパッケージフォルダーの場所は、そのコンポーネント型のソースファイルの場所を基準としています。すべての UI コンポーネントのすべての ActionScript 3.0 ソースファイルを参照するには、ユーザーインターフェイスパッケージのクラスパスに次の場所を追加します。

- \$(AppConfig)/Component Source/ActionScript 3.0/User Interface

注意：これにより、すべての UI コンポーネントのコンパイルインコードが無効になり、アプリケーションのコンパイル時間が増加します。何らかの理由でコンポーネントのソースファイルを変更した場合、そのコンポーネントのビヘイビアが変わる可能性があります。

クラスパスを設定するには、「編集」メニューから「環境設定」を選択し、「カテゴリ」リストで「ActionScript」を選択してから、「ActionScript 3.0 設定」ボタンをクリックします。新しい項目を追加するには、現在の設定を表示しているウィンドウの上にあるプラスをクリックします。

\$(AppConfig) 変数は、Flash CS5 をインストールした場所にある Flash CS5 設定フォルダーを参照しています。通常、このパスは次のようになります。

- Windows 2000 または Windows XP の場合：C:\Program Files\Adobe\Adobe Flash CS5\language\Configuration
- Mac OS X の場合：Macintosh HD/ アプリケーション /Adobe Flash CS5/Configuration

注意：コンポーネントソースファイルを変更する必要があるときは、元のソースファイルを別の場所にコピーし、その場所をクラスパスに追加することを強くお勧めします。

コンポーネントソースファイルの場所について詳しくは、19 ページの「[コンポーネントソースファイルの格納場所](#)」を参照してください。

パラメーターとプロパティの設定

各コンポーネントには、その外観やビヘイビアを変更するためのパラメーターがあります。パラメーターは、コンポーネントのクラスのプロパティであり、プロパティインスペクターとコンポーネントインスペクターに表示されます。最もよく使用されるプロパティはオーサリングパラメーターとして表示されます。その他のプロパティは ActionScript を使用して設定する必要があります。オーサリング時に設定できるすべてのパラメーターは、ActionScript を使用して設定することができます。ActionScript で設定したパラメーターは、オーサリング時に設定した値よりも優先されます。

ほとんどの ActionScript 3.0 ユーザーインターフェイス (UI) コンポーネントは、UIComponent クラスおよび基本クラスからプロパティとメソッドを継承します。例えば、Button クラスと CheckBox クラスは、UIComponent クラスおよび BaseButton クラスの両方からプロパティを継承します。ユーザーは、コンポーネントの継承されたプロパティおよび独自のクラスプロパティにアクセスできます。例えば、ProgressBar コンポーネントは ProgressBar.enabled プロパティを UIComponent から継承しますが、独自の ProgressBar.percentComplete プロパティも持っています。ユーザーは両方のプロ

パティにアクセスし、Progress Bar コンポーネントのインスタンスとやり取りできます。コンポーネントのプロパティについて詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の各クラスエントリを参照してください。

プロパティインスペクターまたはコンポーネントインスペクターを使用してコンポーネントインスタンスのパラメーターを設定できます。

プロパティインスペクターでのコンポーネントのインスタンス名の入力：

- 1 ウィンドウ/プロパティ/プロパティを選択します。
- 2 ステージでコンポーネントのインスタンスを選択します。
- 3 「ムービークリップ」ドロップダウンリストの下にある「<インスタンス名>」ボックスにコンポーネントインスタンスの名前を入力します。または、「パラメーター」タブをクリックして、「コンポーネント」の下にあるボックスに名前を入力します。設定するすべてのパラメーターの値を入力します。

このとき、コンポーネントの種類を表す接尾辞をインスタンス名に付加すると、ActionScript コードが理解しやすくなります。例えば、インスタンス名 **licenseSb** は、コンポーネントが **licenseTa** テキスト領域の使用許諾契約をスクロールするスクロールバーであることを示しています。

コンポーネントインスペクターでのコンポーネントインスタンスのパラメーターの入力：

- 1 ウィンドウ/コンポーネントインスペクターを選択します。
- 2 ステージでコンポーネントのインスタンスを選択します。
- 3 「パラメーター」タブをクリックし、リストされたパラメーターの値を入力します。



コンポーネントインスペクターのコンポーネントパラメーター

ActionScript でのコンポーネントプロパティの設定

ActionScript では、ドット (.) 演算子 (ドットシンタックス) を使用して、ステージ上のオブジェクトまたはインスタンスに属するプロパティまたはメソッドにアクセスします。ドットシンタックス式は、インスタンスの名前、ドット、エレメントの順に記述します。例えば、次の ActionScript コードは、CheckBox インスタンス aCh の width プロパティを 50 ピクセルに設定します。

```
aCh.width = 50;
```

次の if ステートメントは、ユーザーがチェックボックスを選択しているかどうかをチェックします。

```
if (aCh.selected == true) {  
    displayImg (redCar);  
}
```

ライブラリ

ユーザーが最初にコンポーネントをドキュメントに追加したとき、Flash ではムービークリップとしてライブラリパネルに読み込みます。コンポーネントパネルからコンポーネントをライブラリパネルに直接ドラッグし、そのインスタンスをステージに追加することもできます。いずれの場合も、コンポーネントをライブラリに追加した後でなければ、そのクラスエレメントにアクセスできません。

コンポーネントをライブラリに追加し、ActionScript を使用してそのインスタンスを作成する場合は、最初に import ステートメントを使用してそのクラスを読み込む必要があります。import ステートメントには、コンポーネントのパッケージ名とクラス名を両方とも指定する必要があります。例えば、次のステートメントによって Button クラスを読み込みます。

```
import fl.controls.Button;
```

コンポーネントをライブラリに入れると、Flash は、さまざまな状態のスキンを含むそのアセットのフォルダーも読み込みます。コンポーネントのスキンは、アプリケーション内でグラフィック表示を構築するシンボルのコレクションです。1 つのスキンは、コンポーネントの特定の状態を示したグラフィカル表現、またはムービークリップです。

Component Assets フォルダー内のコンテンツを使用して、必要ときにコンポーネントのスキンを変更できます。詳しくは、93 ページの「[UI コンポーネントのカスタマイズ](#)」を参照してください。

コンポーネントをライブラリに入れておくと、コンポーネントパネルかライブラリパネルからアイコンをステージにドラッグすることによって、そのインスタンスをドキュメントに追加できます。

コンポーネントのサイズ変更

コンポーネントインスタンスのサイズを変更するには、自由変形ツールまたは setSize() メソッドを使用します。setSize() メソッドは、すべてのコンポーネントインスタンスから呼び出して、コンポーネントのサイズを変更することができます (UIComponent.setSize() を参照してください)。次のコードでは、List コンポーネントのインスタンスのサイズが、幅 200 ピクセル、高さ 300 ピクセルに変更されます。

```
aList.setSize(200, 300);
```

コンポーネントのラベルは、適切に表示されるようにサイズが自動的に調整されることはありません。ドキュメントに追加されたコンポーネントインスタンスが小さすぎてラベルを表示できない場合、ラベルのテキストは切り詰められます。ラベルが適切に表示されるように、コンポーネントのサイズを調整する必要があります。

コンポーネントのサイズ調整について詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の個々のエントリを参照してください。

ライブプレビュー

ライブプレビュー機能はデフォルトで有効になっており、ステージ上のコンポーネントがパブリッシュされた Flash コンテンツでどのように表示されるかを確認できます。コンポーネントは、おおよそのサイズで表示されます。

ライブプレビューのオンとオフを切り替えるには

- 制御/ライブプレビューを有効にするを選択します。オプションの横にチェックマークが付いていれば、そのオプションは有効になっています。

ライブプレビューには、さまざまなコンポーネントに指定された各パラメーターが反映されます。ライブプレビューに反映されるコンポーネントのパラメーターについては、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の各コンポーネントのエントリを参照してください。



ライブプレビューを有効にした Button コンポーネント



ライブプレビューを無効にした Button コンポーネント

ライブプレビュー内のコンポーネントは機能しません。コンポーネントの機能をテストするには、制御／ムービープレビューコマンドを使用します。

イベント処理

すべてのコンポーネントが、ユーザーとやり取りしたときに、イベントをブロードキャストします。例えば、ユーザーが Button をクリックすると、Button は MouseEvent.CLICK イベントを送り出し、ユーザーがリスト内の項目を選択すると、List は Event.CHANGE イベントを送り出します。コンポーネントにとって重要な事態が発生したときもイベントが発生する場合があります。例えば、UILoader インスタンスでコンテンツのロードが完了すると、Event.COMPLETE イベントが生成されます。イベントを処理するには、イベントが発生したときに実行する ActionScript コードを作成します。

コンポーネントのイベントには、コンポーネントが継承しているすべてのクラスのイベントが含まれます。つまり、すべての ActionScript 3.0 ユーザーインターフェイスコンポーネントが、UIComponent クラスからイベントを継承します。これは、UIComponent クラスが ActionScript 3.0 ユーザーインターフェイスコンポーネントの基本クラスであるためです。コンポーネントがブロードキャストするイベントのリストについては、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」のコンポーネントのクラスエントリにある「イベント」の節を参照してください。

ActionScript 3.0 におけるイベント処理について詳しくは、『[ActionScript 3.0 のプログラミング](#)』を参照してください。

イベントリスナについて

ActionScript 3.0 コンポーネントのイベント処理では次のような点が重要です。

- すべてのイベントは、コンポーネントクラスのインスタンスによってブロードキャストされます。ブロードキャストを実行したコンポーネントインスタンスがブロードキャストターです。
- コンポーネントインスタンスの `addEventListener()` メソッドを呼び出すことによって、イベントリスナを登録します。例えば、次のコード行は、MouseEvent.CLICK イベントのリスナを Button インスタンス `aButton` に追加します。

```
aButton.addEventListener(MouseEvent.CLICK, clickHandler);
```

`addEventListener()` メソッドの 2 番目のパラメーターは、イベントが発生したときに呼び出される関数の名前 `clickHandler` を登録します。この関数は、コールバック関数とも呼ばれます。

- 1 つのコンポーネントインスタンスに複数のリスナを登録できます。

```
aButton.addEventListener(MouseEvent.CLICK, clickHandler1);  
aButton.addEventListener(MouseEvent.CLICK, clickHandler2);
```

- また、1 つのリスナを複数のコンポーネントインスタンスに登録することもできます。

```
aButton.addEventListener(MouseEvent.CLICK, clickHandler1);  
bButton.addEventListener(MouseEvent.CLICK, clickHandler1);
```

- イベントハンドラー関数は、イベントタイプと、イベントをブロードキャストするインスタンスに関する情報を含むイベントオブジェクトに渡されます。詳しくは、24 ページの「[イベントオブジェクトについて](#)」を参照してください。
- リスナは、アプリケーションが終了するか、ユーザーが `removeEventListener()` メソッドを使用して明示的に削除するまではアクティブな状態を維持します。例えば、次のコード行は、`aButton` の MouseEvent.CLICK イベントのリスナを削除します。

```
aButton.removeEventListener(MouseEvent.CLICK, clickHandler);
```

イベントオブジェクトについて

イベントオブジェクトは、Event オブジェクトクラスから継承され、発生したイベントに関する情報を含むプロパティを持ちます。例えば、イベントに関する重要な情報を提供する target プロパティや type プロパティなどがあります。

プロパティ	説明
type	イベントのタイプを示す文字列です。
target	イベントをブロードキャストしたコンポーネントインスタンスへの参照です。

イベントにその他のプロパティがある場合は、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」のイベントのクラスの説明内にそのプロパティの一覧が記載されています。

イベントオブジェクトは、イベントが発生したときに自動的に生成され、イベントハンドラー関数に渡されます。

関数内でイベントオブジェクトを使用すると、ブロードキャストされたイベントの名前や、イベントをブロードキャストするコンポーネントのインスタンス名にアクセスできます。インスタンス名から、他のコンポーネントプロパティにアクセスすることもできます。例えば、次のコードは evtObj イベントオブジェクトの target プロパティを使用して aButton の label プロパティにアクセスし、出力パネルに表示します。

```
import fl.controls.Button;
import flash.events.MouseEvent;

var aButton:Button = new Button();
aButton.label = "Submit";
addChild(aButton);
aButton.addEventListener(MouseEvent.CLICK, clickHandler);

function clickHandler(evtObj:MouseEvent){
    trace("The " + evtObj.target.label + " button was clicked");
}
```

表示リストの操作

すべての ActionScript 3.0 コンポーネントは、DisplayObject クラスから継承されるので、そのメソッドとプロパティにアクセスし、表示リストとやり取りします。表示リストは、表示されるオブジェクトとビジュアルエレメントのアプリケーション内での階層です。この階層には次のエレメントが含まれます。

- ステージ。トップレベルのコンテナです。
- 表示オブジェクト。シェイプ、ムービークリップ、テキストフィールドなどを含みます。
- 表示オブジェクトコンテナ。子表示オブジェクトを含むことができる特殊な表示オブジェクトです。

表示リスト内のオブジェクトの順序によって、親コンテナ内での深度が決まります。オブジェクトの深度は、ステージ上または表示コンテナ内の一番上から下へ、または最前面から背面への位置を示します。深度の順序はオブジェクトが重なっているときには明白ですが、重なっていないときでも存在します。表示リストのすべてのオブジェクトに、ステージ上の対応する深度があります。オブジェクトを他のオブジェクトの前面または背面に移動してオブジェクトの深度を変更する場合は、表示リスト内の位置を変更する必要があります。表示リスト内のオブジェクトのデフォルト順序は、オブジェクトがステージ上に置かれた順序です。表示リストの位置 0 は、深度順序の底にあるオブジェクトです。

表示リストへのコンポーネントの追加

コンテナの `addChild()` または `addChildAt()` メソッドを呼び出して、オブジェクトを `DisplayObjectContainer` オブジェクトに追加できます。ステージの場合はオーサリング時にオブジェクトを作成し、コンポーネントの場合はオブジェクトをコンポーネントパネルからステージにドラッグすることによって、オブジェクトを表示リストに追加することもできます。

`ActionScript` を使用してオブジェクトをコンテナに追加するには、まず `new` 演算子を指定してコンストラクターを起動し、そのインスタンスを作成してから、`addChild()` または `addChildAt()` メソッドを呼び出してステージ上および表示リストに配置します。`addChild()` メソッドはオブジェクトを表示リストの次の位置に配置しますが、`addChildAt()` メソッドはオブジェクトを追加する位置を指定します。指定した位置に既にオブジェクトがある場合、その位置にあるオブジェクト、およびそれよりも上位にあるオブジェクトが1つ上の位置に移動します。`DisplayObjectContainer` オブジェクトの `numChildren` プロパティは、含んでいる表示オブジェクトの数を保持します。表示リストからオブジェクトを取得するには、`getChildAt()` メソッドを呼び出して位置を指定するか、オブジェクトの名前がわかっている場合は、`getChildByName()` メソッドを呼び出します。

注意：表示リスト内のコンポーネントに名前でアクセスするには、`ActionScript` を使用してコンポーネントを追加するときに、`name` プロパティに名前を割り当てる必要があります。

次の例は、表示リスト内の3つのコンポーネントの名前と位置を表示します。最初に、`NumericStepper`、`Button` および `ComboBox` を互いに重なるようにステージにドラッグし、インスタンス名 `aNs`、`aButton` および `aCb` を指定します。続いて、次のコードをタイムラインのフレーム1のアクションパネルに追加します。

```
var i:int = 0;
while(i < numChildren) {
    trace(getChildAt(i).name + " is at position: " + i++);
}
```

出力パネルに次の行が表示されます。

```
aNs is at position: 0
aButton is at position: 1
aCb is at position: 2
```

表示リスト内のコンポーネントの移動

表示リスト内のオブジェクトの位置および表示深度を変更するには、`addChildAt()` メソッドを呼び出して、オブジェクトの名前と配置する位置をメソッドのパラメーターとして指定します。例えば、次のコードを前の例に追加して、`NumericStepper` を一番上に配置し、このループを繰り返して表示リスト内のコンポーネントの新しい位置を表示します。

```
this.addChildAt(aNs, numChildren - 1);
i = 0;
while(i < numChildren) {
    trace(getChildAt(i).name + " is at position: " + i++);
}
```

出力パネルに次の内容が表示されます。

```
aNs is at position: 0
aButton is at position: 1
aCb is at position: 2
aButton is at position: 0
aCb is at position: 1
aNs is at position: 2
```

`NumericStepper` は、画面上の他のコンポーネントの前にも表示されます。

`numChildren` は表示リスト内のオブジェクトの数であり (1 ~ n)、リスト内の最初の位置は 0 です。そのため、リスト内に3つのオブジェクトがある場合、3番目のオブジェクトのインデックス位置は 2 になります。したがって、表示リストの最後の位置、または表示深度の一番上のオブジェクトは、`numChildren - 1` として参照できます。

表示リストからのコンポーネントの削除

表示オブジェクトコンテナおよびその表示リストからコンポーネントを削除するには、`removeChild()` メソッドと `removeChildAt()` メソッドを使用します。次の例は、3つの `Button` コンポーネントをステージ上で互いの前に配置し、それぞれのイベントリスナーを追加します。各 `Button` をクリックすると、イベントハンドラーがその `Button` を表示リストおよびステージから削除します。

- 1 新しい Flash ファイル (ActionScript 3.0) ドキュメントを作成します。
- 2 `Button` をコンポーネントパネルからライブラリパネルまでドラッグします。
- 3 アクションパネルを開き、メインタイムラインのフレーム 1 を選択して、次のコードを追加します。

```
import fl.controls.Button;

var i:int = 0;
while(i++ < 3) {
    makeButton(i);
}
function removeButton(event:MouseEvent):void {
    removeChildAt(numChildren -1);
}
function makeButton(num) {
    var aButton:Button = new Button();
    aButton.name = "Button" + num;
    aButton.label = aButton.name;
    aButton.move(200, 200);
    addChild(aButton);
    aButton.addEventListener(MouseEvent.CLICK, removeButton);
}
```

表示リストの詳細については、『**ActionScript 3.0 のプログラミング**』の「表示のプログラミング」を参照してください。

FocusManager の操作

ユーザーが `Tab` キーを押して Flash アプリケーション内を移動したり、アプリケーション内をクリックしたりすると、`FocusManager` クラスは入力フォーカスを受け取るコンポーネントを判別します。コンポーネントを作成している場合を除いて、`FocusManager` を有効にするために `FocusManager` インスタンスをアプリケーションに追加したり、コードを作成したりする必要はありません。

`RadioButton` オブジェクトがフォーカスを取得した場合、`FocusManager` は、そのオブジェクトと、共通の `groupName` 値を持つすべてのオブジェクトを検査し、`selected` プロパティが `true` であるオブジェクトにのみフォーカスを設定します。

各モーダル `Window` コンポーネントは `FocusManager` のインスタンスを備えており、そのウィンドウ上のコントロールは独自のタブセットになります。そのため、ユーザーが `Tab` キーを押して、意図せずに他のウィンドウ上にあるコンポーネントに移動することはありません。

`FocusManager` では、コンテナ内のエレメントの深度 (z 順序) が、デフォルトのナビゲーション順序 (タブループ) として使用されます。通常、タブループ内の移動には `Tab` キーを使用します。`Tab` キーを押すごとに、フォーカスは先頭の (フォーカスを持つ) コンポーネントから最後のコンポーネントまで順に移動し、再び先頭のコンポーネントに戻ります。深度は最初、コンポーネントがステージにドラッグされた順序によって設定されます。ただし、修正/重ね順/「最前面へ」または「最背面へ」コマンドを使用して、最終的な z 順序を決定することもできます。深度について詳しくは、24 ページの「表示リストの操作」を参照してください。

アプリケーションでコンポーネントインスタンスにフォーカスを設定するには、`setFocus()` メソッドを呼び出します。例えば、次の例では現在のコンテナ (`this`) の `FocusManager` インスタンスを作成し、`Button` インスタンス `aButton` にフォーカスを設定します。

```
var fm:FocusManager = new FocusManager(this);  
fm.setFocus(aButton);
```

getFocus() メソッドを呼び出すことで、フォーカスのあるコンポーネントを特定できます。また、タブグループ内で次にフォーカスを取得するコンポーネントを特定するには、getNextFocusManagerComponent() メソッドを呼び出します。次の例では、ステージ上に CheckBox、RadioButton および Button があり、各コンポーネントに MouseEvent.CLICK イベントおよび FocusEvent.MOUSE_FOCUS_CHANGE イベントのリリスナがあります。ユーザーがコンポーネントをクリックして MouseEvent.CLICK イベントが発生すると、showFocus() 関数が getNextFocusManagerComponent() メソッドを呼び出し、タブグループ内で次にフォーカスを取得するコンポーネントを特定します。次に、setFocus() メソッドを呼び出して、そのコンポーネントにフォーカスを設定します。FocusEvent.MOUSE_FOCUS_CHANGE イベントが発生すると、fc() 関数がこのイベントが発生したコンポーネントの名前を表示します。このイベントは、タブグループで次の順番でないコンポーネントをユーザーがクリックしたときにトリガーされます。

```
// This example assumes a CheckBox (aCh), a RadioButton (aRb) and a Button  
// (aButton) have been placed on the Stage.
```

```
import fl.managers.FocusManager;  
import flash.display.InteractiveObject;  
  
var fm:FocusManager = new FocusManager(this);  
  
aCh.addEventListener(MouseEvent.CLICK, showFocus);  
aRb.addEventListener(MouseEvent.CLICK, showFocus);  
aButton.addEventListener(MouseEvent.CLICK, showFocus);  
aCh.addEventListener(FocusEvent.MOUSE_FOCUS_CHANGE, fc);  
aRb.addEventListener(FocusEvent.MOUSE_FOCUS_CHANGE, fc);  
aButton.addEventListener(FocusEvent.MOUSE_FOCUS_CHANGE, fc);  
  
function showFocus(event:MouseEvent):void {  
    var nextComponent:InteractiveObject = fm.getNextFocusManagerComponent();  
    trace("Next component in tab loop is: " + nextComponent.name);  
    fm.setFocus(nextComponent);  
}  
  
function fc(fe:FocusEvent):void {  
    trace("Focus Change: " + fe.target.name);  
}
```

ユーザーが Enter キー (Windows) または Return キー (Macintosh) を押した場合にフォーカスを取得するボタンを作成するには、次のコードのように、デフォルトボタンにする Button インスタンスを FocusManager.defaultButton プロパティに設定します。

```
import fl.managers.FocusManager;  
  
var fm:FocusManager = new FocusManager(this);  
fm.defaultButton = okButton;
```

FocusManager クラスは、Flash Player のデフォルトのフォーカス領域を表す矩形を無効にして、角の丸いカスタムのフォーカス用矩形を描画します。

Flash アプリケーションにおけるフォーカススキームの作成について詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の FocusManager クラスを参照してください。カスタムフォーカスマネージャーを作成するには、IFocusManager インターフェイスを実装するクラスを作成する必要があります。詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の IFocusManager を参照してください。

リストベースのコンポーネントの操作

List、DataGrid、TileList コンポーネントはすべて SelectableList 基本クラスから継承されます。そのため、これらのコンポーネントはリストベースのコンポーネントと考えられます。ComboBox は、テキストボックスとリストで構成されるので、やはりリストベースのコンポーネントです。

List は行で構成されます。DataGrid と TileList は、複数の列に分割できる行で構成されます。行と列が交差する部分がセルです。単一列の行である List では、各行がセルになります。セルには次に示す 2 つの重要な特徴があります。

- セルに格納されるデータ値はアイテムと呼ばれます。アイテムとは、List 内の情報の単位を格納するために使用される ActionScript オブジェクトです。List は配列のようなものと考えられます。配列内の個々のインデックス付き領域に相当するのがアイテムです。List 内では、アイテムは通常、表示される label プロパティと、データを格納する data プロパティを備えたオブジェクトです。データプロバイダーは、List 内のアイテムのデータモデルです。データプロバイダーを使用すると、コンポーネントの dataProvider プロパティに割り当てただけで、リストベースのコンポーネントにデータを設定することができます。
- セルには、テキストからイメージ、ムービークリップまでのさまざまなタイプのデータや、ユーザーが作成できるすべてのクラスを格納できます。そのため、セルはコンテンツに適した方法で描画またはレンダリングする必要があります。したがって、リストベースのコンポーネントは、セルをレンダリングするためのセルレンダラーを備えています。DataGrid の場合は、各列が DataGridColumn オブジェクトであり、cellRenderer プロパティを含んでいるので、コンテンツに合わせて各列を適切にレンダリングすることができます。

すべてのリストベースのコンポーネントに cellRenderer プロパティと dataProvider プロパティがあり、ユーザーはそれぞれを設定して、各コンポーネントのセルをロードおよびレンダリングすることができます。これらのプロパティの使用とリストベースのコンポーネントの操作について詳しくは、28 ページの「[DataProvider の使用](#)」および 35 ページの「[CellRenderer の操作](#)」を参照してください。

DataProvider の使用

DataProvider は、ComboBox、DataGrid、List、TileList コンポーネントにデータを供給するために使用できるデータソースです。これらの各コンポーネントクラスには dataProvider プロパティがあり、それらのプロパティに DataProvider オブジェクトを割り当てて、コンポーネントのセルにデータを設定できます。通常、データプロバイダーは Array や XML オブジェクトなどのデータのコレクションです。

DataProvider の作成

ComboBox、List および TileList の各コンポーネントの場合は、オーサリング環境で dataProvider パラメーターを使用することによって、DataProvider を作成できます。DataGrid コンポーネントの場合、プロパティインスペクターに dataProvider パラメーターがありません。これは、DataGrid コンポーネントは複数の列を持つことができるので、データプロバイダーが複雑になるからです。ActionScript を使用してこれらのコンポーネント、および DataGrid の DataProvider を作成することもできます。

dataProvider パラメーターの使用

プロパティインスペクターまたはコンポーネントインスペクターの「パラメーター」タブで dataProvider パラメーターをクリックして、ComboBox、List、TileList コンポーネントの簡単なデータプロバイダーを作成できます。

最初は空の配列を表示している「値」セルをダブルクリックすると、値ダイアログボックスが開きます。このダイアログボックスに複数のラベルおよびデータ値を入力し、データプロバイダーを作成できます。



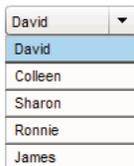
dataProvider の値ダイアログボックス

プラス記号をクリックしてアイテムを `dataProvider` に追加します。マイナス記号をクリックすると、アイテムが削除されます。上矢印をクリックすると選択したアイテムがリストの上方に移動し、下矢印をクリックすると下方に移動します。次の図は、子供の名前と誕生日のリストを作成する値ダイアログボックスを示します。



データを設定した値ダイアログボックス

ユーザーが作成する配列は、ラベルフィールドと値フィールドのペアで構成されています。ラベルフィールドは `label` と `data` であり、値フィールドは子供の名前と誕生日です。ラベルフィールドには、List に表示されるコンテンツを指定します。この場合は、子供の名前です。ComboBox はこのように表示されます。



DataProvider でデータを設定した ComboBox

データの追加が終了したら、「OK」をクリックしてダイアログボックスを閉じます。これで `dataProvider` パラメーターの配列にユーザーが作成したアイテムが設定されました。



データが設定された dataProvider パラメーター

ユーザーが作成したラベルとデータ値にアクセスするには、ActionScript を使用して、コンポーネントの dataProvider プロパティにアクセスします。

ActionScript を使用した DataProvider の作成

DataProvider は、Array または XML オブジェクトにデータを作成し、そのオブジェクトを value パラメーターとして DataProvider コンストラクターに指定することによって作成できます。

注意: ActionScript 3.0 では、Array または XML オブジェクトを dataProvider プロパティに直接割り当てることができません。これは、このプロパティが DataProvider オブジェクトとして定義されており、取得できるのが DataProvider 型のオブジェクトに限られるためです。

次の例では、単一列の行である List コンポーネントに数人の子供の名前と誕生日を設定します。この例は、items Array にリストを定義し、DataProvider インスタンスを作成するときにパラメーターとして指定し (new DataProvider(items))、List コンポーネントの dataProvider プロパティに割り当てます。

```
import fl.controls.List;
import fl.data.DataProvider;

var aList:List = new List();
var items:Array = [
    {label:"David", data:"11/19/1995"},
    {label:"Colleen", data:"4/20/1993"},
    {label:"Sharon", data:"9/06/1997"},
    {label:"Ronnie", data:"7/6/1993"},
    {label:"James", data:"2/15/1994"},
];
aList.dataProvider = new DataProvider(items);
addChild(aList);
aList.move(150,150);
```

Array は、ラベルフィールドと値フィールドのペアで構成されています。ラベルフィールドは label と data であり、値フィールドは子供の名前と誕生日です。ラベルフィールドには、List に表示されるコンテンツを指定します。この場合は、子供の名前です。List は次のように表示されます。

David
Colleen
Sharon
Ronnie
James

DataProvider でデータを設定した List

データフィールドの値は、ユーザーがリスト内のアイテムをクリックして選択し、change イベントが発生したときに利用可能になります。次の例は、TextArea (aTa) とイベントハンドラー (changeHandler) を前の例に追加し、ユーザーがリスト内の名前を選択したときに、その子供の誕生日を表示します。

```
import fl.controls.List;
import fl.controls.TextArea;
import flash.events.Event;
import fl.data.DataProvider;

var aList:List = new List();
var aTa:TextArea = new TextArea();
var items:Array = [
    {label:"David", data:"1/19/1995"},
    {label:"Colleen", data:"4/20/1993"},
    {label:"Sharon", data:"9/06/1994"},
    {label:"Ronnie", data:"7/6/1993"},
    {label:"James", data:"2/15/1994"},
];
aList.dataProvider = new DataProvider(items);

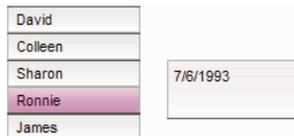
addChild(aList);
addChild(aTa);

aList.move(150,150);
aTa.move(150, 260);

aList.addEventListener(Event.CHANGE, changeHandler);

function changeHandler(event:Event):void {
    aTa.text = event.target.selectedItem.data;
};
```

これで、ユーザーが List 内の子供の名前を選択したとき、次の図のように、その子供の誕生日が TextArea に表示されます。これは、changeHandler() 関数が、TextArea の text プロパティ (aTa.text) を、選択されたアイテムのデータフィールドの値 (event.target.selectedItem.data) に設定することによって実現します。event.target プロパティは、イベントをトリガーしたオブジェクトであり、この場合は List です。



List の DataProvider からのデータフィールドの表示

DataProvider にはテキスト以外のデータも含めることができます。次の例では、TileList にデータを供給する DataProvider に MovieClip が含まれています。これは、色付きのボックスの MovieClip を作成した後に addItem() を呼び出して各アイテムを追加することによって DataProvider を構築します。

```
import fl.data.DataProvider;
import flash.display.DisplayObject;

var aBox:MovieClip = new MovieClip();
var i:uint = 0;
var colors:Array = new Array(0x000000, 0xFF0000, 0x0000CC, 0x00CC00, 0xFFFF00);
var colorNames:Array = new Array("Midnight", "Cranberry", "Sky", "Forest", "July");
var dp:DataProvider = new DataProvider();
for(i=0; i < colors.length; i++) {
    drawBox(aBox, colors[i]); // draw box w next color in array
    dp.addItem( {label:colorNames[i], source:aBox} );
}
aTl.dataProvider = dp;
aTl.columnWidth = 110;
aTl.rowHeight = 130;
aTl.setSize(280,150);
aTl.move(150, 150);
aTl.setStyle("contentPadding", 5);

function drawBox(box:MovieClip,color:uint):void {
    box.graphics.beginFill(color, 1.0);
    box.graphics.drawRect(0, 0, 100, 100);
    box.graphics.endFill();
}
```

XML データを (配列の代わりに) 使用して、DataProvider オブジェクトにデータを設定することもできます。例えば、次のコードは、employeesXML という名前の XML オブジェクトにデータを格納し、そのオブジェクトを DataProvider() コンストラクター関数の値パラメーターとして渡します。

```
import fl.controls.DataGrid;
import fl.data.DataProvider;

var aDg:DataGrid = new DataGrid();
addChild(aDg);

var employeesXML:XML =
    <employees>
        <employee Name="Edna" ID="22" />
        <employee Name="Stu" ID="23" />
    </employees>;

var myDP:DataProvider = new DataProvider(employeesXML);

aDg.columns = ["Name", "ID"];
aDg.dataProvider = myDP;
```

前のコードのように、データを XML データの属性として指定すること、または次のコードのように XML データのプロパティとして指定することができます。

```
var employeesXML:XML =
    <employees>
        <employee>
            <Name>Edna</Name>
            <ID>22</ID>
        </employee>
        <employee>
            <Name>Stu</Name>
            <ID>23</ID>
        </employee>
    </employees>;
```

DataProvider にはメソッドとプロパティのセットもあり、ユーザーはそれを使用して DataProvider にアクセスし、操作できます。DataProvider API を使用して、DataProvider のアイテムの追加、削除、置換、並べ替えおよび結合を行うことができます。

DataProvider の操作

addItem() メソッドと addItemAt() メソッドを使用して、DataProvider にアイテムを追加できます。次の例は、ユーザーが編集可能な ComboBox のテキストフィールドに入力するアイテムを追加します。ComboBox をステージ上にドラッグし、aCb というインスタンス名を指定するとします。

```
import fl.data.DataProvider;
import fl.events.ComponentEvent;

var items:Array = [
    {label:"Roger"},
    {label:"Carolyn"},
    {label:"Darrell"},
    {label:"Rebecca"},
    {label:"Natalie"},
    {label:"Mitchell"},
];
aCb.dataProvider = new DataProvider(items);

aCb.addEventListener(ComponentEvent.ENTER, newItemHandler);

function newItemHandler(event:ComponentEvent):void {
    var newRow:int = event.target.length + 1;
    event.target.addItemAt({label:event.target.selectedLabel},
        event.target.length);
}
```

DataProvider を通じてコンポーネント内のアイテムを置換および削除することもできます。次の例では、2つの別々の List コンポーネント、listA および listB を実装し、Sync というラベルの Button を指定します。この例では、ユーザーがこのボタンをクリックしたときに、replaceItemAt() メソッドを使用して listB のアイテムを listA のアイテムに置換します。listA が listB より長い場合は、addItem() メソッドを呼び出して、不足分のアイテムを listB に追加します。listB が listA より長い場合は、removeItemAt() メソッドを呼び出して、ListB の余分のアイテムを削除します。

```
// Requires the List and Button components to be in the library

import fl.controls.List;
import fl.controls.Button;
import flash.events.Event;
import fl.data.DataProvider;

var listA:List = new List();
var listB:List = new List();
var syncButton:Button = new Button();
syncButton.label = "Sync";

var itemsA:Array = [
    {label:"David"},
    {label:"Colleen"},
    {label:"Sharon"},
    {label:"Ronnie"},
    {label:"James"},
];
var itemsB:Array = [
    {label:"Roger"},
    {label:"Carolyn"},
    {label:"Darrell"},
    {label:"Rebecca"},
    {label:"Natalie"},
    {label:"Mitchell"},
];
listA.dataProvider = new DataProvider(itemsA);
listB.dataProvider = new DataProvider(itemsB);
```

```

addChild(listA);
addChild(listB);
addChild(syncButton);

listA.move(100, 100);
listB.move(250, 100);
syncButton.move(175, 220);

syncButton.addEventListener(MouseEvent.CLICK, syncHandler);

function syncHandler(event:MouseEvent):void {
    var i:uint = 0;
    if(listA.length > listB.length) { //if listA is longer, add items to B
        while(i < listB.length) {
            listB.dataProvider.replaceItemAt(listA.dataProvider.getItemAt(i), i);
            ++i;
        }
        while(i < listA.length) {
            listB.dataProvider.addItem(listA.dataProvider.getItemAt(i++));
        }
    } else if(listA.length == listB.length) { //if listA and listB are equal length
        while(i < listB.length) {
            listB.dataProvider.replaceItemAt(listA.dataProvider.getItemAt(i), i);
            ++i;
        }
    } else { //if listB is longer, remove extra items from B
        while(i < listA.length) {
            listB.dataProvider.replaceItemAt(listA.dataProvider.getItemAt(i), i);
            ++i;
        }
        while(i < listB.length) {
            listB.dataProvider.removeItemAt(i++);
        }
    }
}
}

```

merge()、sort() および sortOn() の各メソッドを使用して、DataProvider の結合と並べ替えを行うこともできます。次の例では、2つの DataGrid インスタンス (aDg と bDg) に2つのソフトボールチームの名簿の一部を設定します。Merge というラベルのボタンを追加し、ユーザーがそのボタンをクリックしたときに、イベントハンドラー (mrgHandler) が bDg の名簿と aDg の名簿を結合し、得られた DataGrid を Name 列の内容で並べ替えます。

```

import fl.data.DataProvider;
import fl.controls.DataGrid;
import fl.controls.Button;

var aDg:DataGrid = new DataGrid();
var bDg:DataGrid = new DataGrid();
var mrgButton:Button = new Button();
addChild(aDg);
addChild(bDg);
addChild(mrgButton);
bldRosterGrid(aDg);
bldRosterGrid(bDg);
var aRoster:Array = new Array();
var bRoster:Array = new Array();
aRoster = [
    {Name:"Wilma Carter", Bats:"R", Throws:"R", Year:"So", Home: "Redlands, CA"},
    {Name:"Sue Pennypacker", Bats:"L", Throws:"R", Year:"Fr", Home: "Athens, GA"},
    {Name:"Jill Smithfield", Bats:"R", Throws:"L", Year:"Sr", Home: "Spokane, WA"},
    {Name:"Shirley Goth", Bats:"R", Throws:"R", Year:"Sr", Home: "Carson, NV"}
];

```

```
bRoster = [
    {Name:"Angelina Davis", Bats:"R", Throws:"R", Year:"So", Home: "Odessa, TX"},
    {Name:"Maria Santiago", Bats:"L", Throws:"L", Year:"Sr", Home: "Tacoma, WA"},
    {Name:"Debbie Ferguson", Bats:"R", Throws:"R", Year: "Jr", Home: "Bend, OR"}
];
aDg.dataProvider = new DataProvider(aRoster);
bDg.dataProvider = new DataProvider(bRoster);
aDg.move(50,50);
aDg.rowCount = aDg.length;
bDg.move(50,200);
bDg.rowCount = bDg.length;
mrgButton.label = "Merge";
mrgButton.move(200, 315);
mrgButton.addEventListener(MouseEvent.CLICK, mrgHandler);

function bldRosterGrid(dg:DataGrid){
    dg.setSize(400, 300);
    dg.columns = ["Name", "Bats", "Throws", "Year", "Home"];
    dg.columns[0].width = 120;
    dg.columns[1].width = 50;
    dg.columns[2].width = 50;
    dg.columns[3].width = 40;
    dg.columns[4].width = 120;
};

function mrgHandler(event:MouseEvent):void {
    aDg.dataProvider.merge(bDg.dataProvider);
    aDg.dataProvider.sortOn("Name");
}
```

詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の DataProvider クラスを参照してください。

CellRenderer の操作

CellRenderer は、List、DataGrid、TileList、ComboBox などのリストベースのコンポーネントが、行のカスタムセルコンテンツを操作および表示するために使用するクラスです。カスタマイズされたセルには、テキスト、CheckBox などの作成済みコンポーネント、またはユーザーが作成できるすべての表示オブジェクトクラスを入れることができます。カスタム CellRenderer を使用してデータをレンダリングするには、CellRenderer クラスを拡張するか、ICellRenderer インターフェイスを実装して独自のカスタム CellRenderer クラスを作成します。

List、DataGrid、TileList、ComboBox クラスは、SelectableList クラスのサブクラスです。SelectableList クラスは cellRenderer スタイルを含みます。このスタイルは、コンポーネントがセルのレンダリングに使用する表示オブジェクトを定義します。

CellRenderer が使用するスタイルの形式を調整するには、List オブジェクトの setRendererStyle() メソッドを呼び出します (35 ページの「[セルの形式の制御](#)」を参照)。また、カスタムクラスを定義して CellRenderer として使用することもできます (36 ページの「[カスタム CellRenderer クラスの定義](#)」を参照)。

セルの形式の制御

CellRenderer クラスには多くのスタイルが含まれており、ユーザーはそれらを使用してセルの形式を制御できます。

次のスタイルを使用すると、さまざまな状態 (無効、ダウン、オーバー、アップ) のセルに使用されるスキンを定義できます。

- disabledSkin および selectedDisabledSkin

- downSkin および selectedDownSkin
- overSkin および selectedOverSkin
- upSkin および selectedUpSkin

次のスタイルはテキスト形式に適用されます。

- disabledTextFormat
- textFormat
- textPadding

これらのスタイルを設定するには、List オブジェクトの `setRendererStyle()` メソッドを呼び出すか、CellRenderer オブジェクトの `setStyle()` メソッドを呼び出します。これらのスタイルを取得するには、List オブジェクトの `getRendererStyle()` メソッドを呼び出すか、CellRenderer オブジェクトの `getStyle()` メソッドを呼び出します。List オブジェクトの `rendererStyles` プロパティまたは CellRenderer オブジェクトの `getStyleDefinition()` メソッドを通じて、すべてのレンダラースタイルを定義するオブジェクトに（オブジェクトの指定プロパティとして）アクセスすることもできます。

`clearRendererStyle()` メソッドを呼び出して、スタイルをデフォルト値にリセットできます。

リスト内の行の高さを取得したり設定したりするには、List オブジェクトの `rowHeight` プロパティを使用します。

カスタム CellRenderer クラスの定義

CellRenderer クラスを拡張するクラスの作成によるカスタム CellRenderer の定義

例えば、次のコードには 2 つのクラスが含まれています。ListSample クラスは List コンポーネントをインスタンス化し、他のクラスの CustomRenderer を使用して List コンポーネントに使用するセルレンダラーを定義します。CustomRenderer クラスは CellRenderer クラスを拡張します。

- 1 ファイル／新規を選択します。
- 2 表示される新規ドキュメントダイアログボックスで、「Flash ファイル (ActionScript 3.0)」を選択して、「OK」をクリックします。
- 3 ウィンドウ／コンポーネントを選択して、コンポーネントパネルを表示します。
- 4 コンポーネントパネルで、List コンポーネントをステージまでドラッグします。
- 5 プロパティインスペクターが表示されていない場合は、ウィンドウ／プロパティ／プロパティを選択します。
- 6 List コンポーネントを選択した状態で、プロパティインスペクターでプロパティを設定します。

- インスタンス名：myList
- W（幅）：200
- H（高さ）：300
- X：20
- Y：20

- 7 タイムラインのレイヤー 1 のフレーム 1 を選択し、ウィンドウ／アクションを選択します。
- 8 アクションパネルで、次のスクリプトを入力します。

```
myList.setStyle("cellRenderer", CustomCellRenderer);  
myList.addItem({label:"Burger -- $5.95"});  
myList.addItem({label:"Fries -- $1.95"});
```

- 9 ファイル／保存を選択します。ファイル名を指定し、「OK」ボタンをクリックします。
- 10 ファイル／新規を選択します。

11 表示される新規ドキュメントダイアログボックスで、「ActionScript ファイル」を選択して、「OK」をクリックします。

12 スクリプトウィンドウで、次のコードを入力し、CustomCellRenderer クラスを定義します。

```
package {
    import fl.controls.listClasses.CellRenderer;
    import flash.text.TextFormat;
    import flash.filters.BevelFilter;
    public class CustomCellRenderer extends CellRenderer {
        public function CustomCellRenderer() {
            var format:TextFormat = new TextFormat("Verdana", 12);
            setStyle("textFormat", format);
            this.filters = [new BevelFilter()];
        }
    }
}
```

13 ファイル/保存を選択します。CustomCellRenderer.as ファイルに名前を付けて、FLA ファイルと同じディレクトリに入れ、「OK」をクリックします。

14 制御/ムービープレビューを選択します。

ICellRenderer インターフェイスを実装するクラスの使用によるカスタム CellRenderer の定義

DisplayObject クラスを継承し、ICellRenderer インターフェイスを実装するいずれかのクラスを使用して CellRenderer を定義することもできます。例えば、次のコードは 2 つのクラスを定義します。ListSample2 クラスは List オブジェクトを表示リストに追加し、その CellRenderer を定義して CustomRenderer クラスを使用します。CustomRenderer クラスは CheckBox クラスを拡張し (CheckBox クラスは DisplayObject クラスを拡張する)、ICellRenderer インターフェイスを実装します。CustomRenderer クラスは、ICellRenderer インターフェイスに定義された data プロパティと listData プロパティに関する getter メソッドと setter メソッドを定義します。ICellRenderer インターフェイスに定義されたその他のプロパティとメソッド (selected プロパティと setSize() メソッド) は既に CheckBox クラスに定義されています。

1 ファイル/新規を選択します。

2 表示される新規ドキュメントダイアログボックスで、「Flash ファイル (ActionScript 3.0)」を選択して、「OK」をクリックします。

3 ウィンドウ/コンポーネントを選択して、コンポーネントパネルを表示します。

4 コンポーネントパネルで、List コンポーネントをステージまでドラッグします。

5 プロパティインスペクターが表示されていない場合は、ウィンドウ/プロパティ/プロパティを選択します。

6 List コンポーネントを選択した状態で、プロパティインスペクターでプロパティを設定します。

- インスタンス名: myList
- W (幅): 100
- H (高さ): 300
- X: 20
- Y: 20

7 タイムラインのレイヤー 1 のフレーム 1 を選択し、ウィンドウ/アクションを選択します。

8 アクションパネルで、次のスクリプトを入力します。

```
myList.setStyle("cellRenderer", CustomCellRenderer);
myList.addItem({name:"Burger", price:"$5.95"});
myList.addItem({name:"Fries", price:"$1.95"});
```

9 ファイル/保存を選択します。ファイル名を指定し、「OK」ボタンをクリックします。

10 ファイル/新規を選択します。

11 表示される新規ドキュメントダイアログボックスで、「ActionScript ファイル」を選択して、「OK」をクリックします。

12 スクリプトウィンドウで、次のコードを入力し、CustomCellRenderer クラスを定義します。

```
package
{
    import fl.controls.CheckBox;
    import fl.controls.listClasses.ICellRenderer;
    import fl.controls.listClasses.ListData;
    public class CustomCellRenderer extends CheckBox implements ICellRenderer {
        private var _listData:ListData;
        private var _data:Object;
        public function CustomCellRenderer() {
        }
        public function set data(d:Object):void {
            _data = d;
            label = d.label;
        }
        public function get data():Object {
            return _data;
        }
        public function set listData(ld:ListData):void {
            _listData = ld;
        }
        public function get listData():ListData {
            return _listData;
        }
    }
}
```

13 ファイル／保存を選択します。CustomCellRenderer.as ファイルに名前を付けて、FLA ファイルと同じディレクトリに入れ、「OK」をクリックします。

14 制御／ムービープレビューを選択します。

シンボルを使用した CellRenderer の定義

ライブラリのシンボルを使用して CellRenderer を定義することもできます。シンボルを ActionScript 用に書き出す必要があります。また、ライブラリシンボルのクラス名に、ICellRenderer インターフェイスを実装するか、CellRenderer クラス（またはそのサブクラスの 1 つ）を拡張するクラスファイルが関連付けられていることが必要です。

次の例では、ライブラリシンボルを使用してカスタム CellRenderer を定義します。

- 1 ファイル／新規を選択します。
- 2 表示される新規ドキュメントダイアログボックスで、「Flash ファイル (ActionScript 3.0)」を選択して、「OK」をクリックします。
- 3 ウィンドウ／コンポーネントを選択して、コンポーネントパネルを表示します。
- 4 コンポーネントパネルで、List コンポーネントをステージまでドラッグします。
- 5 プロパティインスペクターが表示されていない場合は、ウィンドウ／プロパティ／プロパティを選択します。
- 6 List コンポーネントを選択した状態で、プロパティインスペクターでプロパティを設定します。
 - インスタンス名：myList
 - W (幅)：100
 - H (高さ)：400
 - X：20
 - Y:20

- 7 パラメーターパネルをクリックして、**dataProvider** 行の 2 列目をダブルクリックします。
- 8 表示される値ダイアログボックスで、プラス記号 (+) を 2 回クリックして (ラベルが **label0** と **label1** に設定された) 2 つのデータエレメントを追加し、「OK」をクリックします。
- 9 テキストツールで、ステージ上のテキストフィールドを描画します。
- 10 テキストフィールドを選択した状態で、プロパティインスペクターでプロパティを設定します。
 - テキストタイプ：ダイナミックテキスト
 - インスタンス名：**textField**
 - W (幅) :100
 - フォントサイズ：24
 - x : 0
 - Y : 0
- 11 テキストフィールドを選択した状態で、修正／シンボルに変換を選択します。
- 12 シンボルに変換ダイアログボックスで次の設定を行い、「OK」をクリックします。
 - 名前：**MyCellRenderer**
 - タイプ：**MovieClip**
 - ActionScript 用に書き出し：オン
 - 最初のフレームに書き出し：オン
 - クラス：**MyCellRenderer**
 - 基本クラス：**flash.display.MovieClip**

ActionScript クラス警告が表示されたら、警告ボックスの「OK」ボタンをクリックします。
- 13 新しいムービークリップシンボルのインスタンスをステージから削除します。
- 14 タイムラインのレイヤー 1 のフレーム 1 を選択し、ウィンドウ／アクションを選択します。
- 15 アクションパネルで、次のスクリプトを入力します。

```
myList.setStyle("cellRenderer", MyCellRenderer);
```
- 16 ファイル／保存を選択します。ファイル名を指定し、「OK」ボタンをクリックします。
- 17 ファイル／新規を選択します。
- 18 表示される「新規ドキュメント」ダイアログボックスで、「ActionScript ファイル」を選択して、「OK」をクリックします。
- 19 スクリプトウィンドウで、次のコードを入力し、**MyCellRenderer** クラスを定義します。

```
package {
    import flash.display.MovieClip;
    import flash.filters.GlowFilter;
    import flash.text.TextField;
    import fl.controls.listClasses.ICellRenderer;
    import fl.controls.listClasses.ListData;
    import flash.utils.setInterval;
    public class MyCellRenderer extends MovieClip implements ICellRenderer {
        private var _listData:ListData;
        private var _data:Object;
        private var _selected:Boolean;
        private var glowFilter:GlowFilter;
        public function MyCellRenderer() {
            glowFilter = new GlowFilter(0xFFFFF0);
            setInterval(toggleFilter, 200);
        }
        public function set data(d:Object):void {
            _data = d;
            textField.text = d.label;
        }
        public function get data():Object {
            return _data;
        }
        public function set listData(ld:ListData):void {
            _listData = ld;
        }
        public function get listData():ListData {
            return _listData;
        }
        public function set selected(s:Boolean):void {
            _selected = s;
        }
        public function get selected():Boolean {
            return _selected;
        }
        public function setSize(width:Number, height:Number):void {
        }
        public function setStyle(style:String, value:Object):void {
        }
        public function setMouseState(state:String):void{
        }
        private function toggleFilter():void {
            if (textField.filters.length == 0) {
                textField.filters = [glowFilter];
            } else {
                textField.filters = [];
            }
        }
    }
}
```

20 ファイル／保存を選択します。MyCellRenderer.as ファイルに名前を付けて、FLA ファイルと同じディレクトリに入れ、「OK」をクリックします。

21 制御／ムービープレビューを選択します。

CellRenderer プロパティ

data プロパティは、CellRenderer 用に設定されたすべてのプロパティを含むオブジェクトです。例えば、次のクラスは Checkbox クラスを拡張するカスタム CellRenderer を定義しており、data プロパティの setter 関数が、data.label の値を、CheckBox クラスから継承される label プロパティに渡しています。

```
public class CustomRenderer extends CheckBox implements ICellRenderer {
    private var _listData:ListData;
    private var _data:Object;
    public function CustomRenderer() {
    }
    public function set data(d:Object):void {
        _data = d;
        label = d.label;
    }
    public function get data():Object {
        return _data;
    }
    public function set listData(ld:ListData):void {
        _listData = ld;
    }
    public function get listData():ListData {
        return _listData;
    }
}
}
```

selected プロパティは、リスト内でセルが選択されているかどうかを定義します。

DataGrid オブジェクトの列への CellRenderer の適用

DataGrid オブジェクトは複数の列を持つことができ、ユーザーは各列に別々のセルレンダラーを指定できます。DataGrid の各列は DataGridColumn オブジェクトによって表され、DataGridColumn クラスは cellRenderer プロパティを含みません。このプロパティに、列の CellRenderer を定義できます。

編集可能なセルの CellRenderer の定義

DataGridCellEditor クラスは、DataGrid オブジェクトの編集可能なセルに使用されるレンダラーを定義します。これがセルのレンダラーになるのは、DataGrid オブジェクトの editable プロパティが true に設定されていて、ユーザーが編集するセルをクリックしたときです。編集可能なセルの CellRenderer を定義するには、DataGrid オブジェクトの columns 配列の各エレメントについて itemEditor プロパティを設定します。

CellRenderer としてのイメージ、SWF ファイル、ムービークリップの使用

ImageCell クラスは、CellRenderer のサブクラスであり、メインコンテンツがイメージ、SWF ファイル、またはムービークリップであるセルのレンダリングに使用されるオブジェクトを定義します。ImageCell クラスは、セルの表示を定義する次のスタイルを含みます。

- imagePadding — セルのエッジとイメージのエッジを隔てるパディング（ピクセル単位）
- selectedSkin — 選択された状態を示すために使用されるスキン
- textOverlayAlpha — セルラベルの背後のオーバーレイの不透明度
- textPadding — セルのエッジとテキストのエッジを隔てるパディング（ピクセル単位）

ImageCell クラスは、TileList クラスのデフォルト CellRenderer です。

コンポーネントをアクセス可能にする

Flash アプリケーションのビジュアルコンテンツは、画面の内容を音声出力するスクリーンリーダーを使用することにより、視覚障害のあるユーザーも利用できます。Flash アプリケーションをスクリーンリーダーからアクセス可能にする方法について詳しくは、『Flash ユーザーガイド』の第 18 章「アクセシビリティコンテンツの作成」を参照してください。

ActionScript 3.0 コンポーネントをスクリーンリーダーからアクセス可能にするには、その Accessibility クラスを読み込み、そのクラスの `enableAccessibility()` メソッドを呼び出す必要もあります。次の ActionScript 3.0 コンポーネントをスクリーンリーダーからアクセス可能にすることができます。

コンポーネント	Accessibility クラス
Button	ButtonAccImpl
CheckBox	CheckBoxAccImpl
ComboBox	ComboBoxAccImpl
List	ListAccImpl
RadioButton	RadioButtonAccImpl
TileList	TileListAccImpl

コンポーネントの Accessibility クラスは、`fl.accessibility` パッケージに含まれています。例えば、`CheckBox` をスクリーンリーダーからアクセス可能にするには、次のステートメントをアプリケーションに追加します。

```
import fl.accessibility.CheckBoxAccImpl;  
  
CheckBoxAccImpl.enableAccessibility();
```

1 つのコンポーネントに対してアクセシビリティ機能を有効化する処理は、インスタンスをいくつ作成するかに関係なく、一度だけ実行します。

注意：アクセシビリティ機能を有効化すると、必要なクラスがコンパイル時に取り込まれるので、ファイルサイズがわずかに増加します。

ほとんどのコンポーネントはキーボードからも操作が可能です。アクセス可能なコンポーネントの有効化とキーボードからの操作について詳しくは、43 ページの「[UI コンポーネントの使用](#)」のユーザー操作に関する節と、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」のアクセシビリティクラスを参照してください。

第4章：UI コンポーネントの使用

この章では、Flash に付属する ActionScript 3.0 ユーザーインターフェイス（UI）コンポーネントの使用方法について説明します。

Button コンポーネントの使用

Button コンポーネントは、サイズ変更可能な矩形のボタンです。ユーザーは、アプリケーション内でマウスやスペースバーを使用してこのボタンを押すことで、特定のアクションを開始できます。Button には、カスタムアイコンを追加できます。また、Button の動作をプッシュボタンからトグルボタンに変更することもできます。トグルボタンにした Button は、ユーザーが一度クリックすると押し込まれた状態になり、もう一度クリックすると元の状態に戻ります。

Button は、多くのフォームと Web アプリケーションに使用される基本的な要素です。ユーザーの操作でイベントを起動するような場所ならば、どこにでもボタンを配置できます。例えば、ほとんどのフォームには「送信」ボタンがあります。また、プレゼンテーションのアプリケーションでは「前」ボタンや「次」ボタンを使用します。

Button コンポーネントのユーザー操作

ボタンは、アプリケーション内で有効または無効にすることができます。無効状態のときは、マウスおよびキーボードからの入力を受け付けません。有効状態のボタンは、ユーザーがクリックするか Tab キーでナビゲートすると、フォーカスを取得します。フォーカスを取得している Button コンポーネントのインスタンスでは、次のキーを使用した操作が可能です。

キー	説明
Shift+Tab	フォーカスを前のオブジェクトに移動します。
スペースバー	ボタンを押して（または離して）、click イベントをトリガーします。
Tab	フォーカスを次のオブジェクトに移動します。
Enter/Return	ボタンが FocusManager のデフォルト Button として設定されている場合は、フォーカスを次のオブジェクトに移動します。

フォーカスの制御について詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の IFocusManager インターフェイスと FocusManager クラス、および 26 ページの「[FocusManager の操作](#)」を参照してください。

ライブプレビューでは、オーサリング時にプロパティインスペクターまたはコンポーネントインスペクターで各 Button インスタンスのパラメーターに加えた変更が反映されます。

注意：アイコンがボタンよりも大きい場合は、ボタンの境界線からはみ出します。

ボタンをアプリケーションのデフォルトプッシュボタン（ユーザーが Enter キーを押したときにクリックイベントを受け取るボタン）として指定するには、FocusManager.defaultButton を設定します。例えば、次のコードでは、submitButton という Button インスタンスがデフォルトボタンに設定されます。

```
FocusManager.defaultButton = submitButton;
```

Button コンポーネントをアプリケーションに追加する際に、次の ActionScript コード行を追加することで、そのコンポーネントをスクリーンリーダーが読み取れるように設定できます。

```
import fl.accessibility.ButtonAccImpl;  
  
ButtonAccImpl.enableAccessibility();
```

1つのコンポーネントに対してアクセシビリティ機能を有効化する処理は、インスタンスをいくつ作成するかに関係なく、一度だけ実行します。

Button コンポーネントのパラメーター

プロパティインスペクター（ウィンドウ／プロパティ／プロパティ）またはコンポーネントインスペクター（ウィンドウ／コンポーネントインスペクター）で Button の各インスタンスに設定できるオーサリングパラメーターは、**emphasized**、**label**、**labelPlacement**、**selected** および **toggle** です。これらの各パラメーターには、対応する同名の **ActionScript** プロパティがあります。これらのパラメーターに値を割り当てると、アプリケーション内でプロパティの初期状態が設定されます。**ActionScript** で設定したプロパティは、パラメーターで設定した値よりも優先されます。これらのパラメーターに設定できる値については、「**Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド**」の Button クラスを参照してください。

Button コンポーネントを使用したアプリケーションの作成

次の手順は、オーサリング時に Button コンポーネントをアプリケーションに追加する方法を示しています。この例では、Button をクリックしたときに ColorPicker コンポーネントの状態が変わります。

- 1 新しい Flash (ActionScript 3.0) ドキュメントを作成します。
- 2 Button コンポーネントをコンポーネントパネルからステージにドラッグし、プロパティインスペクターで次の値を入力します。
 - インスタンス名として **aButton** と入力します。
 - **label** パラメーターに **Show** と入力します。
- 3 ステージに ColorPicker を追加し、インスタンス名を **aCp** にします。
- 4 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次の **ActionScript** コードを入力します。

```
aCp.visible = false;  
  
aButton.addEventListener(MouseEvent.CLICK, clickHandler);  
  
function clickHandler(event:MouseEvent):void {  
  
    switch(event.currentTarget.label) {  
        case "Show":  
            aCp.visible = true;  
            aButton.label = "Disable";  
            break;  
        case "Disable":  
            aCp.enabled = false;  
            aButton.label = "Enable";  
            break;  
        case "Enable":  
            aCp.enabled = true;  
            aButton.label = "Hide";  
            break;  
        case "Hide":  
            aCp.visible = false;  
            aButton.label = "Show";  
            break;  
    }  
}
```

コードの2行目では、clickHandler() 関数を MouseEvent.CLICK イベントのイベントハンドラー関数として登録します。ユーザーが Button をクリックしてイベントが発生すると、clickHandler() 関数が Button の値に応じて次のいずれかのアクションを実行します。

- Show の場合は ColorPicker を表示し、Button のラベルを「無効」に変更します。
- Disable の場合は ColorPicker を無効にし、Button のラベルを「有効」に変更します。
- Enable の場合は ColorPicker を有効にし、Button のラベルを「非表示」に変更します。
- Hide の場合は ColorPicker を非表示にし、Button のラベルを「表示」に変更します。

5 制御/ムービープレビューを選択して、アプリケーションを実行します。

Button コンポーネントを使用したアプリケーションの作成

次の手順では、ActionScript を使用してトグルボタンにした Button を作成し、Button をクリックしたときに出力パネルにイベントタイプを表示します。この例では、クラスのコンストラクターを呼び出して Button インスタンスを作成し、addChild() メソッドを呼び出してそのインスタンスをステージに追加します。

1 新しい Flash (ActionScript 3.0) ドキュメントを作成します。

2 Button コンポーネントをコンポーネントパネルから現在のドキュメントのライブラリパネルにドラッグします。

これにより、ライブラリにコンポーネントが追加されます。ただし、アプリケーションには表示されません。

3 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次のコードを入力して Button インスタンスを作成します。

```
import fl.controls.Button;

var aButton:Button = new Button();
addChild(aButton);
aButton.label = "Click me";
aButton.toggle = true;
aButton.move(50, 50);
```

move() メソッドにより、ステージ上の 50 (X 座標)、50 (Y 座標) の位置にボタンが配置されます。

4 ここで、次の ActionScript を追加して、イベントリスナーとイベントハンドラー関数を作成します。

```
aButton.addEventListener(MouseEvent.CLICK, clickHandler);

function clickHandler(event:MouseEvent):void {
    trace("Event type: " + event.type);
}
```

5 制御/ムービープレビューを選択します。

ボタンをクリックすると、出力パネルに「イベントタイプ: click」というメッセージが表示されます。

CheckBox コンポーネントの使用

CheckBox は、選択状態と非選択状態がある正方形のボックスです。選択状態のときは、ボックス内にチェックマークが表示されます。CheckBox には、テキストラベルを追加して、上、下、左、右のいずれかの位置に表示することができます。

複数の CheckBox を使用すると、相互排他的ではない true または false 値のセットを収集できます。例えば、購入したいと思う自動車の種類に関する情報を収集するアプリケーションで複数の CheckBox を使用すると、色々な特徴をユーザーに選択させることができます。

CheckBox のユーザー操作

CheckBox は、アプリケーション内で有効または無効にすることができます。CheckBox が有効なときにユーザーが CheckBox (またはそのラベル) をクリックすると、CheckBox は入力フォーカスを取得し、押された状態になります。ユーザーがマウスボタンを押し下げ、そのままの状態でもマウスポインターを CheckBox (またはそのラベル) の領域外に移動すると、その CheckBox は入力フォーカスを保持したまま、最初の状態に戻ります。ユーザーが CheckBox 上でマウスボタンを離さない限り、CheckBox の状態は変化しません。さらに、CheckBox には、選択状態と非選択状態の 2 種類の無効状態があります。それぞれの状態を表すには `selectedDisabledSkin` および `disabledSkin` が使用されます。無効状態の CheckBox はマウスやキーボードで選択できません。

無効状態の CheckBox は、ユーザーの操作に関係なく無効状態の外観で表示されます。無効状態の CheckBox はマウスおよびキーボードからの入力を受け付けません。

CheckBox インスタンスは、ユーザーがクリックするか Tab キーでナビゲートすると、フォーカスを取得します。フォーカスを取得している CheckBox コンポーネントのインスタンスでは、次のキーを使用した操作が可能です。

キー	説明
Shift+Tab	フォーカスを前のエレメントに移動します。
スペースバー	コンポーネントを選択または選択解除して、change イベントをトリガーします。
Tab	フォーカスを次のエレメントに移動します。

フォーカスの制御について詳しくは、26 ページの「[FocusManager の操作](#)」および「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の `FocusManager` クラスを参照してください。

ライブプレビューでは、オーサリング時にプロパティインスペクターまたはコンポーネントインスペクターで各 CheckBox インスタンスのパラメーターに加えた変更が反映されます。

CheckBox コンポーネントをアプリケーションに追加する際に、次の ActionScript コード行を追加することで、そのコンポーネントをスクリーンリーダーが読み取れるように設定できます。

```
import fl.accessibility.CheckBoxAccImpl;
```

```
CheckBoxAccImpl.enableAccessibility();
```

1 つのコンポーネントに対してアクセシビリティ機能を有効化する処理は、そのコンポーネントのインスタンスをいくつ使用するかに関係なく、一度だけ実行します。

CheckBox コンポーネントのパラメーター

プロパティインスペクターまたはコンポーネントインスペクターで CheckBox コンポーネントの各インスタンスに設定できるオーサリングパラメーターは、`label`、`labelPlacement` および `selected` です。これらの各パラメーターには、対応する同名の ActionScript プロパティがあります。これらのパラメーターに設定できる値については、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の `CheckBox` クラスを参照してください。

CheckBox を使用したアプリケーションの作成

次の手順は、オーサリング時に CheckBox コンポーネントをアプリケーションに追加する方法を示しています。ここでは、ローン計算アプリケーションのフォームから抜粋したものを使用しています。このフォームには、ユーザーに持ち家があるかどうかを問う質問があり、「はい」と回答するための CheckBox が用意されています。「はい」と回答があった場合は、その家に関連する値を指定するための 2 つのラジオボタンを表示します。

CheckBox コンポーネントを使用したアプリケーションの作成

- 1 新しい Flash (ActionScript 3.0) ドキュメントを作成します。

- 2 CheckBox コンポーネントをコンポーネントパネルからステージまでドラッグします。
- 3 プロパティインスペクター内で次のように設定します。
 - インスタンス名として、**homeCh** と入力します。
 - 幅 (W) の値として、**140** と入力します。
 - label パラメーターに「**持ち家がある**」と入力します。
- 4 2つの RadioButton コンポーネントをコンポーネントパネルからステージにドラッグし、CheckBox の右側の下に配置します。プロパティインスペクターで、各コンポーネントについて以下の値を入力します。
 - それぞれのインスタンス名として **underRb** および **overRb** と入力します。
 - 両方の RadioButton の W (幅) パラメーターとして **120** と入力します。
 - label パラメーターの underRb に \$500,000 以下と入力します。
 - label パラメーターの overRb に \$500,000 より多いと入力します。
 - 両方の RadioButton の groupName パラメーターとして **valueGrp** と入力します。
- 5 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次の ActionScript コードを入力します。

```
homeCh.addEventListener(MouseEvent.CLICK, clickHandler);
underRb.enabled = false;
overRb.enabled = false;

function clickHandler(event:MouseEvent):void {
    underRb.enabled = event.target.selected;
    overRb.enabled = event.target.selected;
}
```

このコードでは、CLICK イベントのイベントハンドラーを作成しています。このイベントハンドラーは、homeCh CheckBox が選択された場合に underRb および overRb の RadioButton を有効にし、homeCh が選択されていない場合には、それらを無効にします。詳しくは、「**Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド**」の MouseEvent クラスを参照してください。

- 6 制御/ムービープレビューを選択します。

次の例は、前のアプリケーションを複製したものですが、ここでは ActionScript を使用して CheckBox と複数の RadioButton を作成します。

ActionScript を使用した Checkbox の作成

- 1 新しい Flash (ActionScript 3.0) ドキュメントを作成します。
- 2 CheckBox コンポーネントと RadioButton コンポーネントをコンポーネントパネルから現在のドキュメントのライブラリパネルにドラッグします。ライブラリパネルが表示されていない場合は、Ctrl + L キーを押すか、ウィンドウ/ライブラリを選択して、ライブラリパネルを表示します。

これにより、コンポーネントがアプリケーションで利用できるようになりますが、ステージ上には表示されません。
- 3 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次のコードを入力してコンポーネントインスタンスを作成し、配置します。

```
import fl.controls.CheckBox;
import fl.controls.RadioButton;

var homeCh:CheckBox = new CheckBox();
var underRb:RadioButton = new RadioButton();
var overRb:RadioButton = new RadioButton();
addChild(homeCh);
addChild(underRb);
addChild(overRb);
underRb.groupName = "valueGrp";
overRb.groupName = "valueGrp";
homeCh.move(200, 100);
homeCh.width = 120;
homeCh.label = "Own your home?";
underRb.move(220, 130);
underRb.enabled = false;
underRb.width = 120;
underRb.label = "Under $500,000?";
overRb.move(220, 150);
overRb.enabled = false;
overRb.width = 120;
overRb.label = "Over $500,000?";
```

このコードでは、CheckBox() および RadioButton() のコンストラクターを使用してそれらのコンポーネントを作成し、addChild() メソッドを使用してステージに配置しています。ステージ上のコンポーネントを移動するには、move() メソッドを使用します。

- 4 ここで、次の ActionScript を追加して、イベントリスナとイベントハンドラー関数を作成します。

```
homeCh.addEventListener(MouseEvent.CLICK, clickHandler);

function clickHandler(event:MouseEvent):void {
    underRb.enabled = event.target.selected;
    overRb.enabled = event.target.selected;
}
```

このコードでは、CLICK イベントのイベントハンドラーを作成しています。このイベントハンドラーは、homeCh CheckBox が選択された場合に underRb RadioButton と overRb RadioButton を有効にし、homeCh が選択されていない場合には、それらを無効にします。詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の MouseEvent クラスを参照してください。

- 5 制御/ムービープレビューを選択します。

ColorPicker コンポーネントの使用

ColorPicker コンポーネントを使用すると、ユーザーが色見本からカラーを選択できます。ColorPicker のデフォルトモードでは、四角形のボタンに単一のカラーが表示されます。ユーザーがボタンをクリックすると、現在選択されているカラーの 16 進数値を示すテキストフィールドと共に、利用できるカラーの一覧が色見本パネルに表示されます。

ColorPicker に表示するカラーは、カラー値を ColorPicker の colors プロパティに設定することで指定できます。

ColorPicker コンポーネントのユーザー操作

ColorPicker を使用すると、ユーザーがカラーを選択して、アプリケーション内の別のオブジェクトに適用できます。例えば、ユーザーが背景色やテキストの色などのアプリケーションの要素をパーソナライズできるようにする場合は、ColorPicker を組み込むと、ユーザーが選択したカラーを適用できます。

ユーザーがカラーを選択するには、パネルでそのカラーの色見本をクリックするか、テキストフィールドにそのカラーの 16 進数値を入力します。ユーザーがいずれかのカラーを選択すると、ColorPicker の selectedColor プロパティを使用して、アプリケーション内のテキストなどのオブジェクトにそのカラーを適用できます。

ColorPicker インスタンスは、ユーザーがポインターをその上に移動するか Tab キーで移動すると、フォーカスを取得します。ColorPicker の色見本パネルが表示されているときは、次のキーを使用した操作が可能です。

キー	説明
Home	色見本パネルで先頭のカラーを選択します。
上向き矢印	色見本パネルで 1 行上のアイテムを選択します。
下向き矢印	色見本パネルで 1 行下のアイテムを選択します。
右向き矢印	色見本パネルで 1 つ右のカラーを選択します。
左向き矢印	色見本パネルで 1 つ左のカラーを選択します。
End	色見本パネルで最後のカラーを選択します。

ColorPicker コンポーネントのパラメーター

プロパティインスペクターまたはコンポーネントインスペクターで ColorPicker の各インスタンスに設定できるオーサリングパラメーターは、selectedColor および showTextField です。これらの各パラメーターには、対応する同名の ActionScript プロパティがあります。これらのパラメーターに設定できる値については、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の ColorPicker クラスを参照してください。

ColorPicker コンポーネントを使用したアプリケーションの作成

次の例では、オーサリング時に、アプリケーションに ColorPicker コンポーネントを追加します。この例では、ColorPicker でカラーを変更するたびに、changeHandler() 関数が drawBox() 関数を呼び出し、ColorPicker で選択されたカラーの新しいボックスを描画します。

- 1 新しい Flash (ActionScript 3.0) ドキュメントを作成します。
- 2 ColorPicker をコンポーネントパネルからステージの中央までドラッグし、インスタンス名を **aCp** にします。
- 3 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次の ActionScript コードを入力します。

```
import fl.events.ColorPickerEvent;

var aBox:MovieClip = new MovieClip();
drawBox(aBox, 0xFF0000);//draw a red box
addChild(aBox);

aCp.addEventListener(ColorPickerEvent.CHANGE,changeHandler);

function changeHandler(event:ColorPickerEvent):void {
    drawBox(aBox, event.target.selectedColor);
}

function drawBox(box:MovieClip,color:uint):void {
    box.graphics.beginFill(color, 1);
    box.graphics.drawRect(100, 150, 100, 100);
    box.graphics.endFill();
}
```

- 4 制御/ムービープレビューを選択します。
- 5 ColorPicker をクリックし、ボックスに塗るカラーを選択します。

ActionScript を使用した ColorPicker の作成

この例では、ColorPicker() コンストラクターと addChild() を使用して、ステージに ColorPicker を作成します。ColorPicker で表示するカラーを指定するために、colors プロパティを、赤色 (0xFF0000)、緑色 (0x00FF00) および青色 (0x0000FF) のカラー値に設定しています。さらに TextArea を作成して、ColorPicker で別のカラーを選択するたびに、その TextArea 内のテキストが同じカラーになるように変更します。

- 1 新しい Flash (ActionScript 3.0) ドキュメントを作成します。
- 2 ColorPicker コンポーネントをコンポーネントパネルからライブラリパネルにドラッグします。
- 3 TextArea コンポーネントをコンポーネントパネルからライブラリパネルにドラッグします。
- 4 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次の ActionScript コードを入力します。

```
import fl.controls.ColorPicker;
import fl.controls.TextArea;
import fl.events.ColorPickerEvent;

var aCp:ColorPicker = new ColorPicker();
var aTa:TextArea = new TextArea();
var aTf:TextFormat = new TextFormat();

aCp.move(100, 100);
aCp.colors = [0xff0000, 0x00ff00, 0x0000ff];
aCp.addEventListener(ColorPickerEvent.CHANGE, changeHandler);

aTa.text = "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus quis nisl vel tortor nonummy vulputate. Quisque sit amet eros sed purus euismod tempor. Morbi tempor. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Curabitur diam. Suspendisse at purus in ipsum volutpat viverra. Nulla pellentesque libero id libero.";
aTa.setSize(200, 200);
aTa.move(200,100);

addChild(aCp);
addChild(aTa);

function changeHandler(event:ColorPickerEvent):void {
    if(TextFormat(aTa.getStyle("textFormat"))){
        aTf = TextFormat(aTa.getStyle("textFormat"));
    }
    aTf.color = event.target.selectedColor;
    aTa.setStyle("textFormat", aTf);
}
```

- 5 制御/ムービープレビューを選択します。

ComboBox コンポーネントの使用

ComboBox コンポーネントを使用すると、ユーザーがドロップダウンリストから 1 つのアイテムを選択できます。ComboBox には、静的なコンボボックスと編集可能なコンボボックスがあります。編集可能な ComboBox では、リストの一番上にあるテキストフィールドにユーザーがテキストを直接入力できます。ドロップダウンリストがドキュメントの下端にぶつかる場合は、リストが下ではなく上に伸びます。ComboBox は、3 つのサブコンポーネント (BaseButton、TextInput、および List) で構成されます。

編集可能な ComboBox の場合は、ボタンの部分だけがヒット領域になり、テキストボックスはヒット領域に含まれません。静的な ComboBox の場合は、ボタンとテキストボックスの両方がヒット領域になります。ヒット領域をクリックすると、ドロップダウンリストの開閉を切り替えることができます。

ユーザーがマウスまたはキーボードのいずれかを使用してリストからアイテムを選択すると、選択したアイテムのラベルが ComboBox の上部にあるテキストフィールドにコピーされます。

ComboBox コンポーネントのユーザー操作

ComboBox コンポーネントは、フォームやアプリケーション内でユーザーにリストから 1 つのアイテムを選択させたい場合に使用できます。例えば、顧客の住所を入力するフォームで州名をドロップダウンリストから選択できるようにする用途などが考えられます。もっと複雑なシナリオでは、編集可能な ComboBox を使用することもできます。例えば、道順を案内するアプリケーションで、出発地と目的地の住所を入力する欄に編集可能な ComboBox を使用し、以前に入力した住所をドロップダウンリストから選択できる機能を作成できます。

ComboBox が編集可能、つまり `editable` プロパティが `true` の場合は、次のキーを使用するとテキスト入力ボックスからフォーカスが離れ、前の値がそのまま残ります。ユーザーが入力したテキストを、`Enter` キーで新しい値として最初に適用する場合は例外です。

キー	説明
Shift+Tab	フォーカスを前のアイテムに移動します。新しいアイテムが選択されると、 <code>change</code> イベントが送り出されます。
Tab	フォーカスを次のアイテムに移動します。新しいアイテムが選択されると、 <code>change</code> イベントが送り出されます。
下向き矢印	1 つ下のアイテムを選択します。
End	リストの一番下のアイテムを選択します。
Esc	ドロップダウンリストを閉じて、フォーカスを ComboBox に戻します。
Enter	ドロップダウンリストを閉じて、フォーカスを ComboBox に戻します。編集可能な ComboBox にユーザーがテキストを入力した場合は、入力されたテキストが値として設定されます。
Home	リストの一番上のアイテムを選択します。
Page Up	1 ページ上のアイテムを選択します。
Page Down	1 ページ下のアイテムを選択します。

ComboBox コンポーネントをアプリケーションに追加する際に、次の ActionScript コード行を追加することで、そのコンポーネントをスクリーンリーダーが読み取れるように設定できます。

```
import fl.accessibility.ComboBoxAccImpl;  
  
ComboBoxAccImpl.enableAccessibility();
```

1 つのコンポーネントに対してアクセシビリティ機能を有効化する処理は、そのコンポーネントのインスタンスをいくつ使用するかに関係なく、一度だけ実行します。

ComboBox コンポーネントのパラメーター

プロパティインスペクターまたはコンポーネントインスペクターで ComboBox の各インスタンスに設定できるパラメーターは、`dataProvider`、`editable`、`prompt` および `rowCount` です。これらの各パラメーターには、対応する同名の ActionScript プロパティがあります。これらのパラメーターに設定できる値については、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の ComboBox クラスを参照してください。`dataProvider` パラメーターの使用について詳しくは、28 ページの「[dataProvider パラメーターの使用](#)」を参照してください。

ComboBox コンポーネントを使用したアプリケーションの作成

次の手順は、オーサリング時に ComboBox コンポーネントをアプリケーションに追加する方法を示しています。この例では ComboBox が編集可能であり、テキストフィールドに **Add** と入力すると、ドロップダウンリストにアイテムが追加されます。

- 1 新しい Flash (ActionScript 3.0) ドキュメントを作成します。
- 2 ComboBox をステージにドラッグし、インスタンス名を **aCb** にします。「パラメーター」タブで、`editable` パラメーターを `true` に設定します。
- 3 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次のコードを入力します。

```
import fl.data.DataProvider;
import fl.events.ComponentEvent;

var items:Array = [
    {label:"screen1", data:"screenData1"},
    {label:"screen2", data:"screenData2"},
    {label:"screen3", data:"screenData3"},
    {label:"screen4", data:"screenData4"},
    {label:"screen5", data:"screenData5"},
];
aCb.dataProvider = new DataProvider(items);

aCb.addEventListener(ComponentEvent.ENTER, onAddItem);

function onAddItem(event:ComponentEvent):void {
    var newRow:int = 0;
    if (event.target.text == "Add") {
        newRow = event.target.length + 1;
        event.target.addItemAt({label:"screen" + newRow, data:"screenData" + newRow},
            event.target.length);
    }
}
```

- 4 制御/ムービープレビューを選択します。

ActionScript を使用した ComboBox の作成

次の例では、ActionScript を使用して ComboBox を作成し、サンフランシスコおよびカリフォルニアの地域にある大学のリストを設定します。ComboBox の `width` プロパティはプロンプトに表示するテキストの幅に合わせて設定され、`dropdownWidth` プロパティは最も長い大学の名前よりも若干広く設定されます。

この例では、Array インスタンス内に大学のリストを作成します。label プロパティを使用して学校の名前を格納し、data プロパティを使用して各学校の Web サイトの URL を格納します。この Array は、dataProvider プロパティを設定することで ComboBox に割り当てられます。

ユーザーがリストから大学を選択すると、Event.CHANGE イベントおよび `changeHandler()` 関数の呼び出しがトリガーされ、学校の Web サイトにアクセスするために data プロパティが URL 要求にロードされます。

最後の行では、リストを閉じたときにプロンプトを再表示するために、ComboBox インスタンスの `selectedIndex` プロパティを `-1` に設定しています。この設定がないと、プロンプトは選択された学校の名前に置き換えられます。

- 1 新しい Flash (ActionScript 3.0) ドキュメントを作成します。
- 2 ComboBox コンポーネントをコンポーネントパネルからライブラリパネルにドラッグします。
- 3 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次の ActionScript コードを入力します。

```
import fl.controls.ComboBox;
import fl.data.DataProvider;
import flash.net.navigateToURL;

var sfUniversities:Array = new Array(
    {label:"University of California, Berkeley",
     data:"http://www.berkeley.edu/"},
    {label:"University of San Francisco",
     data:"http://www.usfca.edu/"},
    {label:"San Francisco State University",
     data:"http://www.sfsu.edu/"},
    {label:"California State University, East Bay",
     data:"http://www.csuhayward.edu/"},
    {label:"Stanford University", data:"http://www.stanford.edu/"},
    {label:"University of Santa Clara", data:"http://www.scu.edu/"},
    {label:"San Jose State University", data:"http://www.sjsu.edu/" }
);

var aCb:ComboBox = new ComboBox();
aCb.dropdownWidth = 210;
aCb.width = 200;
aCb.move(150, 50);
aCb.prompt = "San Francisco Area Universities";
aCb.dataProvider = new DataProvider(sfUniversities);
aCb.addEventListener(Event.CHANGE, changeHandler);

addChild(aCb);

function changeHandler(event:Event):void {
    var request:URLRequest = new URLRequest();
    request.url = ComboBox(event.target).selectedItem.data;
    navigateToURL(request);
    aCb.selectedIndex = -1;
}
```

4 制御/ムービープレビューを選択します。

この例は Flash オーサリング環境に実装して実行できますが、ComboBox でアイテムをクリックして大学の Web サイトにアクセスしようとする、警告メッセージが表示されます。インターネットから ComboBox の機能に完全にアクセスするには、次の URL を参照してください。

<http://www.helpexamples.com/peter/bayAreaColleges/bayAreaColleges.html>

DataGrid コンポーネントの使用

DataGrid コンポーネントを使用すると、行と列から成るグリッドにデータを表示できます。表示するデータは、配列から、または DataProvider の配列として解析できる外部 XML ファイルから取得できます。DataGrid コンポーネントには、垂直および水平のスクロール、イベントサポート (編集可能なセルのサポートを含む)、および並べ替えの機能が備わっています。

グリッド内の列のサイズを変更したり、フォントやカラー、境界線などの特性をカスタマイズしたりすることができます。グリッド内の列に対して、カスタムムービークリップをセルレンダラーとして使うことができます。セルレンダラーはセルの内容を表示します。スクロールバーを無効にして DataGrid メソッドを使用すると、ページビュースタイルの表示を作成できます。カスタマイズについて詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の DataGridColumn クラスを参照してください。

関連項目

[DataGrid コンポーネントの作成、入力、サイズ変更](#)

[DataGrid コンポーネントのカスタマイズと並べ替え](#)

[DataGrid コンポーネントでのデータのフィルターとフォーマット](#)

DataGrid コンポーネントのユーザー操作

DataGrid コンポーネントは、マウスとキーボードを使って操作することができます。

sortableColumns プロパティと列の sortable プロパティの両方が true の場合は、列ヘッダーをクリックすると、その列の値に従ってデータが並べ替えられます。各列の sortable プロパティを false に設定すると、列ごとに並べ替えを無効にできます。

resizableColumns プロパティが true の場合は、ヘッダー行で列の分割線をドラッグすると、列のサイズを変更できます。

編集可能なセル内をクリックすると、そのセルにフォーカスが移動します。編集不可能なセル内をクリックしても、フォーカスには影響しません。それぞれのセルは、そのセルの DataGrid.editable プロパティと DataGridColumn.editable プロパティの両方が true である場合に編集可能になります。

詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の DataGrid クラスおよび DataGridColumn クラスを参照してください。

クリックするか Tab キーで移動することによって DataGrid インスタンスにフォーカスが移動した場合は、次のキーを使って操作することができます。

キー	説明
下向き矢印	セルの編集中は、挿入ポインターがセルのテキストの最後に移動します。セルが編集不可能な場合は、List コンポーネントと同様の方法で選択操作が処理されます。
上向き矢印	セルの編集中は、挿入ポインターがセルのテキストの最初に移動します。セルが編集不可能な場合は、List コンポーネントと同様の方法で選択操作が処理されます。
Shift+ ↑ / Shift+ ↓	DataGrid が編集不可能で、allowMultipleSelection が true の場合は、連続した複数の行が選択されます。逆向きの矢印キーで反対方向に移動すると、元の方向に複数行の選択を開始した行を過ぎるまで、選択されていた行が選択解除されます。
Shift キーを押しながらクリック	allowMultipleSelection が true の場合は、選択された行と現在のキャレット位置（ハイライトされているセル）の間にある行がすべて選択されます。
Ctrl キーを押しながらクリック	allowMultipleSelection が true の場合は、複数の行を選択できます。これらの行は連続している必要はありません。
右向き矢印	セルが編集中の場合は、挿入ポインターが 1 文字右に移動します。セルが編集不可能な場合は何も起こりません。
左向き矢印	セルが編集中の場合は、挿入ポインターが 1 文字左に移動します。セルが編集不可能な場合は何も起こりません。
Home	DataGrid の先頭の行が選択されます。
End	DataGrid の最後の行が選択されます。
PageUp	DataGrid のページ内の先頭の行が選択されます。ページは、DataGrid がスクロールせずに表示できる行数で構成されます。

キー	説明
PageDown	DataGrid のページ内の最後の行が選択されます。ページは、スクロールせずに DataGrid が表示できる行数で構成されます。
Return/Enter/Shift+Enter	セルが編集可能な場合は、変更内容が確定され、挿入ポインターが次の行の同じ列のセルに移動します。次の行が上か下かは Shift キーの状態によって決まります。
Shift+Tab/Tab	DataGrid が編集可能な場合は、列の終端に達するまで、前のアイテム (次のアイテム) にフォーカスが移動します。列の終端に到達すると、先頭のセル (最後のセル) に達するまで、前の行 (次の行) にフォーカスが移動します。先頭のセルが選択されている場合は、Shift+Tab キーを押すと、前のコントロールにフォーカスが移動します。最後のセルが選択されている場合は、Tab キーを押すと、次のコントロールにフォーカスが移動します。 DataGrid が編集不可能な場合は、前のコントロール (次のコントロール) にフォーカスが移動します。

DataGrid コンポーネントは、データを中心とするさまざまな種類のアプリケーションの基盤として使うことができます。データを書式設定し、表形式で表示することが簡単にできます。また、セルレンダラー機能を使用して、さらに洗練された編集可能なユーザーインターフェイス要素を作成することもできます。次に、DataGrid コンポーネントの使用例を示します。

- Web メールクライアント
- 検索結果ページ
- ローン計算や税金報告書などのスプレッドシートアプリケーション

DataGrid クラスは SelectableList クラスを拡張したもので、DataGrid コンポーネントを使用してアプリケーションをデザインするときは、List コンポーネントのデザインを理解することが大切です。SelectableList クラスおよび List コンポーネントについて詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の SelectableList クラスおよび List クラスを参照してください。

DataGrid コンポーネントをアプリケーションに追加する際に、次の ActionScript コード行を追加することで、そのコンポーネントをスクリーンリーダーが読み取れるように設定できます。

```
import fl.accessibility.DataGridAccImpl;
DataGridAccImpl.enableAccessibility();
```

1 つのコンポーネントに対してアクセシビリティ機能を有効化する処理は、そのコンポーネントがインスタンスをいくつ持つかに関係なく、一度だけ実行します。詳しくは、『Flash ユーザーガイド』の第 18 章「アクセシビリティコンテンツの作成」を参照してください。

DataGrid コンポーネントのパラメーター

プロパティインスペクターまたはコンポーネントインスペクターで DataGrid コンポーネントの各インスタンスに設定できるオーサリングパラメーターは、allowMultipleSelection、editable、headerHeight、horizontalLineScrollSize、horizontalPageScrollSize、horizontalScrollPolicy、resizableColumns、rowHeight、showHeaders、verticalLineScrollSize、verticalPageScrollSize および verticalScrollPolicy です。これらの各パラメーターには、対応する同名の ActionScript プロパティがあります。これらのパラメーターに設定できる値については、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の DataGrid クラスを参照してください。

DataGrid コンポーネントを使用したアプリケーションの作成

DataGrid コンポーネントを使用してアプリケーションを作成するには、まずデータをどこから取り出すかを定める必要があります。通常は、dataProvider プロパティを設定することでグリッドに読み込むことができる Array からデータを取り出します。また、DataGrid クラスと DataGridColumn クラスのメソッドを使用して、データをグリッドに追加することもできます。

DataGrid コンポーネントでのローカルデータプロバイダーの使用

この例では、DataGrid を作成して、ソフトボールチームの名簿を表示します。この例では、名簿を Array (aRoster) に定義し、DataGrid の dataProvider プロパティに割り当てています。

- 1 Flash で、ファイル/新規を選択し、「Flash ファイル (ActionScript 3.0)」を選択します。
- 2 DataGrid コンポーネントをコンポーネントパネルからステージまでドラッグします。
- 3 プロパティインスペクターで、インスタンス名として **aDg** と入力します。
- 4 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次の ActionScript コードを入力します。

```
import fl.data.DataProvider;

bldRosterGrid(aDg);
var aRoster:Array = new Array();
aRoster = [
    {Name:"Wilma Carter", Bats:"R", Throws:"R", Year:"So", Home: "Redlands, CA"},
    {Name:"Sue Pennypacker", Bats:"L", Throws:"R", Year:"Fr", Home: "Athens, GA"},
    {Name:"Jill Smithfield", Bats:"R", Throws:"L", Year:"Sr", Home: "Spokane, WA"},
    {Name:"Shirley Goth", Bats:"R", Throws:"R", Year:"Sr", Home: "Carson, NV"},
    {Name:"Jennifer Dunbar", Bats:"R", Throws:"R", Year:"Fr", Home: "Seaside, CA"},
    {Name:"Patty Crawford", Bats:"L", Throws:"L", Year:"Jr", Home: "Whittier, CA"},
    {Name:"Angelina Davis", Bats:"R", Throws:"R", Year:"So", Home: "Odessa, TX"},
    {Name:"Maria Santiago", Bats:"L", Throws:"L", Year:"Sr", Home: "Tacoma, WA"},
    {Name:"Debbie Ferguson", Bats:"R", Throws:"R", Year:"Jr", Home: "Bend, OR"},
    {Name:"Karen Bronson", Bats:"R", Throws:"R", Year:"Sr", Home: "Billings, MO"},
    {Name:"Sylvia Munson", Bats:"R", Throws:"R", Year:"Jr", Home: "Pasadena, CA"},
    {Name:"Carla Gomez", Bats:"R", Throws:"L", Year:"Sr", Home: "Corona, CA"},
    {Name:"Betty Kay", Bats:"R", Throws:"R", Year:"Fr", Home: "Palo Alto, CA"},
];
aDg.dataProvider = new DataProvider(aRoster);
aDg.rowCount = aDg.length;

function bldRosterGrid(dg:DataGrid){
    dg.setSize(400, 300);
    dg.columns = ["Name", "Bats", "Throws", "Year", "Home"];
    dg.columns[0].width = 120;
    dg.columns[1].width = 50;
    dg.columns[2].width = 50;
    dg.columns[3].width = 40;
    dg.columns[4].width = 120;
    dg.move(50,50);
};
```

bldRosterGrid() 関数は、DataGrid のサイズと共に、列の順序とそのサイズを設定します。

- 5 制御/ムービープレビューを選択します。

アプリケーションでの列の指定と DataGrid コンポーネントの並べ替え

列ヘッダーをクリックすると、その列の値に従って DataGrid の内容を降順に並べ替えることができます。

次の例では、addColumn() メソッドを使用して DataColumn インスタンスを DataGrid に追加します。各列は、選手の名前と得点を表しています。この例では、各列の並べ替えオプションを指定するために、sortOptions プロパティも設定します。Name 列に Array.CASEINSENSITIVE、Score 列に Array.NUMERIC を設定します。長さを行数に、幅を 200 に設定することで、DataGrid のサイズが適切に設定されます。

- 1 Flash で、ファイル/新規を選択し、「Flash ファイル (ActionScript 3.0)」を選択します。
- 2 DataGrid コンポーネントをコンポーネントパネルからステージまでドラッグします。
- 3 プロパティインスペクターで、インスタンス名として **aDg** と入力します。

- 4 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次の ActionScript コードを入力します。

```
import fl.controls.dataGridClasses.DataGridColumn;
import fl.events.DataGridEvent;
import fl.data.DataProvider;
// Create columns to enable sorting of data.
var nameDGC:DataGridColumn = new DataGridColumn("name");
nameDGC.sortOptions = Array.CASEINSENSITIVE;
var scoreDGC:DataGridColumn = new DataGridColumn("score");
scoreDGC.sortOptions = Array.NUMERIC;
aDg.addColumn(nameDGC);
aDg.addColumn(scoreDGC);
var aDP_array:Array = new Array({name:"clark", score:3135}, {name:"Bruce", score:403}, {name:"Peter",
score:25})
aDg.dataProvider = new DataProvider(aDP_array);
aDg.rowCount = aDg.length;
aDg.width = 200;
```

- 5 制御/ムービープレビューを選択します。

ActionScript を使用した DataGrid コンポーネントインスタンスの作成

この例では、ActionScript を使用して DataGrid を作成し、選手の名前と得点の Array を設定します。

- 1 新しい Flash (ActionScript 3.0) ドキュメントを作成します。
- 2 DataGrid コンポーネントをコンポーネントパネルから現在のドキュメントのライブラリパネルにドラッグします。
これにより、ライブラリにコンポーネントが追加されます。ただし、アプリケーションには表示されません。
- 3 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次の ActionScript コードを入力します。

```
import fl.controls.DataGrid;
import fl.data.DataProvider;

var aDg:DataGrid = new DataGrid();
addChild(aDg);
aDg.columns = [ "Name", "Score" ];
aDg.setSize(140, 100);
aDg.move(10, 40);
```

このコードでは、DataGrid インスタンスを作成し、グリッドのサイズと配置を設定しています。

- 4 配列を作成し、データを配列に追加して、配列を DataGrid のデータプロバイダーとして指定します。

```
var aDP_array:Array = new Array();
aDP_array.push({Name:"Clark", Score:3135});
aDP_array.push({Name:"Bruce", Score:403});
aDP_array.push({Name:"Peter", Score:25});
aDg.dataProvider = new DataProvider(aDP_array);
aDg.rowCount = aDg.length;
```

- 5 制御/ムービープレビューを選択します。

XML ファイルを使用した DataGrid のロード

次の例では、DataGridColumn クラスを使用して、DataGrid の列を作成します。XML オブジェクトを DataProvider() コンストラクターの value パラメーターとして渡し、DataGrid に格納します。

- 1 テキストエディターを使用して次のデータで XML ファイルを作成し、FLA を保存するフォルダーに team.xml として保存します。

```
<team>
  <player name="Player A" avg="0.293" />
  <player name="Player B" avg="0.214" />
  <player name="Player C" avg="0.317" />
</team>
```

- 2 新しい Flash (ActionScript 3.0) ドキュメントを作成します。
- 3 コンポーネントパネルで、DataGrid コンポーネントをダブルクリックしてステージに追加します。
- 4 プロパティインスペクターで、インスタンス名として **aDg** と入力します。
- 5 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次の ActionScript コードを入力します。

```
import fl.controls.dataGridClasses.DataGridColumn;
import fl.data.DataProvider;
import flash.net.*;
import flash.events.*;

var request:URLRequest = new URLRequest("team.xml");
var loader:URLLoader = new URLLoader;

loader.load(request);
loader.addEventListener(Event.COMPLETE, loaderCompleteHandler);

function loaderCompleteHandler(event:Event):void {

    var teamXML:XML = new XML(loader.data);

    var nameCol:DataGridColumn = new DataGridColumn("name");
    nameCol.headerText = "Name";
    nameCol.width = 120;
    var avgCol:DataGridColumn = new DataGridColumn("avg");
    avgCol.headerText = "Average";
    avgCol.width = 60;

    var myDP:DataProvider = new DataProvider(teamXML);

    aDg.columns = [nameCol, avgCol];
    aDg.width = 200;
    aDg.dataProvider = myDP;
    aDg.rowCount = aDg.length;
}
```

- 6 制御/ムービープレビューを選択します。

Label コンポーネントの使用

Label コンポーネントは、単一行のテキストを表示し、通常は Web ページ上で他のエレメントやアクティビティの識別情報を示すために使用します。ラベルを HTML でフォーマットするように設定すると、そのテキストでフォーマットタグを利用できます。また、ラベルの行揃えやサイズも制御できます。Label コンポーネントには境界線がなく、フォーカスを受け取らず、イベントもブロードキャストしません。

ライブプレビューでは、オーサリング時にプロパティインスペクターまたはコンポーネントインスペクターで各 Label インスタンスのパラメーターに加えた変更が反映されます。Label には境界線がないので、ライブプレビューを確認するには text パラメーターを設定する必要があります。

Label コンポーネントのユーザー操作

Label コンポーネントは、フォーム内の別のコンポーネントにテキストラベルを作成する場合に使用します（ユーザーの名前を受け取る TextInput フィールドの左側に「名前:」というラベルを作成する場合など）。スタイルを使用して一貫性のある外観と操作性を保つことができるので、プレーンテキストフィールドではなく Label コンポーネントを使用することをお勧めします。

Label コンポーネントを回転する場合は、フォントを埋め込む必要があります。そうしないと、ムービープレビュー時にテキストが表示されません。

Label コンポーネントのパラメーター

プロパティインスペクターまたはコンポーネントインスペクターで Label コンポーネントの各インスタンスに設定できるオーサリングパラメーターは、autoSize、condenseWhite、selectable、text および wordWrap です。これらの各パラメーターには、対応する同名の ActionScript プロパティがあります。これらのパラメーターに設定できる値については、「**Adobe Flash Professional CS5** 用 [ActionScript 3.0 リファレンスガイド](#)」の Label クラスを参照してください。

Label コンポーネントを使用したアプリケーションの作成

次の手順は、オーサリング時に Label コンポーネントをアプリケーションに追加する方法を示しています。この例では、単に「有効期限」というテキストをラベルで表示します。

- 1 新しい Flash (ActionScript 3.0) ドキュメントを作成します。
- 2 Label コンポーネントをコンポーネントパネルからステージにドラッグし、プロパティインスペクターで次の値を設定します。
 - インスタンス名として **aLabel** と入力します。
 - W 値として **80** と入力します。
 - X 値として **100** と入力します。
 - Y 値として **100** と入力します。
 - text パラメーターとして**有効期限**と入力します。
- 3 TextArea コンポーネントをステージにドラッグし、プロパティインスペクターで次の値を設定します。
 - インスタンス名として **aTa** と入力します。
 - H 値として **22** と入力します。
 - X 値として **200** と入力します。
 - Y 値として **100** と入力します。
- 4 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次の ActionScript コードを入力します。

```
var today:Date = new Date();
var expDate:Date = addDays(today, 14);
aTa.text = expDate.toString();

function addDays(date:Date, days:Number):Date {
    return addHours(date, days*24);
}

function addHours(date:Date, hrs:Number):Date {
    return addMinutes(date, hrs*60);
}

function addMinutes(date:Date, mins:Number):Date {
    return addSeconds(date, mins*60);
}

function addSeconds(date:Date, secs:Number):Date {
    var mSecs:Number = secs * 1000;
    var sum:Number = mSecs + date.getTime();
    return new Date(sum);
}
```

- 5 制御/ムービープレビューを選択します。

ActionScript を使用した Label コンポーネントインスタンスの作成

次の例では、ActionScript を使用して Label パラメーターを作成します。Label を使用して ColorPicker コンポーネントの機能を識別し、htmlText プロパティを使用して Label のテキストに書式を適用します。

- 1 新しい Flash (ActionScript 3.0) ドキュメントを作成します。
- 2 Label コンポーネントをコンポーネントパネルから現在のドキュメントのライブラリパネルにドラッグします。
- 3 ColorPicker コンポーネントをコンポーネントパネルから現在のドキュメントのライブラリパネルにドラッグします。
- 4 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次の ActionScript コードを入力します。

```
import fl.controls.Label;
import fl.controls.ColorPicker;

var aLabel:Label = new Label();
var aCp:ColorPicker = new ColorPicker();

addChild(aLabel);
addChild(aCp);

aLabel.htmlText = '<font face="Arial" color="#FF0000" size="14">Fill:</font>';
aLabel.x = 200;
aLabel.y = 150;
aLabel.width = 25;
aLabel.height = 22;

aCp.x = 230;
aCp.y = 150;
```

- 5 制御/ムービープレビューを選択します。

List コンポーネントの使用

List コンポーネントは、スクロール可能な単一選択または複数選択のリストボックスです。リストにはグラフィックや他のコンポーネントを表示することもできます。リスト内に表示するアイテムを追加するには、`label` または `data` パラメーターフィールドをクリックすると表示される値ダイアログボックスを使用します。また、`List.addItem()` メソッドや `List.addItemAt()` メソッドを使用してリストにアイテムを追加することもできます。

List コンポーネントでは、0 から始まるインデックスを使用します。つまり、リストの先頭アイテムはインデックス 0 で表されます。List クラスのメソッドとプロパティを使用してリストアイテムを追加、削除、または置換するときには、リストアイテムのインデックスを指定する必要がある場合があります。

List コンポーネントのユーザー操作

リストは、ユーザーが単一アイテムのみ選択できるようにも、また複数アイテムを選択できるようにも設定できます。例えば、ユーザーが電子商取引 Web サイトで購入したい商品を選択するとします。リストには 30 個のアイテムがあり、ユーザーはそれをスクロールし、クリックして選択できます。

また、より詳しい情報をユーザーに提供するために、独自のムービークリップを行として使用する List を作成することもできます。例えば、電子メールアプリケーションのメールボックスを List コンポーネントで表示する場合、重要度や状態を示すアイコンを各行に含めることができます。

List をクリックしたり、Tab キーを押したことにより List にフォーカスが移ると、次のキーを使用して List を制御できます。

キー	説明
英数字キー	ラベルの先頭文字が <code>Key.getAscii()</code> と一致する次のアイテムにジャンプします。
Ctrl	連続しない複数のアイテムを選択および選択解除するときに使用するトグルキーです。
下向き矢印	1 つ下のアイテムを選択します。
Home	リストの一番上にあるアイテムを選択します。
Page Down	1 ページ下のアイテムを選択します。
Page Up	1 ページ上のアイテムを選択します。
Shift	連続する複数のアイテムを選択するときに使用します。
上向き矢印	1 つ上のアイテムを選択します。

注意：また、スクロールするサイズはピクセル単位であり、行単位ではありません。

注意：Page Up キーと Page Down キーでは、一度に表示できるアイテム数から 1 を引いた数をページのサイズとして使用します。例えば、10 行のドロップダウンリストで Page Down キーを押すと、0 ~ 9、9 ~ 18、18 ~ 27 番目のアイテムが順に表示されます。必ず 1 つのアイテムが前後のページと重複することになります。

フォーカスの制御について詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の IFocusManager インターフェイスと FocusManager クラス、および 26 ページの「[FocusManager の操作](#)」を参照してください。

ライブプレビューでは、オーサリング時にプロパティインスペクターまたはコンポーネントインスペクターでパラメーターに加えた変更が、ステージ上の各 List インスタンスに反映されます。

List コンポーネントをアプリケーションに追加する際に、次の ActionScript コード行を追加することで、そのコンポーネントをスクリーンリーダーが読み取れるように設定できます。

```
import fl.accessibility.ListAccImpl;  
  
ListAccImpl.enableAccessibility();
```

1つのコンポーネントに対してアクセシビリティ機能を有効化する処理は、そのコンポーネントがインスタンスをいくつ持つかに関係なく、一度だけ実行します。詳しくは、『Flash ユーザーガイド』の第18章「アクセシビリティコンテンツの作成」を参照してください。

List コンポーネントのパラメーター

プロパティインスペクターまたはコンポーネントインスペクターで List コンポーネントの各インスタンスに設定できるパラメーターは、allowMultipleSelection、dataProvider、horizontalLineScrollSize、horizontalPageScrollSize、horizontalScrollPolicy、multipleSelection、verticalLineScrollSize、verticalPageScrollSize および verticalScrollPolicy です。これらの各パラメーターには、対応する同名の ActionScript プロパティがあります。これらのパラメーターに設定できる値については、「**Adobe Flash Professional CS5** 用 **ActionScript 3.0** リファレンスガイド」の List クラスを参照してください。dataProvider パラメーターの使用について詳しくは、28 ページの「**dataProvider** パラメーターの使用」を参照してください。

List コンポーネントを使用したアプリケーションの作成

次の例は、オーサリング時に List コンポーネントをアプリケーションに追加する方法を示しています。

単純な List コンポーネントのアプリケーションへの追加

この例では、自動車の車種が記載されたラベルと価格が入力されたデータフィールドでリストを構成します。

- 1 新しい Flash (ActionScript 3.0) ドキュメントを作成します。
- 2 List コンポーネントをコンポーネントパネルからステージまでドラッグします。
- 3 プロパティインスペクター内で次のように設定します。
 - インスタンス名として **aList** と入力します。
 - 値 **200** を W (幅) に割り当てます。
- 4 テキストツールを使用して、aList の下にテキストフィールドを作成し、インスタンス名を **aTf** にします。
- 5 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次の ActionScript コードを入力します。

```
import fl.controls.List;  
import flash.text.TextField;  
  
aTf.type = TextFieldType.DYNAMIC;  
aTf.border = false;  
  
// Create these items in the Property inspector when data and label  
// parameters are available.  
aList.addItem({label:"1956 Chevy (Cherry Red)", data:35000});  
aList.addItem({label:"1966 Mustang (Classic)", data:27000});  
aList.addItem({label:"1976 Volvo (Xcllnt Cond)", data:17000});  
aList.allowMultipleSelection = true;  
  
aList.addEventListener(Event.CHANGE, showData);  
  
function showData(event:Event) {  
    aTf.text = "This car is priced at: $" + event.target.selectedItem.data;  
}
```

このコードでは、addItem() メソッドを使用して aList に 3 つのアイテムを設定しています。各アイテムには、リストに表示される label 値と data 値が割り当てられます。リスト内のアイテムを選択すると、イベントリスナによって showData() 関数が呼び出され、これにより選択したアイテムの data 値が表示されます。

6 制御/ムービープレビューを選択し、このアプリケーションをコンパイルして実行します。

データプロバイダーを使用した List インスタンスの設定

この例では、自動車の車種と価格の List を作成します。ここではデータプロバイダーを使用して List を設定し、addItem() メソッドは使用しません。

- 1 新しい Flash (ActionScript 3.0) ドキュメントを作成します。
- 2 List コンポーネントをコンポーネントパネルからステージまでドラッグします。
- 3 プロパティインスペクター内で次のように設定します。
 - インスタンス名として **aList** と入力します。
 - 値 **200** を W (幅) に割り当てます。
- 4 テキストツールを使用して、aList の下にテキストフィールドを作成し、インスタンス名を **aTf** にします。
- 5 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次の ActionScript コードを入力します。

```
import fl.controls.List;
import fl.data.DataProvider;
import flash.text.TextField;

aTf.type = TextFieldType.DYNAMIC;
aTf.border = false;

var cars:Array = [
    {label:"1956 Chevy (Cherry Red)", data:35000},
    {label:"1966 Mustang (Classic)", data:27000},
    {label:"1976 Volvo (Xcllnt Cond)", data:17000},
];
aList.dataProvider = new DataProvider(cars);
aList.allowMultipleSelection = true;

aList.addEventListener(Event.CHANGE, showData);

function showData(event:Event) {
    aTf.text = "This car is priced at: $" + event.target.selectedItem.data;
}
```

6 制御/ムービープレビューを選択して、List にアイテムが追加されたことを確認します。

List コンポーネントを使用した MovieClip インスタンスの制御

次の例では、カラー名の List を作成します。いずれかのカラー名を選択すると、MovieClip にそのカラーが適用されます。

- 1 Flash (ActionScript 3.0) ドキュメントを作成します。
- 2 List コンポーネントをコンポーネントパネルからステージにドラッグし、プロパティインスペクターで次の値を設定します。
 - インスタンス名として **aList** と入力します。
 - H 値として **60** と入力します。
 - X 値として **100** と入力します。
 - Y 値として **150** と入力します。
- 3 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次の ActionScript コードを入力します。

```
aList.addItem({label:"Blue", data:0x0000CC});
aList.addItem({label:"Green", data:0x00CC00});
aList.addItem({label:"Yellow", data:0xFFFF00});
aList.addItem({label:"Orange", data:0xFF6600});
aList.addItem({label:"Black", data:0x000000});

var aBox:MovieClip = new MovieClip();
addChild(aBox);

aList.addEventListener(Event.CHANGE, changeHandler);
function changeHandler(event:Event) {
    drawBox(aBox, event.target.selectedItem.data);
};

function drawBox(box:MovieClip,color:uint):void {
    box.graphics.beginFill(color, 1.0);
    box.graphics.drawRect(225, 150, 100, 100);
    box.graphics.endFill();
}
```

- 4 制御/ムービープレビューを選択して、アプリケーションを実行します。
- 5 List 内のカラーをクリックし、そのカラーが MovieClip に表示されることを確認します。

ActionScript を使用した List コンポーネントインスタンスの作成

この例では、ActionScript を使用して単純な List を作成し、addItem() メソッドを使用して入力します。

- 1 新しい Flash (ActionScript 3.0) ドキュメントを作成します。
- 2 List コンポーネントをコンポーネントパネルからライブラリパネルにドラッグします。
- 3 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次の ActionScript コードを入力します。

```
import fl.controls.List;

var aList:List = new List();
aList.addItem({label:"One", data:1});
aList.addItem({label:"Two", data:2});
aList.addItem({label:"Three", data:3});
aList.addItem({label:"Four", data:4});
aList.addItem({label:"Five", data:5});
aList.setSize(60, 40);
aList.move(200,200);
addChild(aList);
aList.addEventListener(Event.CHANGE, changeHandler);
function changeHandler(event:Event):void {
    trace(event.target.selectedItem.data);
}
```

- 4 制御/ムービープレビューを選択して、アプリケーションを実行します。

NumericStepper コンポーネントの使用

NumericStepper コンポーネントを使用すると、ユーザーが一定の間隔で増減する数値を選択できます。このコンポーネントは、小さな上下の矢印ボタンの隣に表示されるテキストボックス内の数値で構成されています。ユーザーがボタンを押すと、そのボタンを離すまで、または数値が最大値または最小値に到達するまで、stepSize パラメーターで単位として指定した値ずつ数値が増減します。NumericStepper コンポーネントのテキストボックスに表示されるテキストは、編集することもできます。

各 NumericStepper インスタンスのライブプレビューでは、プロパティインスペクターまたはコンポーネントインスペクターで指定した value パラメーターの設定が反映されます。ただし、ライブプレビューでは NumericStepper の矢印ボタンをマウスやキーボードで操作することはできません。

NumericStepper コンポーネントのユーザー操作

NumericStepper コンポーネントは、ユーザーに数値を選択させる場合に使用します。例えば、フォームでクレジットカードの有効期限の年月日を設定する場合などに NumericStepper コンポーネントを使用できます。ユーザーがフォントサイズを増減できるようにする場合にも、NumericStepper コンポーネントを使用できます。

NumericStepper コンポーネントは数値データのみを扱います。また、2 桁より大きな数値 (例えば 5246 や 1.34 など) を表示するには、オーサリング時にサイズを変更する必要があります。

NumericStepper は、アプリケーション内で有効または無効にすることができます。無効状態のときは、マウスやキーボードからの入力を受け付けません。有効状態のときは、ユーザーがクリックするか Tab キーで移動すると、フォーカスを取得します。このとき、内部フォーカスはテキストボックスに設定されます。フォーカスを取得している NumericStepper コンポーネントのインスタンスでは、次のキーを使用した操作が可能です。

キー	説明
下向き矢印	値を 1 単位ずつ変化させます。
左向き矢印	挿入ポインタをテキストボックス内で左に移動します。
右向き矢印	挿入ポインタをテキストボックス内で右に移動します。
Shift+Tab	フォーカスを前のオブジェクトに移動します。
Tab	フォーカスを次のオブジェクトに移動します。
上向き矢印	値を 1 単位ずつ変化させます。

フォーカスの制御について詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の FocusManager クラスおよび 26 ページの「[FocusManager の操作](#)」を参照してください。

NumericStepper コンポーネントのパラメーター

プロパティインスペクターまたはコンポーネントインスペクターで NumericStepper の各インスタンスに設定できるパラメーターは、maximum、minimum、stepSize および value です。これらの各パラメーターには、対応する同名の ActionScript プロパティがあります。これらのパラメーターに設定できる値については、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の NumericStepper クラスを参照してください。

NumericStepper を使用したアプリケーションの作成

次の手順は、オーサリング時に NumericStepper コンポーネントをアプリケーションに追加する方法を示しています。次の例では、NumericStepper コンポーネントと Label コンポーネントをステージ上に配置し、NumericStepper インスタンスの Event.CHANGE イベントのリスナーを作成します。数値ステッパの値が変わると、新しい値を Label インスタンスの text プロパティに表示します。

- 1 NumericStepper コンポーネントをコンポーネントパネルからステージまでドラッグします。
- 2 プロパティインスペクターで、インスタンス名として **aNs** と入力します。
- 3 Label コンポーネントをコンポーネントパネルからステージまでドラッグします。
- 4 プロパティインスペクターで、インスタンス名として **aLabel** と入力します。
- 5 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次の ActionScript コードを入力します。

```
import flash.events.Event;

aLabel.text = "value = " + aNs.value;

aNs.addEventListener(Event.CHANGE, changeHandler);
function changeHandler(event:Event) :void {
    aLabel.text = "value = " + event.target.value;
};
```

この例では、ラベルの text プロパティを NumericStepper の値に設定します。NumericStepper インスタンスの値が変わるたびに、changeHandler() 関数によってラベルの text プロパティが更新されます。

- 6 制御/ムービープレビューを選択します。

ActionScript を使用した NumericStepper の作成

この例では、ActionScript コードを使用して 3 つの NumericSteppers を作成します。それぞれ、ユーザーの誕生日の年、月および日を入力するためのものです。また、各 NumericSteppers のプロンプトや識別子として使用する Label も追加します。

- 1 新しい Flash (ActionScript 3.0) ドキュメントを作成します。
- 2 Label をライブラリパネルにドラッグします。
- 3 NumericStepper コンポーネントをライブラリパネルにドラッグします。
- 4 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次の ActionScript コードを入力します。

```
import fl.controls.Label;
import fl.controls.NumericStepper;

var dobPrompt:Label = new Label();
var moPrompt:Label = new Label();
var dayPrompt:Label = new Label();
var yrPrompt:Label = new Label();

var moNs:NumericStepper = new NumericStepper();
var dayNs:NumericStepper = new NumericStepper();
var yrNs:NumericStepper = new NumericStepper();

addChild(dobPrompt);
addChild(moPrompt);
addChild(dayPrompt);
addChild(yrPrompt);
addChild(moNs);
addChild(dayNs);
addChild(yrNs);

dobPrompt.setSize(65, 22);
dobPrompt.text = "Date of birth:";
dobPrompt.move(80, 150);

moNs.move(150, 150);
moNs.setSize(40, 22);
moNs.minimum = 1;
moNs.maximum = 12;
moNs.stepSize = 1;
moNs.value = 1;

moPrompt.setSize(25, 22);
moPrompt.text = "Mo.";
moPrompt.move(195, 150);

dayNs.move(225, 150);
```

```
dayNs.setSize(40, 22);
dayNs.minimum = 1;
dayNs.maximum = 31;
dayNs.stepSize = 1;
dayNs.value = 1;

dayPrompt.setSize(25, 22);
dayPrompt.text = "Day";
dayPrompt.move(270, 150);

yrNs.move(300, 150);
yrNs.setSize(55, 22);
yrNs.minimum = 1900;
yrNs.maximum = 2006;
yrNs.stepSize = 1;
yrNs.value = 1980;

yrPrompt.setSize(30, 22);
yrPrompt.text = "Year";
yrPrompt.move(360, 150);
```

- 5 制御/ムービープレビューを選択して、アプリケーションを実行します。

ProgressBar コンポーネントの使用

ProgressBar コンポーネントは、コンテンツのロードの進行状況を表示します。これにより、コンテンツが大きく、アプリケーションの実行までに時間がかかる場合でも、ユーザーを安心させることができます。ProgressBar は、イメージやアプリケーションのロードの進行状況を表示させるときに役立ちます。ロードプロセスは有限または無限にすることができます。有限のプログレスバーは、処理の進行状況を時間経過に沿って直線的に表現するもので、ロードするコンテンツの量が事前にわかっているときに使用します。無限のプログレスバーは、ロードするコンテンツの量が不明なときに使用します。Label コンポーネントを追加して、ロードの進行状況をパーセンテージとして表示することもできます。

ProgressBar コンポーネントでは、9 スライスの拡大 / 縮小を使用しており、バーのスキン、トラックのスキン、および無限のバーのスキンが使用されます。

ProgressBar コンポーネントのユーザー操作

ProgressBar コンポーネントには、使用するモードが 3 種類あります。最もよく使用されるのは、イベントモードおよびポーリングモードです。これらのモードは、ロードプロセスで progress イベントと complete イベントを実行するのか（イベントモードおよびポーリングモード）、または bytesLoaded プロパティと bytesTotal プロパティを公開するのか（ポーリングモード）を指定します。また、ProgressBar コンポーネントをマニュアルモードで使用することもできます。その場合は、maximum、minimum および value プロパティを設定し、ProgressBar.setProgress() メソッドを呼び出します。indeterminate プロパティを設定して、ProgressBar にサイズ不明のソースをストライプ模様のバーで表示するのか（true）、またはサイズわかっているソースを塗りつぶして表示するのか（false）を指定できます。

ProgressBar のモードは、mode プロパティを設定して指定します。このプロパティは、プロパティインスペクターまたはコンポーネントインスペクターの mode パラメーターを設定するか、ActionScript を使用して設定します。

100,000 個のアイテムを解析するような処理の状況を ProgressBar を使用して表示する場合、その処理が単一フレーム内でループするときは、画面が再描画されないで、ProgressBar の表示内容は更新されません。

ProgressBar コンポーネントのパラメーター

プロパティインスペクターまたはコンポーネントインスペクターで ProgressBar の各インスタンスに設定できるパラメーターは、direction、mode および source です。これらの各パラメーターには、対応する同名の ActionScript プロパティがあります。

ActionScript を記述すれば、ProgressBar コンポーネントのプロパティ、メソッド、イベントを使用して、これらのオプションやその他のオプションを制御できます。詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の ProgressBar クラスを参照してください。

ProgressBar を使用したアプリケーションの作成

次の手順は、オーサリング時に ProgressBar コンポーネントをアプリケーションに追加する方法を示しています。この例では、ProgressBar でイベントモードを使用します。イベントモードでは、コンテンツのロードで progress イベントと complete イベントが実行されます。このイベントを ProgressBar が送り出して進行状況を示します。progress イベントが発生すると、ラベルが更新されて、ロードされたコンテンツのパーセンテージが表示されます。complete イベントが発生すると、「ロード完了」というメッセージと、ファイルのサイズを示す bytesTotal プロパティの値が表示されます。

- 1 新しい Flash (ActionScript 3.0) ドキュメントを作成します。
- 2 ProgressBar コンポーネントをコンポーネントパネルからステージにドラッグします。
 - プロパティインスペクターで、インスタンス名として **aPb** と入力します。
 - 「パラメーター」セクションに X 値として **200** と入力します。
 - Y 値として **260** と入力します。
 - mode パラメーターで event を選択します。
- 3 Button コンポーネントをコンポーネントパネルからステージまでドラッグします。
 - プロパティインスペクターで、インスタンス名として **loadButton** と入力します。
 - X パラメーターとして **220** と入力します。
 - Y パラメーターとして **290** と入力します。
 - label パラメーターとして **サウンドをロード** と入力します。
- 4 Label コンポーネントをステージにドラッグし、インスタンス名を **progLabel** にします。
 - W 値として **150** と入力します。
 - X パラメーターとして **200** と入力します。
 - Y パラメーターとして **230** と入力します。
 - 「パラメーター」セクションで、text パラメーターの値をクリアします。
- 5 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次の ActionScript コードを入力します。このコードで、.mp3 オーディオファイルがロードされます。

```
import fl.controls.ProgressBar;
import flash.events.ProgressEvent;
import flash.events.IOErrorEvent;

var aSound:Sound = new Sound();
aPb.source = aSound;
var url:String = "http://www.helpexamples.com/flash/sound/song1.mp3";
var request:URLRequest = new URLRequest(url);

aPb.addEventListener(ProgressEvent.PROGRESS, progressHandler);
aPb.addEventListener(Event.COMPLETE, completeHandler);
aSound.addEventListener(IOErrorEvent.IO_ERROR, ioErrorHandler);
loadButton.addEventListener(MouseEvent.CLICK, clickHandler);

function progressHandler(event:ProgressEvent):void {
    progLabel.text = ("Sound loading ... " + aPb.percentComplete);
}

function completeHandler(event:Event):void {
    trace("Loading complete");
    trace("Size of file: " + aSound.bytesTotal);
    aSound.close();
    loadButton.enabled = false;
}

function clickHandler(event:MouseEvent) {
    aSound.load(request);
}

function ioErrorHandler(event:IOErrorEvent):void {
    trace("Load failed due to: " + event.text);
}
```

6 制御/ムービープレビューを選択します。

ポーリングモードの ProgressBar コンポーネントを使用したアプリケーションの作成

次の例では、ProgressBar をポーリングモードに設定します。ポーリングモードでは、ロードされているコンテンツで発生する progress イベントをリスンし、コンテンツの bytesLoaded プロパティと bytesTotal プロパティを使用して計算することにより進行状況が決定されます。この例では、Sound オブジェクトをロードし、その progress イベントをリスンし、bytesLoaded プロパティおよび bytesTotal プロパティを使用してロードされたパーセンテージを計算します。ラベルと出力パネルの両方に、ロードされたパーセンテージが表示されます。

- 1 新しい Flash (ActionScript 3.0) ドキュメントを作成します。
- 2 ProgressBar コンポーネントをコンポーネントパネルからステージにドラッグし、プロパティインスペクターで次の値を入力します。
 - インスタンス名として **aPb** と入力します。
 - X 値として **185** と入力します。
 - Y 値として **225** と入力します。
- 3 Label コンポーネントをステージにドラッグし、プロパティインスペクターで次の値を入力します。
 - インスタンス名として **progLabel** と入力します。
 - X 値として **180** と入力します。
 - Y 値として **180** と入力します。
 - 「パラメーター」セクションで、text パラメーターの値をクリアします。

- 4 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次の ActionScript コードを入力します。このコードで Sound オブジェクト (aSound) を作成し、loadSound() を呼び出して Sound オブジェクトにサウンドをロードします。

```
import fl.controls.ProgressBarMode;
import flash.events.ProgressEvent;
import flash.media.Sound;

var aSound:Sound = new Sound();
var url:String = "http://www.helpexamples.com/flash/sound/song1.mp3";
var request:URLRequest = new URLRequest(url);

aPb.mode = ProgressBarMode.POLLED;
aPb.source = aSound;
aSound.addEventListener(ProgressEvent.PROGRESS, loadListener);

aSound.load(request);

function loadListener(event:ProgressEvent) {
    var percentLoaded:int = event.target.bytesLoaded / event.target.bytesTotal * 100;
    progLabel.text = "Percent loaded: " + percentLoaded + "%";
    trace("Percent loaded: " + percentLoaded + "%");
}
```

- 5 制御/ムービープレビューを選択して、アプリケーションを実行します。

マニュアルモードの ProgressBar コンポーネントを使用したアプリケーションの作成

次の例では、ProgressBar をマニュアルモードに設定します。マニュアルモードでは、現在の値と最大値を指定して setProgress() メソッドを呼び出し、進捗状況を決定して、進行状況を手動で設定する必要があります。マニュアルモードでは source プロパティを設定しません。この例では、NumericStepper コンポーネントを使用し、ProgressBar を最大値 250 までインクリメントできるようにしています。NumericStepper の値が変更されて CHANGE イベントがトリガーされると、イベントハンドラー (nsChangeHandler) で setProgress() メソッドが呼び出されて ProgressBar を進行します。また、指定された最大値を基準として、進行が完了したパーセンテージも表示されます。

- 1 新しい Flash (ActionScript 3.0) ドキュメントを作成します。
- 2 ProgressBar コンポーネントをコンポーネントパネルからステージにドラッグし、プロパティインスペクターで次の値を設定します。
 - インスタンス名として **aPb** と入力します。
 - X 値として **180** と入力します。
 - Y 値として **175** と入力します。
- 3 NumericStepper コンポーネントをステージにドラッグし、プロパティインスペクターで次の値を入力します。
 - インスタンス名として **aNs** と入力します。
 - X 値として **220** と入力します。
 - Y 値として **215** と入力します。
 - 「パラメーター」セクションで、maximum パラメーターとして **250**、minimum 値として **0**、stepSize パラメーターとして **1**、value パラメーターとして **0** をそれぞれ入力します。
- 4 Label コンポーネントをステージにドラッグし、プロパティインスペクターで次の値を入力します。
 - インスタンス名として **progLabel** と入力します。
 - W 値として **150** と入力します。
 - X 値として **180** と入力します。

- Y 値として **120** と入力します。
- 「パラメーター」タブで、**text** パラメーターの **Label** の値をクリアします。

5 アクションパネルを開いて、メインタイムライン内でフレーム **1** を選択し、次のコードを入力します。

```
import fl.controls.ProgressBarDirection;
import fl.controls.ProgressBarMode;
import flash.events.Event;

aPb.direction = ProgressBarDirection.RIGHT;
aPb.mode = ProgressBarMode.MANUAL;
aPb.minimum = aNs.minimum;
aPb.maximum = aNs.maximum;
aPb.indeterminate = false;

aNs.addEventListener(Event.CHANGE, nsChangeHandler);

function nsChangeHandler(event:Event):void {
    aPb.value = aNs.value;
    aPb.setProgress(aPb.value, aPb.maximum);
    progLabel.text = "Percent of progress = " + int(aPb.percentComplete) + "%";
}
```

- 6** 制御/ムービープレビューを選択して、アプリケーションを実行します。
- 7** **NumericStepper** で上向き矢印を押して、**ProgressBar** を進行します。

ActionScript を使用した **ProgressBar** の作成

この例では、ActionScript を使用して **ProgressBar** を作成しています。これは、マニュアルモードで **ProgressBar** を作成する前の例の機能と重複しています。

- 1** 新しい Flash (ActionScript 3.0) ドキュメントを作成します。
- 2** **ProgressBar** コンポーネントをライブラリパネルにドラッグします。
- 3** **NumericStepper** コンポーネントをライブラリパネルにドラッグします。
- 4** **Label** コンポーネントをライブラリパネルにドラッグします。
- 5** アクションパネルを開いて、メインタイムライン内でフレーム **1** を選択し、次のコードを入力します。

```
import fl.controls.ProgressBar;
import fl.controls.NumericStepper;
import fl.controls.Label;
import fl.controls.ProgressBarDirection;
import fl.controls.ProgressBarMode;
import flash.events.Event;

var aPb:ProgressBar = new ProgressBar();
var aNs:NumericStepper = new NumericStepper();
var progLabel:Label = new Label();

addChild(aPb);
addChild(aNs);
addChild(progLabel);

aPb.move(180,175);
aPb.direction = ProgressBarDirection.RIGHT;
aPb.mode = ProgressBarMode.MANUAL;

progLabel.setSize(150, 22);
progLabel.move(180, 150);
progLabel.text = "";

aNs.move(220, 215);
aNs.maximum = 250;
aNs.minimum = 0;
aNs.stepSize = 1;
aNs.value = 0;

aNs.addEventListener(Event.CHANGE, nsChangeHandler);

function nsChangeHandler(event:Event):void {
    aPb.setProgress(aNs.value, aNs.maximum);
    progLabel.text = "Percent of progress = " + int(aPb.percentComplete) + "%";
}
```

- 6 制御/ムービープレビューを選択して、アプリケーションを実行します。
- 7 NumericStepper で上向き矢印を押して、ProgressBar を進行します。

RadioButton コンポーネントの使用

RadioButton コンポーネントを使用すると、複数の選択肢の中から 1 つだけをユーザーに選択させることができます。このコンポーネントを使用する場合は、少なくとも 2 つの RadioButton インスタンスが含まれるグループを構成して使用します。グループ内のメンバーは一度に 1 つしか選択できません。1 つのラジオボタンを選択すると、同じグループ内でそれまで選択されていたラジオボタンは選択解除されます。各ラジオボタンが所属するグループは、groupName パラメーターによって設定します。

ラジオボタンは、Web 上の多くのフォームアプリケーションに使用される基本的な要素です。いくつかの選択肢から 1 つをユーザーに選択させる場合に、ラジオボタンを使用します。例えば、フォームでユーザーが使用するクレジットカードを選択させる場合などはラジオボタンを使用します。

RadioButton コンポーネントのユーザー操作

ラジオボタンは有効または無効にすることができます。無効なラジオボタンは、マウスおよびキーボードからの入力を受け付けません。ユーザーが RadioButton コンポーネントグループをクリックするか、Tab キーで移動すると、選択されているラジオボタンだけがフォーカスを取得します。これにより、ユーザーは次のキーを使用した操作が可能になります。

キー	説明
上向き矢印、左向き矢印	そのラジオボタングループ内で、選択を前のラジオボタンに移動します。
下向き矢印、右向き矢印	そのラジオボタングループ内で、選択を次のラジオボタンに移動します。
Tab	フォーカスを現在のラジオボタングループから次のコンポーネントに移動します。

フォーカスの制御について詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の IFocusManager インターフェイスと FocusManager クラス、および 26 ページの「[FocusManager の操作](#)」を参照してください。

ライブプレビューでは、オーサリング時にプロパティインスペクターまたはコンポーネントインスペクターでパラメーターに加えた変更が、ステージ上の各 RadioButton インスタンスに反映されます。ただし、一度に 1 つしか選択できないという動作はライブプレビューでは再現されません。同じグループ内にある 2 つのラジオボタンについて selected パラメーターを true に設定した場合、両方が選択状態になったように見えますが、実行時には後に作成したインスタンスだけが選択状態になります。詳しくは、73 ページの「[RadioButton コンポーネントのパラメーター](#)」を参照してください。

RadioButton コンポーネントをアプリケーションに追加する際に、次のコード行を追加することで、そのコンポーネントをスクリーンリーダーが読み取れるように設定できます。

```
import fl.accessibility.RadioButtonAccImpl;
RadioButtonAccImpl.enableAccessibility();
```

1 つのコンポーネントに対してアクセシビリティ機能を有効化する処理は、そのコンポーネントのインスタンスをいくつ使用するかに関係なく、一度だけ実行します。詳しくは、『Flash ユーザーガイド』の第 18 章「アクセシビリティコンテンツの作成」を参照してください。

RadioButton コンポーネントのパラメーター

プロパティインスペクターまたはコンポーネントインスペクターで RadioButton コンポーネントの各インスタンスに設定できるオーサリングパラメーターは、groupName、label、LabelPlacement、selected および value です。これらの各パラメーターには、対応する同名の ActionScript プロパティがあります。これらのパラメーターに設定できる値については、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の RadioButton クラスを参照してください。

ActionScript を記述すれば、RadioButton クラスのメソッド、プロパティ、イベントを使用して、RadioButton インスタンスのその他のオプションを設定できます。

RadioButton コンポーネントを使用したアプリケーションの作成

次の手順は、オーサリング時に RadioButton コンポーネントをアプリケーションに追加する方法を示しています。この例では、「はい」または「いいえ」で回答する質問を表示するために RadioButton を使用します。RadioButton のデータは TextArea に表示されます。

- 1 新しい Flash (ActionScript 3.0) ドキュメントを作成します。
- 2 2 つの RadioButton コンポーネントをコンポーネントパネルからステージまでドラッグします。
- 3 1 番目のラジオボタンを選択します。プロパティインスペクターで、インスタンス名を **yesRb**、グループ名を **rbGroup** にします。
- 4 2 番目のラジオボタンを選択します。プロパティインスペクターで、インスタンス名を **noRb**、グループ名を **rbGroup** にします。
- 5 TextArea コンポーネントをコンポーネントパネルからステージまでドラッグし、インスタンス名を **aTa** にします。
- 6 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次の ActionScript コードを入力します。

```
yesRb.label = "Yes";
yesRb.value = "For";
noRb.label = "No";
noRb.value = "Against";

yesRb.move(50, 100);
noRb.move(100, 100);
aTa.move(50, 30);
noRb.addEventListener(MouseEvent.CLICK, clickHandler);
yesRb.addEventListener(MouseEvent.CLICK, clickHandler);

function clickHandler(event:MouseEvent):void {
    aTa.text = event.target.value;
}
```

- 7 制御/ムービープレビューを選択して、アプリケーションを実行します。

ActionScript を使用した RadioButton の作成

この例では、ActionScript を使用して、赤、青、緑の各カラーの 3 つの RadioButton を作成し、灰色のボックスを描画します。各 RadioButton の value プロパティは、ボタンに関連付けられたカラーの 16 進数値を指定します。ユーザーがいずれかの RadioButton をクリックすると、clickHandler() 関数で drawBox() が呼び出され、RadioButton の value プロパティからカラーが渡されて、ボックスにカラーが適用されます。

- 1 新しい Flash (ActionScript 3.0) ドキュメントを作成します。
- 2 RadioButton コンポーネントをライブラリパネルにドラッグします。
- 3 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次の ActionScript コードを入力します。

```
import fl.controls.RadioButton;
import fl.controls.RadioButtonGroup;

var redRb:RadioButton = new RadioButton();
var blueRb:RadioButton = new RadioButton();
var greenRb:RadioButton = new RadioButton();
var rbGrp:RadioButtonGroup = new RadioButtonGroup("colorGrp");

var aBox:MovieClip = new MovieClip();
drawBox(aBox, 0xCCCCCC);

addChild(redRb);
addChild(blueRb);
addChild(greenRb);
addChild(aBox);

redRb.label = "Red";
redRb.value = 0xFF0000;
blueRb.label = "Blue";
blueRb.value = 0x0000FF;
greenRb.label = "Green";
greenRb.value = 0x00FF00;
redRb.group = blueRb.group = greenRb.group = rbGrp;
redRb.move(100, 260);
blueRb.move(150, 260);
greenRb.move(200, 260);

rbGrp.addEventListener(MouseEvent.CLICK, clickHandler);

function clickHandler(event:MouseEvent):void {
    drawBox(aBox, event.target.selection.value);
}

function drawBox(box:MovieClip,color:uint):void {
    box.graphics.beginFill(color, 1.0);
    box.graphics.drawRect(125, 150, 100, 100);
    box.graphics.endFill();
}
```

- 4 制御/ムービープレビューを選択して、アプリケーションを実行します。

詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の RadioButton クラスを参照してください。

ScrollPane コンポーネントの使用

ロードされた領域に収まらない大きなコンテンツを表示する場合、ScrollPane コンポーネントを使用します。例えば、大きなイメージがあり、アプリケーションに小さなスペースしかない場合でも、ScrollPane にロードすれば表示できます。ScrollPane には、ムービークリップ、JPEG、PNG、GIF、および SWF ファイルをロードできます。

ScrollPane や UILoader などのコンポーネントには、コンテンツのロードがいつ完了したかを判別できる complete イベントがあります。ScrollPane コンポーネントや UILoader コンポーネントのコンテンツにプロパティを設定する場合は、complete イベントをリスンし、イベントハンドラーでプロパティを設定します。例えば、次のコードでは、Event.COMPLETE イベントのリスナと、ScrollPane のコンテンツの alpha プロパティを 0.5 に設定するイベントハンドラーを作成します。

```
function spComplete(event:Event):void{
    aSp.content.alpha = .5;
}
aSp.addEventListener(Event.COMPLETE, spComplete);
```

ScrollPane にコンテンツをロードするときに位置を指定する場合は、その位置 (X 座標および Y 座標) を 0, 0 と指定する必要があります。例えば、次のコードでは、ボックスが 0, 0 の位置に描画されるため、ScrollPane が正しくロードされます。

```
var box:MovieClip = new MovieClip();
box.graphics.beginFill(0xFF0000, 1);
box.graphics.drawRect(0, 0, 150, 300);
box.graphics.endFill();
aSp.source = box;//load ScrollPane
```

詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の ScrollPane クラスを参照してください。

ScrollPane コンポーネントのユーザー操作

ScrollPane は有効または無効にすることができます。無効な ScrollPane は、マウスおよびキーボードからの入力を受け付けません。フォーカスを取得している ScrollPane では、次のキーを使用した操作が可能です。

キー	説明
下向き矢印	コンテンツを垂直方向に 1 行上にスクロールします。
上向き矢印	コンテンツを垂直方向に 1 行下にスクロールします。
End	コンテンツを ScrollPane の一番下に移動します。
左向き矢印	コンテンツを水平方向に 1 行右にスクロールします。
右向き矢印	コンテンツを水平方向に 1 行左にスクロールします。
Home	コンテンツを ScrollPane の最上部に移動します。
End	コンテンツを ScrollPane の一番下に移動します。
Page Down	コンテンツを垂直方向に 1 ページ上にスクロールします。
Page Up	コンテンツを垂直方向に 1 ページ下にスクロールします。

ユーザーはマウスを使用し、コンテンツ上および垂直スクロールバーや水平スクロールバー上で ScrollPane を操作することができます。scrollDrag プロパティが true に設定されている場合、ユーザーは、マウスを使用してコンテンツをドラッグできます。コンテンツ上で表示されるハンドポインターは、ユーザーがそのコンテンツをドラッグできることを示しています。他の多くのコントロールとは異なり、アクションはマウスボタンが押されると発生し、ボタンが離されるまで続きます。コンテンツに有効なタブストップが指定されている場合は、scrollDrag を false に設定する必要があります。そうしないと、マウスでコンテンツをクリックするたびにスクロールドラッグが実行されます。

ScrollPane コンポーネントのパラメーター

プロパティインスペクターまたはコンポーネントインスペクターで、ScrollPane の各インスタンスに設定できるパラメーターは、horizontalLineScrollSize、horizontalPageScrollSize、horizontalScrollPolicy、scrollDrag、source、verticalLineScrollSize、verticalPageScrollSize および verticalScrollPolicy です。これらの各パラメーターには、対応する同名の ActionScript プロパティがあります。これらのパラメーターに設定できる値については、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の ScrollPane クラスを参照してください。

ActionScript を記述すると、ScrollPane コンポーネントのプロパティ、メソッド、およびイベントを使用して、コンポーネントのこれらのオプションやその他のオプションを制御できます。

ScrollPane コンポーネントを使用したアプリケーションの作成

次の手順は、オーサリング時に ScrollPane コンポーネントをアプリケーションに追加する方法を示しています。この例では、source プロパティで指定されたパスから ScrollPane に画像をロードします。

- 1 新しい Flash (ActionScript 3.0) ドキュメントを作成します。
- 2 ScrollPane コンポーネントをコンポーネントパネルからステージまでドラッグし、インスタンス名を **aSp** にします。
- 3 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次の ActionScript コードを入力します。

```
import fl.events.ScrollEvent;

aSp.setSize(300, 200);

function scrollListener(event:ScrollEvent):void {
    trace("horizontalScPosition: " + aSp.horizontalScrollPosition +
        ", verticalScrollPosition = " + aSp.verticalScrollPosition);
};
aSp.addEventListener(ScrollEvent.SCROLL, scrollListener);

function completeListener(event:Event):void {
    trace(event.target.source + " has completed loading.");
};
// Add listener.
aSp.addEventListener(Event.COMPLETE, completeListener);

aSp.source = "http://www.helpexamples.com/flash/images/image1.jpg";
```

- 4 制御/ムービープレビューを選択して、アプリケーションを実行します。

ActionScript を使用した ScrollPane インスタンスの作成

この例では、ScrollPane を作成し、そのサイズを設定して、source プロパティを使用してイメージをロードします。さらに、リスナを 2 つ作成します。1 番目のリスナは、scroll イベントをリスンし、ユーザーが垂直方向または水平方向にスクロールしたときに、イメージの位置を表示します。2 番目のリスナは、complete イベントをリスンし、イメージのロードが完了したことを通知するメッセージを出力パネルに表示します。

この例では、ActionScript を使用して ScrollPane を作成し、150 ピクセル (幅) × 300 ピクセル (高さ) の MovieClip (赤色のボックス) を配置します。

- 1 新しい Flash (ActionScript 3.0) ドキュメントを作成します。
- 2 ScrollPane コンポーネントをコンポーネントパネルからライブラリパネルにドラッグします。
- 3 DataGrid コンポーネントをコンポーネントパネルからライブラリパネルにドラッグします。
- 4 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次の ActionScript コードを入力します。

```
import fl.containers.ScrollPane;
import fl.controls.ScrollPolicy;
import fl.controls.DataGrid;
import fl.data.DataProvider;

var aSp:ScrollPane = new ScrollPane();
var aBox:MovieClip = new MovieClip();
drawBox(aBox, 0xFF0000); //draw a red box

aSp.source = aBox;
aSp.setSize(150, 200);
aSp.move(100, 100);

addChild(aSp);

function drawBox(box:MovieClip,color:uint):void {
    box.graphics.beginFill(color, 1);
    box.graphics.drawRect(0, 0, 150, 300);
    box.graphics.endFill();
}
```

- 5 制御/ムービープレビューを選択して、アプリケーションを実行します。

Slider コンポーネントの使用

Slider コンポーネントを使用すると、ユーザーが値の範囲に対応するトラックの両端の間でグラフィカルな「サム」をスライドすることによって、値を選択できます。スライダーを使用すると、例えば、数値やパーセンテージなどの値をユーザーが選択できます。また、ActionScript を使用すると、スライダーの値で別のオブジェクトの動作を制御することもできます。例えば、スライダーを画像に関連付け、スライダーのサムの相対的な位置や値に応じて、その画像を拡大または縮小できます。

Slider の現在の値は、トラックの両端の間、または Slider の最小値と最大値の間のサムの相対的な位置によって決定されます。

Slider には、最小値と最大値の間の連続した値の範囲を指定できますが、snapInterval パラメーターを設定して、最小値と最大値の間で間隔を指定することもできます。Slider には、スライダーに割り当てられた値とは関係なく、指定した間隔でトラックに沿って目盛りを表示できます。

デフォルトでは、スライダーは水平方向ですが、direction パラメーターの値を vertical に設定することで垂直方向にすることができます。スライダーのトラックは端から端まで広がり、目盛りはトラックのすぐ上に左側から右側に向かって配置されます。

Slider コンポーネントのユーザー操作

フォーカスを取得している Slider インスタンスでは、次のキーを使用した操作が可能です。

キー	説明
右向き矢印	水平方向のスライダーに関連付けられた値を増加させます。
上向き矢印	垂直方向のスライダーに関連付けられた値を増加させます。
左向き矢印	水平方向のスライダーに関連付けられた値を減少させます。

キー	説明
下向き矢印	垂直方向のスライダーに関連付けられた値を減少させます。
Shift+Tab	フォーカスを前のオブジェクトに移動します。
Tab	フォーカスを次のオブジェクトに移動します。

フォーカスの制御について詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の IFocusManager インターフェイスと FocusManager クラス、および 26 ページの「[FocusManager の操作](#)」を参照してください。

ライブプレビューでは、オーサリング時にプロパティインスペクターまたはコンポーネントインスペクターで各 Slider インスタンスのパラメーターに加えた変更が反映されます。

Slider コンポーネントのパラメーター

プロパティインスペクターまたはコンポーネントインスペクターで Slider コンポーネントの各インスタンスに設定できるオーサリングパラメーターは、direction、liveDragging、maximum、minimum、snapInterval、tickInterval および value です。これらの各パラメーターには、対応する同名の ActionScript プロパティがあります。これらのパラメーターに設定できる値については、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の Slider クラスを参照してください。

Slider を使用したアプリケーションの作成

次の例では、Slider インスタンスを作成し、ユーザーが、ある特定の事項に関する満足度を回答できるようにしています。ユーザーは Slider を左右に移動することで、回答する満足度の値を増減できます。

- 1 新しい Flash (ActionScript 3.0) ドキュメントを作成します。
- 2 Label コンポーネントをコンポーネントパネルからステージの中央にドラッグします。
 - インスタンス名を **valueLabel** にします。
 - text パラメーターに値 **0percent** を割り当てます。
- 3 Slider コンポーネントをコンポーネントパネルからドラッグし、value_lbl の下の中央に配置します。
 - インスタンス名を **aSlider** にします。
 - 幅 (W) として **200** を割り当てます。
 - 高さ (H) として **10** を割り当てます。
 - 値 **100** を maximum パラメーターに割り当てます。
 - 値 **10** を snapInterval パラメーターと tickInterval パラメーターの両方に割り当てます。
- 4 別の Label インスタンスをライブラリパネルからドラッグし、aSlider の下の中央に配置します。
 - インスタンス名を **promptLabel** にします。
 - 幅 (W) として **250** を割り当てます。
 - 高さ (H) として **22** を割り当てます。
 - text パラメーターとして **満足度を入力してください** と入力します。
- 5 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次の ActionScript コードを入力します。

```
import fl.controls.Slider;
import fl.events.SliderEvent;
import fl.controls.Label;

aSlider.addEventListener(SliderEvent.CHANGE, changeHandler);

function changeHandler(event:SliderEvent):void {
    valueLabel.text = event.value + "percent";
}
```

6 制御/ムービープレビューを選択します。

この例では、スライダーのサムを別の間隔に移動したときに、SliderEvent.CHANGE イベントのリリスナによって valueLabel の text プロパティを更新し、サムの位置に対応するパーセンテージを表示します。

ActionScript で Slider コンポーネントを使用したアプリケーションの作成

次の例では、ActionScript を使用して Slider を作成します。花のイメージをダウンロードし、ユーザーが Slider を使用してそのイメージの明暗を調整できるようにします。具体的には、Slider の値に応じてイメージの alpha プロパティを変更します。

- 1 新しい Flash (ActionScript 3.0) ドキュメントを作成します。
- 2 Label コンポーネントと Slider コンポーネントをコンポーネントパネルから現在のドキュメントのライブラリパネルにドラッグします。
これにより、ライブラリにコンポーネントが追加されます。ただし、アプリケーションには表示されません。
- 3 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次のコードを入力してコンポーネントインスタンスを作成し、配置します。

```
import fl.controls.Slider;
import fl.events.SliderEvent;
import fl.controls.Label;
import fl.containers.UILoader;

var sliderLabel:Label = new Label();
sliderLabel.width = 120;
sliderLabel.text = "< Fade - Brighten >";
sliderLabel.move(170, 350);

var aSlider:Slider = new Slider();
aSlider.width = 200;
aSlider.snapInterval = 10;
aSlider.tickInterval = 10;
aSlider.maximum = 100;
aSlider.value = 100;
aSlider.move(120, 330);

var aLoader:UILoader = new UILoader();
aLoader.source = "http://www.flash-mx.com/images/imagel.jpg";
aLoader.scaleContent = false;

addChild(sliderLabel);
addChild(aSlider);
addChild(aLoader);

aLoader.addEventListener(Event.COMPLETE, completeHandler);

function completeHandler(event:Event) {
    trace("Number of bytes loaded: " + aLoader.bytesLoaded);
}

aSlider.addEventListener(SliderEvent.CHANGE, changeHandler);

function changeHandler(event:SliderEvent):void {
    aLoader.alpha = event.value * .01;
}
```

- 4 制御/ムービープレビューを選択して、アプリケーションを実行します。
- 5 スライダーのサムを左右に移動して、イメージの明暗を調整します。右側に移動するほど明るくなります。

TextArea コンポーネントの使用

TextArea コンポーネントは、ネイティブの ActionScript TextField オブジェクトのラッパーです。TextArea コンポーネントは、テキストの表示に使用できます。また、`editable` プロパティが `true` の場合は、テキスト入力の編集や取得にも使用できます。このコンポーネントでは、複数行のテキストを表示または取得できます。`wordWrap` プロパティが `true` に設定されていると、行数の多いテキストが折り返されます。`restrict` プロパティを使用すると、ユーザーが入力できる文字数を制限できます。また、`maxChars` を使用すると、ユーザーが入力できる最大文字数を指定できます。テキストが、テキスト領域の水平方向または垂直方向の範囲を超えた場合は、水平スクロールバーまたは垂直スクロールバーが自動的に表示されます。ただし、テキスト領域に関連付けられたプロパティである `horizontalScrollPolicy` および `verticalScrollPolicy` が `off` に設定されている場合は表示されません。

TextArea コンポーネントは、複数行のテキストフィールドが必要な場所ならどこでも使用できます。例えば、TextArea コンポーネントをフォームのコメントフィールドに使用することが考えられます。また、ユーザーが Tab キーでフィールドの外に移動したときに、フィールドが空かどうかをチェックするリスナをセットアップすることができます。このリスナを使うと、フィールドにコメントを入力する必要があることをユーザーに知らせるエラーメッセージを表示できます。

単一行のテキストフィールドが必要な場合は、`TextInput` コンポーネントを使用します。

`setStyle()` メソッドを使用して `textFormat` スタイルを設定すると、`TextArea` インスタンスに表示するテキストのスタイルを変更できます。`ActionScript` で `htmlText` プロパティを使用すると、`TextArea` コンポーネントを HTML でフォーマットすることもできます。また、`displayAsPassword` プロパティを `true` に設定すると、テキストをアスタリスクでマスクすることもできます。`condenseWhite` プロパティを `true` に設定すると、スペースや改行などによる余分な空白が新しいテキストから削除されます。これは、既にコントロールに入力済みのテキストでは無効です。

TextArea コンポーネントのユーザー操作

アプリケーション内で `TextArea` コンポーネントを有効または無効にすることができます。無効状態のときは、マウスおよびキーボードからの入力を受け付けることができません。有効状態のときは、`ActionScript TextField` オブジェクトと同じフォーカス、選択、およびナビゲーションの規則に従います。フォーカスを取得している `TextArea` インスタンスでは、次のキーを使用した操作が可能です。

キー	説明
矢印キー	テキストが編集可能な場合は、挿入ポインターをテキスト内で上、下、左、右に移動します。
Page Down	テキストが編集可能な場合は、挿入ポインターをテキストの末尾に移動します。
Page Up	テキストが編集可能な場合は、挿入ポインターをテキストの先頭に移動します。
Shift+Tab	タブグループ内で、フォーカスを前のオブジェクトに移動します。
Tab	タブグループ内で、フォーカスを次のオブジェクトに移動します。

フォーカスの制御について詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の `FocusManager` クラスおよび 26 ページの「[FocusManager の操作](#)」を参照してください。

TextArea コンポーネントのパラメーター

プロパティインスペクターまたはコンポーネントインスペクターで、`TextArea` コンポーネントの各インスタンスに設定できるオーサリングパラメーターは、`condenseWhite`、`editable`、`horizontalScrollPolicy`、`maxChars`、`restrict`、`text`、`verticalScrollPolicy` および `wordwrap` です。これらの各パラメーターには、対応する同名の `ActionScript` プロパティがあります。これらのパラメーターに設定できる値については、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の `TextArea` クラスを参照してください。

各 `TextArea` インスタンスのライブプレビューには、オーサリング時にプロパティインスペクターまたはコンポーネントインスペクターでパラメーターに加えた変更が反映されます。スクロールバーが必要な場合は、ライブプレビューに表示されますが、このスクロールバーは機能しません。ライブプレビューではテキストを選択できず、ステージ上のコンポーネントインスタンスにテキストを入力することはできません。

`ActionScript` を記述すれば、`TextArea` コンポーネントのプロパティ、メソッド、イベントを使用して、これらのオプションやその他のオプションを制御できます。詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の `TextArea` クラスを参照してください。

TextArea コンポーネントを使用したアプリケーションの作成

次の手順は、オーサリング時に `TextArea` コンポーネントをアプリケーションに追加する方法を示しています。この例では、`TextArea` インスタンスの `focusOut` イベントハンドラーをセットアップします。このイベントハンドラーは、インターフェイスの別の場所にフォーカスを移す前に、ユーザーがテキスト領域に入力したかどうかを確認します。

- 1 新しい Flash ドキュメント (`ActionScript 3.0`) を作成します。

- 2 **TextArea** コンポーネントをコンポーネントパネルからステージまでドラッグし、インスタンス名を **aTa** にします。パラメーターは、デフォルト設定のままにします。
- 3 2 番目の **TextArea** コンポーネントをコンポーネントパネルからステージまでドラッグし、1 番目のコンポーネントの下に配置して、インスタンス名を **bTa** にします。パラメーターは、デフォルト設定のままにします。
- 4 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次の **ActionScript** コードを入力します。

```
import flash.events.FocusEvent;

aTa.restrict = "a-z, \\\" \\\";
aTa.addEventListener(Event.CHANGE, changeHandler);
aTa.addEventListener(FocusEvent.KEY_FOCUS_CHANGE, k_m_fHandler);
aTa.addEventListener(FocusEvent.MOUSE_FOCUS_CHANGE, k_m_fHandler);

function changeHandler(ch_evt:Event):void {
    bTa.text = aTa.text;
}
function k_m_fHandler(kmf_event:FocusEvent):void {
    kmf_event.preventDefault();
}
```

この例では、aTa テキスト領域に入力できる文字を、小文字、コンマ、アポストロフィおよびスペースに制限しています。また、aTa テキスト領域の `change`、`KEY_FOCUS_CHANGE` および `MOUSE_FOCUS_CHANGE` の各イベントのイベントハンドラーも設定します。`changeHandler()` 関数によって、各 `change` イベントの発生時に `aTa.text` を `bTa.text` に割り当てることで、aTa テキスト領域に入力するテキストが bTa テキスト領域に自動的に表示されます。

`KEY_FOCUS_CHANGE` イベントと `MOUSE_FOCUS_CHANGE` イベントの `k_m_fHandler()` 関数は、ユーザーがテキストを入力しないまま `Tab` キーを押して次のフィールドに移動できないようにします。この動作は、デフォルトのビヘイビアを回避することで実行されます。

- 5 制御/ムービープレビューを選択します。

テキストを入力せずに `Tab` キーを押して 2 番目のテキスト領域にフォーカスを移動した場合は、エラーメッセージが表示され、1 番目のテキスト領域にフォーカスが戻ります。1 番目のテキスト領域にテキストを入力すると、そのテキストが複製されて 2 番目のテキスト領域に表示されます。

ActionScript を使用した TextArea インスタンスの作成

次の例では、**ActionScript** を使用して **TextArea** コンポーネントを作成します。空白を圧縮するために `condenseWhite` プロパティを `true` に設定し、HTML テキストフォーマットの属性を使用するために `htmlText` プロパティにテキストを割り当てます。

- 1 新しい **Flash (ActionScript 3.0)** ドキュメントを作成します。
- 2 **TextArea** コンポーネントをライブラリパネルにドラッグします。
- 3 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次の **ActionScript** コードを入力します。

```
import fl.controls.TextArea;

var aTa:TextArea = new TextArea();

aTa.move(100,100);
aTa.setSize(200, 200);
aTa.condenseWhite = true;
aTa.htmlText = '<b>Lorem ipsum dolor</b> sit amet, consectetur adipiscing elit. <u>Vivamus quis nisl vel tortor nonummy vulputate.</u> Quisque sit amet eros sed purus euismod tempor. Morbi tempor. <font color="#FF0000">Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos.</font> Curabitur diam. Suspendisse at purus in ipsum volutpat viverra. Nulla pellentesque libero id libero.';
addChild(aTa);
```

この例では、htmlText プロパティを使用してテキストブロックに HTML の bold 属性と underline 属性を適用し、a_ta テキスト領域にそのテキストを表示します。また、condenseWhite プロパティを true に設定してテキストブロック内の空白を圧縮します。setSize() メソッドでテキスト領域の高さと幅を設定し、move() メソッドでその位置を設定します。addChild() メソッドで TextArea インスタンスをステージに追加します。

4 制御/ムービープレビューを選択します。

TextInput コンポーネントの使用

TextInput コンポーネントは、ネイティブの ActionScript TextField オブジェクトのラッパーである単一行テキストコンポーネントです。複数行のテキストフィールドが必要な場合は、TextArea コンポーネントを使用します。例えば、TextInput コンポーネントをフォームのパスワードフィールドに使用することが考えられます。また、ユーザーが Tab キーでフィールドの外に移動したときに、フィールドに十分な文字が入力されているかどうかをチェックするリスナをセットアップすることができます。このリスナを使うと、適切な文字数を入力する必要があることをユーザーに知らせるエラーメッセージを表示できます。

setStyle() メソッドを使用して textFormat プロパティを設定すると、TextInput インスタンスに表示するテキストのスタイルを変更できます。TextInput コンポーネントは、HTML 形式でフォーマットすることも、テキストを隠すパスワードフィールドとしてフォーマットすることもできます。

TextInput コンポーネントのユーザー操作

アプリケーション内で TextInput コンポーネントを有効または無効にすることができます。無効状態のときは、マウスおよびキーボードからの入力を受け付けません。有効状態のときは、ActionScript TextField オブジェクトと同じフォーカス、選択、およびナビゲーションの規則に従います。フォーカスを取得している TextInput インスタンスでは、次のキーを使用した操作も可能です。

キー	説明
矢印キー	挿入ポインターを 1 文字左または右に移動します。
Shift+Tab	フォーカスを前のオブジェクトに移動します。
Tab	フォーカスを次のオブジェクトに移動します。

フォーカスの制御について詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の FocusManager クラスおよび 26 ページの「[FocusManager の操作](#)」を参照してください。

各 TextInput インスタンスのライブプレビューには、オーサリング時にプロパティインスペクターまたはコンポーネントインスペクターでパラメーターに加えた変更が反映されます。ライブプレビューではテキストを選択できず、ステージ上のコンポーネントインスタンスにテキストを入力することはできません。

TextInput コンポーネントをアプリケーションに追加する際にアクセシビリティパネルを使用することで、そのコンポーネントをスクリーンリーダーが読み取れるように設定できます。

TextInput コンポーネントのパラメーター

プロパティインスペクターまたはコンポーネントインスペクターで、TextInput コンポーネントの各インスタンスに設定できるオーサリングパラメーターは、editable、displayAsPassword、maxChars、restrict および text です。これらの各パラメーターには、対応する同名の ActionScript プロパティがあります。これらのパラメーターに設定できる値については、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の TextInput クラスを参照してください。

ActionScript を記述すれば、TextInput コンポーネントのプロパティ、メソッド、イベントを使用して、これらのオプションやその他のオプションを制御できます。詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の TextInput クラスを参照してください。

TextInput コンポーネントを使用したアプリケーションの作成

次の手順は、TextInput コンポーネントをアプリケーションに追加する方法を示しています。この例では、2つの TextInput フィールドを使用して、パスワードを入力および確認します。イベントリスナを使用して、最小文字数である 8 文字が入力されたことと、2つのフィールドのテキストが一致していることを確認します。

- 1 新しい Flash (ActionScript 3.0) ドキュメントを作成します。
- 2 Label コンポーネントをコンポーネントパネルからステージにドラッグし、プロパティインスペクターで次の値を設定します。
 - インスタンス名として **pwdLabel** と入力します。
 - W 値として **100** と入力します。
 - X 値として **50** と入力します。
 - Y 値として **150** と入力します。
 - 「パラメーター」セクションで、text パラメーターの値として **パスワード** と入力します。
- 3 2 番目の Label コンポーネントをコンポーネントパネルからステージにドラッグし、次の値を設定します。
 - インスタンス名として **confirmLabel** と入力します。
 - W 値として **100** と入力します。
 - X 値として **50** と入力します。
 - Y 値として **200** と入力します。
 - 「パラメーター」セクションで、text パラメーターの値として **パスワードの確認** と入力します。
- 4 TextInput コンポーネントをコンポーネントパネルからステージにドラッグし、次の値を設定します。
 - インスタンス名として **pwdTi** と入力します。
 - W 値として **150** と入力します。
 - X 値として **190** と入力します。
 - Y 値として **150** と入力します。
 - 「パラメーター」セクションで、displayAsPassword パラメーターの値をダブルクリックし、**true** を選択します。これにより、テキストフィールドに入力された値がアスタリスクでマスクされるようになります。
- 5 2 番目の TextInput コンポーネントをコンポーネントパネルからステージにドラッグし、次の値を設定します。
 - インスタンス名として **confirmTi** と入力します。
 - W 値として **150** と入力します。
 - X 値として **190** と入力します。
 - Y 値として **200** と入力します。
 - 「パラメーター」セクションで、displayAsPassword パラメーターの値をダブルクリックし、**true** を選択します。これにより、テキストフィールドに入力された値がアスタリスクでマスクされるようになります。
- 6 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次の ActionScript コードを入力します。

```
function tiListener(evt_obj:Event){
    if(confirmTi.text != pwdTi.text || confirmTi.length < 8)
    {
        trace("Password is incorrect. Please reenter it.");
    }
    else {
        trace("Your password is: " + confirmTi.text);
    }
}
confirmTi.addEventListener("enter", tiListener);
```

このコードでは、confirmTi という TextInput インスタンスの enter イベントハンドラーをセットアップします。2つのパスワードが一致しない場合や、ユーザーの入力した文字が 8 文字未満の場合は、「パスワードが正しくありません。再入力してください。」というメッセージが表示されます。パスワードが 8 文字以上で 2つが一致している場合は、入力された値を出力パネルに表示します。

- 7 制御/ムービープレビューを選択します。

ActionScript を使用した TextInput インスタンスの作成

次の例では、ActionScript を使用して TextInput コンポーネントを作成します。さらに、ユーザーに名前を入力を要求するために使用する Label も作成します。この例では、大文字と小文字、ピリオドおよびスペースのみを使用できるようにコンポーネントの restrict プロパティを設定します。また、Label コンポーネントと TextInput コンポーネントの両方のテキストをフォーマットするために使用する TextFormat オブジェクトも作成します。

- 1 新しい Flash (ActionScript 3.0) ドキュメントを作成します。
- 2 TextInput コンポーネントをコンポーネントパネルからライブラリパネルにドラッグします。
- 3 Label コンポーネントをコンポーネントパネルからライブラリパネルにドラッグします。
- 4 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次の ActionScript コードを入力します。

```
import fl.controls.Label;
import fl.controls.TextInput;

var nameLabel:Label = new Label();
var nameTi:TextInput = new TextInput();
var tf:TextFormat = new TextFormat();

addChild(nameLabel);
addChild(nameTi);

nameTi.restrict = "A-Z .a-z";

tf.font = "Georgia";
tf.color = 0x0000CC;
tf.size = 16;

nameLabel.text = "Name: ";
nameLabel.setSize(50, 25);
nameLabel.move(100,100);
nameLabel.setStyle("textFormat", tf);
nameTi.move(160, 100);
nameTi.setSize(200, 25);
nameTi.setStyle("textFormat", tf);
```

- 5 制御/ムービープレビューを選択して、アプリケーションを実行します。

TileList コンポーネントの使用

TileList コンポーネントは、データプロバイダーからのデータが入力された行と列から成るリストで構成されます。アイテムとは、TileList のセル内に格納されるデータの単位です。データプロバイダーによって指定されるアイテムには、通常、label プロパティと source プロパティがあります。label プロパティでセル内に表示するコンテンツを識別し、source プロパティでその値を指定します。

Array インスタンスは、作成することも、サーバーから取得することもできます。TileList コンポーネントには、データプロバイダーの代理となるメソッドが用意されています (例えば addItem() メソッドや removeItem() メソッドなど)。リストに外部のデータプロバイダーを指定しない場合、これらのメソッドによってデータプロバイダーインスタンスが自動的に作成され、List.dataProvider を通じて公開されます。

TileList コンポーネントのユーザー操作

TileList は、ICellRenderer インターフェイスを実装した Sprite を使用して、各セルをレンダリングします。このレンダラーは、TileList の cellRenderer プロパティで指定できます。TileList コンポーネントのデフォルトの CellRenderer は ImageCell です。これにより、イメージ (クラス、ビットマップ、インスタンスまたは URL) やオプションのラベルを表示します。このラベルは単一行で、常にセルの下部に配置されます。TileList は、1 方向だけにスクロールすることができます。

フォーカスを取得している TileList インスタンスでは、次のキーを使用して中のアイテムにアクセスすることも可能です。

キー	説明
上向き矢印、下向き矢印	列を上下に移動できます。allowMultipleSelection プロパティが true の場合は、これらのキーを Shift キーと組み合わせて使用し、複数のセルを選択できます。
左向き矢印、右向き矢印	行を左右に移動できます。allowMultipleSelection プロパティが true の場合は、これらのキーを Shift キーと組み合わせて使用し、複数のセルを選択できます。
Home	TileList の先頭のセルを選択します。allowMultipleSelection プロパティが true の場合は、Shift キーを押しながら Home キーを押すと、現在の位置から先頭のセルまでのすべてのセルが選択されます。
End	TileList の最後のセルを選択します。allowMultipleSelection プロパティが true の場合は、Shift キーを押しながら End キーを押すと、現在の位置から最後のセルまでのすべてのセルが選択されます。
Ctrl	allowMultipleSelection プロパティが true に設定されている場合は、複数のセルを任意の順序で選択できます。

TileList コンポーネントをアプリケーションに追加する際に、次の ActionScript コード行を追加することで、そのコンポーネントをスクリーンリーダーが読み取れるように設定できます。

```
import fl.accessibility.TileListAccImpl;  
  
TileListAccImpl.enableAccessibility();
```

1 つのコンポーネントに対してアクセシビリティ機能を有効化する処理は、そのコンポーネントがインスタンスをいくつ持つかに関係なく、一度だけ実行します。詳しくは、『Flash ユーザーガイド』の第 18 章「アクセシビリティコンテンツの作成」を参照してください。

TileList コンポーネントのパラメーター

プロパティインスペクターまたはコンポーネントインスペクターで TileList コンポーネントの各インスタンスに設定できるオーサリングパラメーターは、allowMultipleSelection、columnCount、columnWidth、dataProvider、direction、horizontalScrollLineSize、horizontalScrollPageSize、labels、rowCount、rowHeight、ScrollPolicy、verticalScrollLineSize、

verticalScrollPageSize です。これらの各パラメーターには、対応する同名の ActionScript プロパティがあります。dataProvider パラメーターの使用について詳しくは、28 ページの「[dataProvider パラメーターの使用](#)」を参照してください。

ActionScript を記述すれば、TileList コンポーネントのメソッド、プロパティ、イベントを使用して、TileList インスタンスのその他のオプションを設定できます。詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の TileList クラスを参照してください。

TileList コンポーネントを使用したアプリケーションの作成

この例では、複数の MovieClip を使用して、TileList にペイントカラーの配列を格納します。

- 1 新しい Flash (ActionScript 3.0) ドキュメントを作成します。
- 2 TileList コンポーネントをステージにドラッグし、インスタンス名を **aTl** にします。
- 3 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次の ActionScript コードを入力します。

```
import fl.data.DataProvider;
import flash.display.DisplayObject;

var aBoxes:Array = new Array();
var i:uint = 0;
var colors:Array = new Array(0x000000, 0xFF0000, 0x0000CC, 0x00CC00, 0xFFFF00);
var colorNames:Array = new Array("Midnight", "Cranberry", "Sky", "Forest", "July");
var dp:DataProvider = new DataProvider();
for(i=0; i < colors.length; i++) {
    aBoxes[i] = new MovieClip();
    drawBox(aBoxes[i], colors[i]); // draw box w next color in array
    dp.addItem( {label:colorNames[i], source:aBoxes[i]} );
}
aTl.dataProvider = dp;
aTl.columnWidth = 110;
aTl.rowHeight = 130;
aTl.setSize(280,150);
aTl.move(150, 150);
aTl.setStyle("contentPadding", 5);

function drawBox(box:MovieClip,color:uint):void {
    box.graphics.beginFill(color, 1.0);
    box.graphics.drawRect(0, 0, 100, 100);
    box.graphics.endFill();
}
```

- 4 制御/ムービープレビューを選択してアプリケーションをテストします。

ActionScript を使用した TileList コンポーネントの作成

この例では、TileList インスタンスを動的に作成し、ColorPicker、ComboBox、NumericStepper、CheckBox の各コンポーネントのインスタンスを追加します。表示するコンポーネントのラベルと名前が格納された Array を作成し、その Array (dp) を TileList の dataProvider プロパティに割り当てます。columnWidth プロパティと rowHeight プロパティを使用し、setSize() メソッドで TileList のレイアウトを指定して、move() メソッドで TileList をステージに配置します。また、contentPadding スタイルで TileList インスタンスの境界線とコンテンツの間に余白をあげ、sortItemsOn() メソッドでコンテンツをラベルの順序で並べ替えます。

- 1 新しい Flash (ActionScript 3.0) ドキュメントを作成します。
- 2 ColorPicker、ComboBox、NumericStepper、CheckBox、および TileList の各コンポーネントをコンポーネントパネルからライブラリパネルにドラッグします。
- 3 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次の ActionScript コードを入力します。

```
import fl.controls.CheckBox;
import fl.controls.ColorPicker;
import fl.controls.ComboBox;
import fl.controls.NumericStepper;
import fl.controls.TileList;
import fl.data.DataProvider;

var aCp:ColorPicker = new ColorPicker();
var aCb:ComboBox = new ComboBox();
var aNs:NumericStepper = new NumericStepper();
var aCh:CheckBox = new CheckBox();
var aTl:TileList = new TileList();

var dp:Array = [
    {label:"ColorPicker", source:aCp},
    {label:"ComboBox", source:aCb},
    {label:"NumericStepper", source:aNs},
    {label:"CheckBox", source:aCh},
];
aTl.dataProvider = new DataProvider(dp);
aTl.columnWidth = 110;
aTl.rowHeight = 100;
aTl.setSize(280,130);
aTl.move(150, 150);
aTl.setStyle("contentPadding", 5);
aTl.sortItemsOn("label");
addChild(aTl);
```

- 4 制御/ムービープレビューを選択してアプリケーションをテストします。

UI Loader コンポーネントの使用

UI Loader コンポーネントは、SWF、JPEG、プログレッシブ JPEG、PNG および GIF の各ファイルを表示できるコンテナです。異なる場所からコンテンツを取得して Flash アプリケーションに読み込む必要がある場合には UI Loader コンポーネントを使用します。例えば、UI Loader を使用して企業のロゴ (JPEG ファイル) をフォームに追加することが可能です。また、アプリケーションで UI Loader コンポーネントを使用すると、写真を表示することもできます。load() メソッドを使用してコンテンツをロードし、percentLoaded プロパティでどれだけのコンテンツがロードされたかを検出し、complete イベントでロードが完了するタイミングを検出します。

UI Loader のコンテンツを拡大および縮小したり、コンテンツのサイズに合わせて UI Loader 自身のサイズを変更したりできます。デフォルトでは、UI Loader に合わせてコンテンツが拡大または縮小されます。さらに、実行時にコンテンツをロードして、ロードの進行状況を監視できます (ただし、コンテンツは一度ロードされるとキャッシュに残るので、ロードの進行状況はすぐに 100% に達します)。UI Loader にコンテンツをロードするときに位置を指定する場合は、その位置 (X 座標および Y 座標) を 0, 0 と指定する必要があります。

UI Loader コンポーネントのユーザー操作

UI Loader コンポーネントはフォーカスを取得できません。ただし、UI Loader コンポーネント内にロードしたコンテンツ自体は、フォーカスの取得やフォーカス操作が可能です。フォーカスの制御について詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の FocusManager クラスおよび 26 ページの「[FocusManager の操作](#)」を参照してください。

UILoader コンポーネントのパラメーター

プロパティインスペクターまたはコンポーネントインスペクターで UILoader コンポーネントの各インスタンスに設定できるオーサリングパラメーターは、autoLoad、maintainAspectRatio、source および scaleContent です。これらの各パラメーターには、対応する同名の ActionScript プロパティがあります。

ライブプレビューでは、オーサリング時にプロパティインスペクターまたはコンポーネントインスペクターで各 UILoader インスタンスのパラメーターに加えた変更が反映されます。

ActionScript を記述すれば、UILoader コンポーネントのメソッド、プロパティ、イベントを使用して、UILoader インスタンスのその他のオプションを設定できます。詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の UILoader クラスを参照してください。

UILoader コンポーネントを使用したアプリケーションの作成

次の手順は、オーサリング時に UILoader コンポーネントをアプリケーションに追加する方法を示しています。この例では、ローダーでロゴの GIF イメージをロードします。

- 1 新しい Flash (ActionScript 3.0) ドキュメントを作成します。
- 2 UILoader コンポーネントをコンポーネントパネルからステージまでドラッグします。
- 3 プロパティインスペクターで、インスタンス名として **aUI** と入力します。
- 4 ステージ上およびコンポーネントインスペクターでローダーを選択して、source パラメーターに <http://www.helpexamples.com/images/logo.gif> と入力します。

ActionScript を使用した UILoader コンポーネントインスタンスの作成

この例では、ActionScript を使用して UILoader コンポーネントを作成し、花の JPEG イメージをロードします。complete イベントが発生すると、ロードされたバイト数が出力パネルに表示されます。

- 1 新しい Flash (ActionScript 3.0) ドキュメントを作成します。
- 2 UILoader コンポーネントをコンポーネントパネルからライブラリパネルにドラッグします。
- 3 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次の ActionScript コードを入力します。

```
import fl.containers.UILoader;

var aLoader:UILoader = new UILoader();
aLoader.source = "http://www.flash-mx.com/images/image1.jpg";
aLoader.scaleContent = false;
addChild(aLoader);

aLoader.addEventListener(Event.COMPLETE, completeHandler);
function completeHandler(event:Event) {
    trace("Number of bytes loaded: " + aLoader.bytesLoaded);
}
```

- 4 制御/ムービープレビューを選択します。

UIScrollBar コンポーネントの使用

UIScrollBar コンポーネントを使用すると、スクロールバーをテキストフィールドに追加できます。テキストフィールドのスクロールバーは、オーサリング時に追加することも、ActionScript を使用して実行時に追加することもできます。

UIScrollBar コンポーネントを使用するには、テキストフィールドをステージ上に作成し、UIScrollBar コンポーネントをコンポーネントパネルからテキストフィールドの境界ボックス上の任意の辺にドラッグします。

スクロールバーの長さが、両端の矢印ボタンの合計サイズより小さい場合は、正しく表示されません。矢印ボタンの一方が、もう一方の下に隠れます。Flash では、これに関するエラーチェックは実行されません。そのような場合は、ActionScript を使用してスクロールバーを非表示にすることをお勧めします。スクロールバーのサイズがスクロールボックス (サム) を表示できないほど小さい場合、Flash ではスクロールボックスが非表示になります。

UIScrollBar コンポーネントは、他のスクロールバーと同じように機能します。両端に矢印ボタンがあり、その間にスクロールトラックとスクロールボックス (サム) があります。UIScrollBar コンポーネントは、テキストフィールドの任意の端に追加でき、垂直にも水平にも使用できます。

TextField については、「**Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド**」の TextField クラスを参照してください。

UIScrollBar コンポーネントのユーザー操作

他の多くのコンポーネントとは異なり、UIScrollBar コンポーネントでは、連続的なマウス入力を受け付けます。例えば、クリックを繰り返すのではなくマウスボタンを押したままでの操作ができます。

UIScrollBar コンポーネントは、キーボードで操作することはできません。

UIScrollBar コンポーネントのパラメーター

プロパティインスペクターまたはコンポーネントインスペクターで UIScrollBar コンポーネントの各インスタンスに設定できるオーサリングパラメーターは、direction および scrollTargetName です。これらの各パラメーターには、対応する同名の ActionScript プロパティがあります。

ActionScript を記述すれば、UIScrollBar コンポーネントのメソッド、プロパティ、イベントを使用して、UIScrollBar インスタンスのその他のオプションを設定できます。詳しくは、「**Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド**」の UIScrollBar クラスを参照してください。

UIScrollBar コンポーネントを使用したアプリケーションの作成

次の手順は、オーサリング時に UIScrollBar コンポーネントをアプリケーションに追加する方法を示しています。

- 1 新しい Flash (ActionScript 3.0) ドキュメントを作成します。
- 2 1 行または 2 行のテキストを格納するのに十分な高さを持つダイナミックテキストフィールドを作成し、プロパティインスペクターでインスタンス名を **myText** にします。
- 3 プロパティインスペクターで、テキスト入力フィールドの「行タイプ」を「複数行」に設定します。または、スクロールバーを水平方向で使用する場合は、「折り返しなし」に設定します。
- 4 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次の ActionScript コードを入力します。このコードで、ユーザーがテキストをすべて表示するにはスクロールが必要となるように、text プロパティが設定されます。

```
myText.text="When the moon is in the seventh house and Jupiter aligns with Mars, then peace will guide the planet and love will rule the stars."
```

注意：スクロールしないとすべてのテキストを表示できないように、ステージ上のテキストフィールドは小さくしておいてください。そうしないと、スクロールバーが表示されないか、2 行だけ表示されてサムグリッ (コンテンツをスクロールする際にドラッグする部分) が表示されません。

- 5 オブジェクトの吸着がオンになっていることを確認します (表示/吸着/オブジェクトの吸着)。
- 6 テキスト入力フィールド上の、UIScrollBar インスタンスを追加する端の付近に、コンポーネントパネルから UIScrollBar インスタンスをドラッグします。コンポーネントをテキストフィールドに適切にバインドするためには、マ

ウスを離すとき、コンポーネントがテキストフィールドに重なっている必要があります。インスタンス名を **mySb** にします。

コンポーネントの `scrollTargetName` プロパティには、プロパティインスペクターおよびコンポーネントインスペクターでのテキストフィールドのインスタンス名が自動的に反映されます。「パラメーター」タブに表示されない場合は、`UIScrollBar` インスタンスの重なりが十分でない可能性があります。

- 7 制御/ムービープレビューを選択します。

ActionScript を使用した UIScrollBar コンポーネントインスタンスの作成

ActionScript を使用して、実行時に `UIScrollBar` インスタンスを作成し、テキストフィールドに関連付けることができます。次の例では、水平方向の `UIScrollBar` インスタンスを作成し、**myTxt** という名前のテキストフィールドインスタンスの下部に割り当てます。このフィールドには、特定の URL からテキストがロードされます。また、テキストフィールドのサイズに一致するように、スクロールバーのサイズを設定します。

- 1 新しい Flash (ActionScript 3.0) ドキュメントを作成します。
- 2 `ScrollBar` コンポーネントをライブラリパネルにドラッグします。
- 3 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次の ActionScript コードを入力します。

```
import flash.net.URLLoader;
import fl.controls.UIScrollBar;
import flash.events.Event;

var myTxt:TextField = new TextField();
myTxt.border = true;
myTxt.width = 200;
myTxt.height = 16;
myTxt.x = 200;
myTxt.y = 150;

var mySb:UIScrollBar = new UIScrollBar();
mySb.direction = "horizontal";
// Size it to match the text field.
mySb.setSize(myTxt.width, myTxt.height);

// Move it immediately below the text field.
mySb.move(myTxt.x, myTxt.height + myTxt.y);

// put them on the Stage
addChild(myTxt);
addChild(mySb);
// load text
var loader:URLLoader = new URLLoader();
var request:URLRequest = new URLRequest("http://www.helpexamples.com/flash/lorem.txt");
loader.load(request);
loader.addEventListener(Event.COMPLETE, loadcomplete);

function loadcomplete(event:Event) {
    // move loaded text to text field
    myTxt.text = loader.data;
    // Set myTxt as target for scroll bar.
    mySb.scrollTarget = myTxt;
}
```

- 4 制御/ムービープレビューを選択します。

第5章：UI コンポーネントのカスタマイズ

UI コンポーネントのカスタマイズについて

次の1つまたは両方のエレメントを変更すると、アプリケーションでのコンポーネントの外観をカスタマイズできます。

スタイル 各コンポーネントには、その外観を描画するために Flash で使用する値を指定できる、一連のスタイルがあります。スタイルによって、通常はコンポーネントのさまざまな状態に使用するスキンとアイコン、および使用するテキストフォーマットとパディングの値が指定されます。

スキン 特定の状態のコンポーネントのグラフィック表示を構成するシンボルのコレクションです。使用するスキンはスタイルによって指定されますが、スキンはコンポーネントを描画するために Flash で使用されるグラフィカルエレメントです。スキンの適用は、グラフィックを修正または置換することでコンポーネントの外観を変更するプロセスです。

注意： ActionScript 3.0 コンポーネントのデフォルトの外観はテーマ (Aeon Halo) と見なすことができますが、これらのスキンはコンポーネントに組み込まれています。ActionScript 3.0 コンポーネントは、ActionScript 2.0 コンポーネントのように外部テーマファイルをサポートしていません。

スタイルの設定

Flash で様々な状態のコンポーネントが描画される時、通常はコンポーネントのスタイルによって、そのスキン、アイコン、テキストフォーマットおよびパディングの値が指定されます。例えば、Button がダウン状態のときは、アップ状態または通常状態とは異なるスキンで描画されます。これが発生するのは、Button の上でマウスボタンをクリックしたときです。また、無効状態のときも別のスキンが使用されます。この状態は、enabled プロパティを false に設定すると発生します。

コンポーネントのスタイルは、ドキュメント、クラスおよびインスタンスの各レベルで設定します。さらに、一部のスタイルプロパティは親コンポーネントから継承できます。たとえば、List コンポーネントは BaseScrollPane からの継承によって ScrollBar スタイルを継承します。

コンポーネントをカスタマイズするスタイルは、次のように設定できます。

- コンポーネントインスタンスにスタイルを設定する。個々のコンポーネントインスタンスで、カラーやテキストのプロパティを変更することができます。これは場合によっては効果的ですが、ドキュメント内のすべてのコンポーネントについて個別にプロパティを設定する必要がある場合などは、かえって時間がかかる可能性もあります。
- ドキュメントの特定のタイプのすべてのコンポーネントのスタイルを設定する。ドキュメント内のすべての CheckBox またはすべての Button など、特定のタイプのすべてのコンポーネントのスタイルに一貫性を持たせるには、コンポーネントレベルでスタイルを設定します。

コンテナにスタイルのプロパティを設定すると、そのコンテナに含まれるコンポーネントにプロパティの値が継承されません。

ライブプレビュー機能を使ってステージにコンポーネントを表示する場合、スタイルプロパティに加えた変更は表示されません。

スタイル設定について

スタイルの使用に関しては次のような点が重要です。

継承 コンポーネントの子は、設計上、親コンポーネントからスタイルを継承するように設定されます。ActionScript 内でスタイルの継承を設定することはできません。

優先順位 同じコンポーネントスタイルを複数の方法で設定した場合、優先順位に従って最初に検出されたスタイルが使用されます。Flash は値が見つかるまで次の順序でスタイルを調べます。

- 1 コンポーネントインスタンスのスタイルプロパティを探します。
- 2 継承スタイルに含まれるスタイルの場合は、親階層から継承される値を探します。
- 3 コンポーネント上のスタイルを探します。
- 4 StyleManager のグローバル設定を探します。
- 5 検索の結果、そのプロパティが定義されていない場合は、値として `undefined` が使用されます。

コンポーネントのデフォルトスタイルへのアクセス

コンポーネントクラスの静的 `getStyleDefinition()` メソッドを使用するコンポーネントでは、そのデフォルトスタイルにアクセスできます。例えば、次のコードは `ComboBox` コンポーネントのデフォルトスタイルを取得し、`buttonWidth` プロパティおよび `downArrowDownSkin` プロパティのデフォルト値を表示します。

```
import fl.controls.ComboBox;
var styleObj:Object = ComboBox.getStyleDefinition();
trace(styleObj.buttonWidth); // 24
trace(styleObj.downArrowDownSkin); // ScrollArrowDown_downSkin
```

コンポーネントインスタンスのスタイルの設定と取得

任意の UI コンポーネントインスタンスは、`setStyle()` メソッドおよび `getStyle()` メソッドを直接呼び出して、スタイルを設定したり、取得したりできます。次のシンタックスで、コンポーネントインスタンスのスタイルおよび値を指定します。

```
instanceName.setStyle("styleName", value);
```

このシンタックスで、コンポーネントインスタンスのスタイルを取得します。

```
var a_style:Object = new Object();
a_style = instanceName.getStyle("styleName");
```

`getStyle()` メソッドは `Object` 型を返します。これは、異なるデータ型を持つ複数のスタイルを返すことができるためです。例えば、次のコードは `TextArea` インスタンス (`aTa`) のフォントスタイルを設定し、`getStyle()` メソッドを使用してそのスタイルを取得します。この例では、戻り値を `TextFormat` オブジェクトにキャストし、その値を `TextFormat` 変数に割り当てます。キャストを行わないと、コンパイラーは `Object` 変数を `TextFormat` 変数に強制しようとしてエラーが発生します。

```
import flash.text.TextFormat;

var tf:TextFormat = new TextFormat();
tf.font = "Georgia";
aTa.setStyle("textFormat", tf);
aTa.text = "Hello World!";
var aStyle:TextFormat = aTa.getStyle("textFormat") as TextFormat;
trace(aStyle.font);
```

TextFormat を使用したテキストプロパティの設定

コンポーネントインスタンスのテキストをフォーマットするには、`TextFormat` オブジェクトを使用します。`TextFormat` オブジェクトには、`bold`、`bullet`、`color`、`font`、`italic`、`size` など、テキスト特性を指定できるプロパティがあります。これらのプロパティは `TextFormat` オブジェクトで設定し、`setStyle()` メソッドを呼び出してコンポーネントインスタンスに適用します。例えば、次のコードでは `TextFormat` オブジェクトの `font`、`size` および `bold` プロパティを設定し、それらのプロパティを `Button` インスタンスに適用します。

```
/* Create a new TextFormat object to set text formatting properties. */  
var tf:TextFormat = new TextFormat();  
tf.font = "Arial";  
tf.size = 16;  
tf.bold = true;  
a_button.setStyle("textFormat", tf);
```

次の図は、Submit ラベルを持つボタンでの、これらの設定の効果を示しています。



setStyle() を使用してコンポーネントインスタンスに設定したスタイルプロパティは、最も優先度が高く、その他のスタイル設定よりも優先されます。ただし、単一のコンポーネントインスタンスに対して setStyle() を使用して設定したプロパティが多ければ多いほど、該当するコンポーネントの実行時のレンダリング速度は遅くなります。

コンポーネントのすべてのインスタンスへのスタイルの設定

StyleManager クラスの静的 setComponentStyle() メソッドを使用すると、コンポーネントクラスのすべてのインスタンスのスタイルを設定できます。例えば、すべての Button のテキストのカラーを赤に設定するには、最初に Button をステージまでドラッグし、次の ActionScript コードをタイムラインのフレーム 1 のアクションパネルに追加します。

```
import fl.managers.StyleManager;  
import fl.controls.Button;  
  
var tf:TextFormat = new TextFormat();  
tf.color = 0xFF0000;  
StyleManager.setComponentStyle(Button, "textFormat", tf);
```

ステージにそれ以降に追加するすべての Button のラベルは赤になります。

すべてのコンポーネントのスタイルの設定

StyleManager クラスの静的 setStyle() メソッドを使用すると、すべてのコンポーネントにスタイルを設定できます。

- 1 List コンポーネントをステージまでドラッグし、インスタンス名を **aList** にします。
- 2 Button コンポーネントをステージまでドラッグし、インスタンス名を **aButton** にします。
- 3 アクションパネルが開いていない場合は、**F9** キーを押すか、ウィンドウメニューからアクションを選択してアクションパネルを開きます。タイムラインのフレーム 1 に次のコードを入力して、すべてのコンポーネントのテキストのカラーを赤に設定します。

```
import fl.managers.StyleManager;  
  
var tf:TextFormat = new TextFormat();  
tf.color = 0xFF0000;  
StyleManager.setStyle("textFormat", tf);
```

- 4 次のコードをアクションパネルに追加して List にテキストを埋め込みます。

```
aList.addItem({label:"1956 Chevy (Cherry Red)", data:35000});  
aList.addItem({label:"1966 Mustang (Classic)", data:27000});  
aList.addItem({label:"1976 Volvo (Xcllnt Cond)", data:17000});  
aList.allowMultipleSelection = true;
```

- 5 制御/ムービーをプレビューするを選択するか、**Ctrl + Enter** キーを押してコードをコンパイルし、コンテンツをテストします。ボタンのラベルとリストの両方のテキストが赤になります。

スキンについて

コンポーネントの外観は、アウトライン、塗りのカラー、アイコンおよびその他のコンポーネントのグラフィカルエレメントで構成されます。例えば、ComboBox には List コンポーネントが含まれ、List コンポーネントには ScrollBar が含まれます。これらのグラフィカルエレメントによって ComboBox の外観が構成されます。ただし、コンポーネントの外観は現在の状態に応じて変わります。例えば、CheckBox は、ラベルがないとアプリケーションで次のように表示されます。



通常のアップ状態の CheckBox

CheckBox 上でマウスボタンを押したままにすると、その外観は次のようになります。



ダウン状態の CheckBox

マウスボタンを離すと、CheckBox は元の外観に戻りますが、選択されたことを示すチェックマークが付きます。



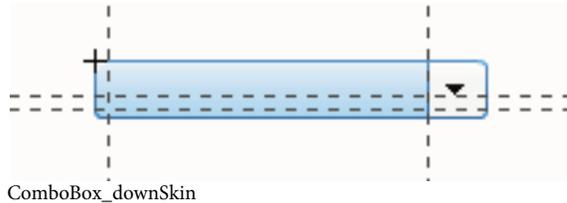
選択状態の CheckBox

コンポーネントを様々な状態で表すアイコンを総称してスキンと呼びます。コンポーネントの外観は、Flash のその他のシンボルの場合と同じように、そのスキンを Flash で編集すると、任意の状態またはすべての状態の外観を変更できます。次の 2 つの方法で、コンポーネントのスキンにアクセスできます。最も簡単なのは、コンポーネントをステージにドラッグしてダブルクリックする方法です。この操作を行うと、コンポーネントのスキンのパレットが開きます。CheckBox の場合、その外観は次のようになります。



CheckBox のスキン

コンポーネントのスキンはライブラリパネルから個別にアクセスすることもできます。コンポーネントをステージにドラッグすると、コンポーネントはそのアセットおよび含まれるその他のコンポーネントのフォルダーと共にライブラリにコピーされます。例えば、ComboBox をステージにドラッグする場合、ライブラリパネルには List、ScrollBar および TextInput の各コンポーネントも含まれており、これらは各コンポーネントのスキンのフォルダー、およびこれらのコンポーネントが共有するエレメントを含む共有アセットフォルダーと共に ComboBox に組み込まれています。これらのコンポーネントのスキンを編集するには、スキンフォルダー ("ComboBoxSkins"、"ListSkins"、"ScrollBarSkins"、または "TextInputSkins") を開き、編集するスキンのアイコンをダブルクリックします。例えば、ComboBox_downSkin をダブルクリックすると、次の図のようにスキンがシンボル編集モードで開きます。



スキンの新規作成

ドキュメント内のコンポーネントの新しい外観を作成するには、コンポーネントのスキンを編集して外観を変更します。コンポーネントのスキンにアクセスするには、ステージ上のコンポーネントをダブルクリックして、そのスキンのパレットを開きます。次に、編集するスキンをダブルクリックし、シンボル編集モードで開きます。例えば、ステージ上の TextArea コンポーネントをダブルクリックして、そのアセットをシンボル編集モードで開きます。ズームコントロールを 400% または必要に応じてそれ以上に設定し、シンボルを編集して外観を変更します。完了すると、ドキュメント内のコンポーネントのすべてのインスタンスに変更内容が影響します。この他にも、ライブラリパネルの特定のスキンをダブルクリックし、ステージ上でシンボル編集モードで開く方法があります。

コンポーネントのスキンは次の方法で変更できます。

- すべてのインスタンスの新しいスキンを作成する
- 一部のインスタンスの新しいスキンを作成する

すべてのインスタンスのスキンの作成

コンポーネントのスキンを編集する場合、デフォルトではドキュメント内のそのコンポーネントのすべてのインスタンスの外観が変更されます。同じコンポーネントに対して異なる外観を作成する場合は、変更するスキンを複製し、別の名前を付けて編集し、適切なスタイルを設定して適用する必要があります。詳しくは、97 ページの「[一部のインスタンスのスキンの作成](#)」を参照してください。

この章では、各 UI コンポーネントのスキンを変更する方法について説明します。これらのいずれかの手順に従って UI コンポーネントのスキンを変更すると、ドキュメント内のすべてのインスタンスのスキンが変更されます。

一部のインスタンスのスキンの作成

次の一般的な手順を使用すると、コンポーネントの一部のインスタンスにスキンを作成できます。

- ライブラリパネルのコンポーネントの Assets フォルダーでスキンを選択します。
- スキンを複製し、一意のクラス名を割り当てます。
- スキンを編集して目的の外観を設定します。
- コンポーネントインスタンスに対して `setStyle()` メソッドを呼び出し、新しいスキンをスキンスタイルに割り当てます。

次の手順では、2 つの Button インスタンスのいずれかに対して、新しい `selectedDownSkin` を作成します。

- 1 新しい Flash ファイル (ActionScript 3.0) ドキュメントを作成します。

- 2 コンポーネントパネルから 2 つの Button をステージまでドラッグし、インスタンス名を **aButton** および **bButton** にします。
- 3 ライブラリパネルを開きます。次に、このパネル内の Component Assets フォルダーと ButtonSkins フォルダーを順に開きます。
- 4 selectedDownSkin スキンをクリックして選択します。
- 5 右クリックしてショートカットメニューを表示し、「複製」を選択します。
- 6 シンボルの複製ダイアログボックスで、新しいスキンに **Button_mySelectedDownSkin** などの一意の名前を付けます。次に「OK」をクリックします。
- 7 ライブラリパネルの Component Assets - ButtonSkins フォルダーで Button_mySelectedDownSkin をクリックし、右クリックしてコンテキストメニューを開きます。「リンケージ」を選択して、リンケージプロパティダイアログボックスを開きます。
- 8 「ActionScript に書き出し」チェックボックスをオンにします。「最初のフレームに書き出し」チェックボックスをオンのままにし、クラス名が一意であることを確認します。「OK」をクリックします。次に、クラス定義が見つからなかったので作成することを示す警告に対して、もう一度「OK」をクリックします。
- 9 ライブラリパネルで Button_mySelectedDownSkin スキンをダブルクリックし、シンボル編集モードで開きます。
- 10 プロパティインスペクターの塗りカラーピッカーにカラーが表示されるまで、スキンの中央の青い塗りをクリックします。カラーピッカーをクリックし、スキンの塗りのカラー #00CC00 を選択します。
- 11 ステージ上の編集バーの左側にある「戻る」ボタンをクリックし、ドキュメント編集モードに戻ります。
- 12 プロパティインスペクターで、各ボタンの「パラメーター」タブをクリックし、toggle パラメーターを true に設定します。
- 13 タイムラインのフレーム 1 のアクションパネルに次のコードを追加します。

```
bButton.setStyle("selectedDownSkin", Button_mySelectedDownSkin);  
bButton.setStyle("downSkin", Button_mySelectedDownSkin);
```
- 14 制御/ムービープレビューを選択します。
- 15 各ボタンをクリックします。bButton オブジェクトのダウン状態のスキン（選択および非選択）では、新しいスキンシンボルが使用されます。

Button コンポーネントのカスタマイズ

Button コンポーネントは、オーサリング時および実行時に、縦横に変形できます。オーサリング時には、ステージ上でコンポーネントを選択して、自由変形ツールを使用するか、修正/変形でいずれかのコマンドを使用します。実行時には、setSize() メソッドを使用するか、Button クラスの任意の適切なプロパティ（height および width、scaleX および scaleY など）を使用します。

ボタンのサイズを変更しても、アイコンやラベルのサイズは変化しません。Button の境界ボックスは Button の境界線に対応し、インスタンスのヒット領域を指定します。インスタンスのサイズを拡大すると、ヒット領域も広くなります。境界ボックスが小さすぎてラベル全体が収まらない場合は、境界ボックスに合わせてラベルが切り詰められます。

Button にアイコンがあり、そのアイコンが Button より大きい場合、アイコンは Button の境界線からはみ出します。

Button コンポーネントでのスタイルの使用

様々な状態でコンポーネントが描画される時、通常は Button のスタイルにより、そのスキン、アイコン、テキストフォーマットおよびパディングの値が指定されます。

次の手順では、2つの Button をステージに配置し、ユーザーがそのいずれかをクリックしたときに、両方の Button の emphasized プロパティを true に設定します。また、ユーザーがクリックしたとき 2 番目の Button の emphasizedSkin スタイルを selectedOverSkin スタイルに設定し、2つの Button が同じ状態の異なるスキンを表示するようにします。

- 1 Flash ファイル (ActionScript 3.0) を作成します。
- 2 2つの Button を 1 つずつステージまでドラッグし、インスタンス名を **aBtn** および **bBtn** にします。プロパティインスペクターの「パラメーター」タブで、Button A および Button B というラベルを付けます。
- 3 タイムラインのフレーム 1 のアクションパネルに次のコードを追加します。

```
bBtn.emphasized = true;
aBtn.emphasized = true;
bBtn.addEventListener(MouseEvent.CLICK, Btn_handler);
function Btn_handler(evt:MouseEvent):void {
    bBtn.setStyle("emphasizedSkin", "Button_selectedOverSkin");
}
```

- 4 制御/ムービープレビューを選択します。
- 5 いずれかのボタンをクリックし、各ボタンでの emphasizedSkin スタイルの効果を確認します。

Button コンポーネントでのスキンの使用

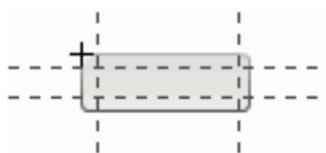
Button コンポーネントは、さまざまな状態に対応する次のスキンを使用します。スキンを編集して Button の外観を変更するには、次の図に示すように、ステージ上の Button インスタンスをダブルクリックして、スキンのパレットを開きます。



Button のスキン

ボタンが有効状態のとき、マウスポインターがボタン上に移動すると、そのボタンはオーバー状態で表示されます。クリックすると、ボタンは入力フォーカスを取得し、ダウン状態で表示されます。ユーザーがマウスボタンを離すと、ボタンはオーバー状態に戻ります。ボタン上でマウスボタンを押し下げ、そのままの状態でもマウスポインターをボタンの範囲外に移動すると、ボタンは最初の状態に戻ります。toggle パラメーターが true に設定されている場合、ダウン状態は selectedDownSkin、アップ状態は selectedUpSkin、オーバー状態は selectedOverSkin によってそれぞれ表示されます。無効状態の Button は、ユーザーの操作に関係なく無効状態の外観で表示されます。

いずれかのスキンを編集するには、次の図に示すように、そのスキンをダブルクリックしてシンボル編集モードで開きます。



シンボル編集モードの Button

このときに、Flash オーサリングツールを使用してスキンを目的どおりに編集できます。

次の手順では、Button の `selected_over` スキンのカラーを変更します。

- 1 新しい Flash ファイル (ActionScript 3.0) を作成します。
- 2 Button をコンポーネントパネルからステージまでドラッグします。「パラメーター」タブで、`toggle` パラメーターを `true` に設定します。
- 3 Button をダブルクリックして、スキンのパレットを開きます。
- 4 `selected_over` スキンをダブルクリックして、シンボル編集モードで開きます。
- 5 ズームコントロールを 400% に設定して、編集のためにアイコンを大きくします。
- 6 プロパティインスペクターの塗りカラーピッカーにカラーが表示されるまで、背景をダブルクリックします。
- 7 塗りカラーピッカーでカラー #CC0099 を選択し、`selected_over` スキンの背景に適用します。
- 8 ステージ上の編集バーの左側にある「戻る」ボタンをクリックし、ドキュメント編集モードに戻ります。
- 9 制御/ムービープレビューを選択します。
- 10 ボタンをクリックして選択状態にします。

Button の上にマウスポインターを移動すると、次の図に示すように `selected_over` 状態が表示されます。



カラーを変更した `selected_over` スキンを示す Button

CheckBox コンポーネントのカスタマイズ

CheckBox コンポーネントは、オーサリング時および実行時に、縦横に変形できます。オーサリング時には、ステージ上でコンポーネントを選択して、自由変形ツールを使用するか、修正/変形でいずれかのコマンドを使用します。実行時には、`setSize()` メソッドを使用するか、CheckBox クラスの適切なプロパティを使用します。例えば、CheckBox のサイズを変更するには、その `height` プロパティと `width` プロパティ、および `scaleX` プロパティと `scaleY` プロパティを設定します。CheckBox のサイズを変更しても、ラベルやチェックボックスアイコンのサイズは変化しません。境界ボックスのサイズが変化するだけです。

CheckBox インスタンスの境界ボックスは不可視ですが、そのインスタンスのヒット領域を表しています。インスタンスのサイズを拡大すると、ヒット領域も広くなります。境界ボックスが小さすぎてラベル全体が収まらない場合は、境界ボックスに合わせてラベルが切り詰められます。

CheckBox でのスタイルの使用

一連のスタイルプロパティを設定して、CheckBox インスタンスの外観を変更できます。例えば、次の手順では CheckBox ラベルのサイズとカラーを変更します。

- 1 CheckBox コンポーネントをコンポーネントパネルからステージまでドラッグし、インスタンス名を `myCb` にします。

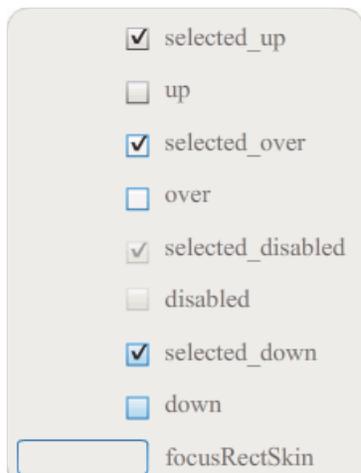
- 2 プロパティインスペクターの「パラメーター」タブをクリックし、label パラメーターの値として「\$500 以下」を入力します。
- 3 メインタイムラインのフレーム 1 で、アクションパネルに次のコードを入力します。

```
var myTf:TextFormat = new TextFormat();  
myCb.setSize(150, 22);  
myTf.size = 16;  
myTf.color = 0xFF0000;  
myCb.setStyle("textFormat", myTf);
```

詳しくは、93 ページの「[スタイルの設定](#)」を参照してください。コンポーネントのアイコンおよびスキンを変更するためのスタイルのプロパティの設定について詳しくは、97 ページの「[スキンの新規作成](#)」および 101 ページの「[CheckBox でのスキンの使用](#)」を参照してください。

CheckBox でのスキンの使用

CheckBox コンポーネントには次のスキンがあり、これらを編集して外観を変更することができます。



CheckBox のスキン

この例では、コンポーネントの up 状態および selectedUp 状態のアウトラインカラーと背景色を変更します。他の状態のスキンを変更する場合も、同様の手順に従います。

- 1 新しい Flash ファイル (ActionScript 3.0) ドキュメントを作成します。
- 2 CheckBox コンポーネントをステージまでドラッグします。コンポーネントは、そのアセットのフォルダーと共に、ライブラリにも配置されます。
- 3 CheckBox コンポーネントをステージ上でダブルクリックして、スキンアイコンのパネルを開きます。
- 4 selected_up アイコンをダブルクリックして、シンボル編集モードで開きます。
- 5 ズームコントロールを 800% に設定して、編集のためにアイコンを大きくします。
- 6 CheckBox の境界線をクリックして選択します。プロパティインスペクターの塗りカラーピッカーでカラー #0033FF を選択して、境界線に適用します。
- 7 CheckBox の背景をダブルクリックして選択し、再び塗りカラーピッカーで背景色を #00CCFF に設定します。
- 8 CheckBox のアップ状態スキンに対して手順 4 ~ 8 を繰り返します。
- 9 制御/ムービープレビューを選択します。

ColorPicker コンポーネントのカスタマイズ

ColorPicker に対するサイズの変更は、swatchWidth、swatchHeight、backgroundPadding、textFieldWidth および textFieldHeight の各スタイルのみを使用して行うことができます。自由変形ツールを使用したり、ActionScript で setSize() メソッドまたは width、height、scaleX、scaleY の各プロパティを使用したりして ColorPicker のサイズを変更しようとすると、それらの値は SWF ファイルの作成時に無視され、ColorPicker はデフォルトのサイズで表示されます。パレットの背景は、columnCount スタイルの setStyle() を使用して設定された列数に合わせてサイズが変更されます。デフォルトの列数は 18 です。カスタムカラーを 1024 に設定すると、色見本の数に合わせて垂直方向にパレットのサイズが変更されます。

ColorPicker コンポーネントでのスタイルの使用

ColorPicker コンポーネントの外観を変更するには、複数のスタイルを設定できます。例えば、次の手順では ColorPicker の列数 (columnCount) を 12 に変更し、色見本の高さ (swatchHeight) と幅 (swatchWidth) を変更し、両方のテキストフィールドのパディング (textPadding) と背景 (backgroundPadding) を変更します。

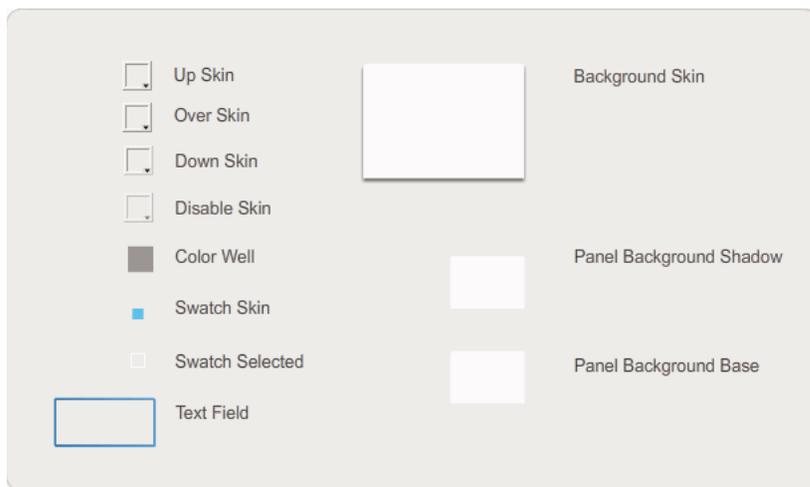
- 1 新しい Flash ファイル (ActionScript 3.0) ドキュメントを作成します。
- 2 ColorPicker コンポーネントをステージまでドラッグし、インスタンス名を **aCp** にします。
- 3 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次のコードを入力します。

```
aCp.setStyle("columnCount", 12);  
aCp.setStyle("swatchWidth", 8);  
aCp.setStyle("swatchHeight", 12);  
aCp.setStyle("swatchPadding", 2);  
aCp.setStyle("backgroundPadding", 3);  
aCp.setStyle("textPadding", 7);
```

- 4 制御/ムービープレビューを選択します。
- 5 ColorPicker をクリックして開き、設定によって外観がどのように変更されたのかを確認します。

ColorPicker コンポーネントでのスキンの使用

ColorPicker コンポーネントでは、次のスキンを使用して表示状態を表します。



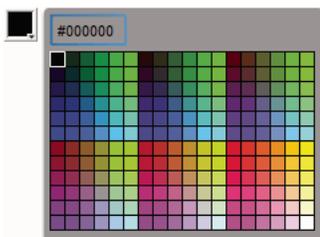
ColorPicker のスキン

Background スキンのカラーを変更すると、パレットの背景のカラーを変更できます。

- 1 新しい Flash ファイル (ActionScript 3.0) ドキュメントを作成します。

- 2 ColorPicker コンポーネントをステージにドラッグします。
- 3 コンポーネントをダブルクリックしてスキンのパレットを開きます。
- 4 スキンが選択されてプロパティインスペクターに塗りカラーピッカーが表示されるまで、Background スキンをダブルクリックします。
- 5 塗りカラーピッカーを使用してカラー #999999 を選択し、Background スキンに適用します。
- 6 ステージの上の編集バーの左側にある「戻る」ボタンをクリックし、ドキュメント編集モードに戻ります。
- 7 制御／ムービープレビューを選択します。

ColorPicker 上をクリックすると、次の図に示すようにパレットの背景はグレーになります。



濃いグレーの Background スキンが適用された ColorPicker

ComboBox コンポーネントのカスタマイズ

ComboBox コンポーネントは、オーサリング時および実行時に、縦横に変形できます。オーサリング時には、ステージ上でコンポーネントを選択して、自由変形ツールを使用するか、修正／変形でいずれかのコマンドを使用します。実行時には、setSize() メソッドを使用するか、ComboBox クラスの適切なプロパティ (height、width、scaleX、scaleY など) を使用します。

指定した幅と高さに合わせて ComboBox のサイズが変更されます。dropdownWidth プロパティが設定されていない限り、コンポーネントの幅に合わせてリストのサイズが変更されます。

テキストが長すぎて ComboBox に収まらない場合は、サイズに合わせてテキストが切り詰められます。ComboBox のサイズを変更し、テキストに合わせて dropdownWidth プロパティを設定する必要があります。

ComboBox コンポーネントでのスタイルの使用

一連のスタイルプロパティを設定して、ComboBox コンポーネントの外観を変更できます。スタイルにより、コンポーネントのスキン、セルレンダラー、パディングおよびボタンの幅の値が指定されます。次の例では、buttonWidth および textPadding のスタイルを設定します。buttonWidth スタイルは、ボタンのヒット領域の幅を設定し、ComboBox が編集可能ときに有効になります。ボタンを押すことでドロップダウンリストが表示されます。textPadding スタイルは、テキストフィールドの外側の境界線とテキストの間のスペースの量を指定します。ComboBox の高さを大きくする場合は、テキストフィールド内でテキストを垂直方向に中央揃えにすると便利です。この操作を行わないと、テキストはテキストフィールドの上部に表示されることがあります。

- 1 新しい Flash ファイル (ActionScript 3.0) ドキュメントを作成します。
- 2 ComboBox コンポーネントをステージまでドラッグし、インスタンス名を aCb にします。
- 3 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次のコードを入力します。

```
import fl.data.DataProvider;

aCb.setSize(150, 35);
aCb.setStyle("textPadding", 10);
aCb.setStyle("buttonWidth", 10);
aCb.editable = true;

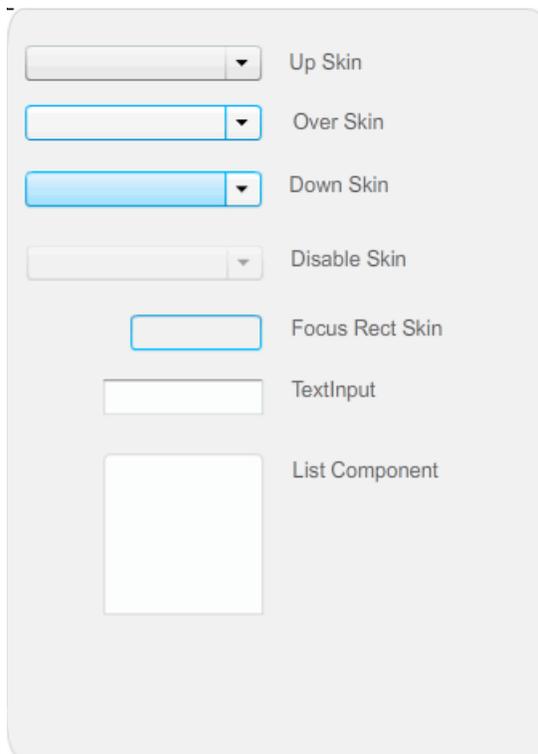
var items:Array = [
    {label:"San Francisco", data:"601 Townsend St."},
    {label:"San Jose", data:"345 Park Ave."},
    {label:"San Diego", data:"10590 West Ocean Air Drive, Suite 100"},
    {label:"Santa Rosa", data:"2235 Mercury Way, Suite 105"},
    {label:"San Luis Obispo", data:"3220 South Higuera Street, Suite 311"}
];
aCb.dataProvider = new DataProvider(items);
```

4 制御/ムービープレビューを選択します。

ドロップダウンリストを開くためにクリックするボタンの領域は、右側の狭い領域です。また、テキストはテキストフィールドに垂直方向に中央揃えで配置されます。2つの `setStyle()` ステートメントを含めずに例を実行すると、その効果を確認できます。

ComboBox でのスキンの使用

ComboBox では、次のスキンを使用して表示状態を表します。



ComboBox のスキン

Up スキンのカラーを変更すると、ステージ上の非アクティブ状態のコンポーネントのカラーを変更できます。

- 1 新しい Flash ファイル (ActionScript 3.0) ドキュメントを作成します。
- 2 ComboBox コンポーネントをステージにドラッグします。

- 3 コンポーネントをダブルクリックしてスキンのパレットを開きます。
- 4 Up スキンが選択されて編集できるように開くまで、Up スキンをダブルクリックします。
- 5 ズームコントロールを 400% に設定します。
- 6 プロパティインスペクターの塗りカラーピッカーにカラーが表示されるまで、スキンの中央の領域をクリックします。
- 7 塗りカラーピッカーを使用してカラー #33FF99 を選択し、Up スキンに適用します。
- 8 ステージ上の編集バーの左側にある「戻る」ボタンをクリックし、ドキュメント編集モードに戻ります。
- 9 制御/ムービープレビューを選択します。

次の図に示すように、ComboBox がステージに表示されます。



カスタムカラーの Background スキンが適用された ComboBox

DataGrid コンポーネントのカスタマイズ

DataGrid コンポーネントは、オーサリング時および実行時に、縦横に変形できます。オーサリング時には、ステージ上でコンポーネントを選択して、自由変形ツールを使用するか、修正/変形でいずれかのコマンドを使用します。実行時には、setSize() メソッドを使用するか、適切なプロパティ (width、height、scaleX、scaleY など) を使用します。横方向のスクロールバーがない場合は、列の幅が相対的に調整されます。列 (したがってセル) のサイズが変わると、そのセル内のテキストが切り詰められる可能性があります。

DataGrid コンポーネントでのスタイルの使用

一連のスタイルプロパティを設定して、DataGrid コンポーネントの外観を変更できます。DataGrid コンポーネントは、List コンポーネントからスタイルを継承します (110 ページの「[List コンポーネントでのスタイルの使用](#)」を参照してください)。

各列のスタイルの設定

DataGrid オブジェクトは複数の列を持つことができ、ユーザーは各列に別々のセルレンダラーを指定できます。DataGrid の各列は DataGridColumn オブジェクトによって表され、DataGridColumn クラスは cellRenderer プロパティを含みます。このプロパティに、列の CellRenderer を定義できます。

- 1 新しい Flash ドキュメント (ActionScript 3.0) を作成します。
- 2 DataGrid コンポーネントをライブラリパネルにドラッグします。
- 3 タイムラインのフレーム 1 のアクションパネルに次のコードを追加します。このコードは、3 番目の列に長いテキストのストリングのある DataGrid を作成します。最後に、列の cellRenderer プロパティを、複数行のセルをレンダリングするセルレンダラーの名前に設定します。

```
/* This is a simple cell renderer example. It invokes
the MultiLineCell cell renderer to display a multiple
line text field in one of a DataGrid's columns. */

import fl.controls.DataGrid;
import fl.controls.dataGridClasses.DataGridColumn;
import fl.data.DataProvider;
import fl.controls.ScrollPolicy;

// Create a new DataGrid component instance.
var aDg:DataGrid = new DataGrid();

var aLongString:String = "An example of a cell renderer class that displays a multiple line TextField"
var myDP:Array = new Array();
myDP = [{firstName:"Winston", lastName:"Elstad", note:aLongString, item:100},
        {firstName:"Ric", lastName:"Dietrich", note:aLongString, item:101},
        {firstName:"Ewing", lastName:"Canepa", note:aLongString, item:102},
        {firstName:"Kevin", lastName:"Wade", note:aLongString, item:103},
        {firstName:"Kimberly", lastName:"Dietrich", note:aLongString, item:104},
        {firstName:"AJ", lastName:"Bilow", note:aLongString, item:105},
        {firstName:"Chuck", lastName:"Yushan", note:aLongString, item:106},
        {firstName:"John", lastName:"Roo", note:aLongString, item:107},
    ];

// Assign the data provider to the DataGrid to populate it.
// Note: This has to be done before applying the cellRenderers.
aDg.dataProvider = new DataProvider(myDP);

/* Set some basic grid properties.
Note: The data grid's row height should reflect
the number of lines you expect to show in the multiline cell.
The cell renderer will size to the row height.
About 40 for 2 lines or 60 for 3 lines.*/

aDg.columns = ["firstName", "lastName", "note", "item"];
aDg.setSize(430,190);
aDg.move(40,40);
aDg.rowHeight = 40;// Allows for 2 lines of text at default text size.
aDg.columns[0].width = 70;
aDg.columns[1].width = 70;
aDg.columns[2].width = 230;
aDg.columns[3].width = 60;
aDg.resizableColumns = true;
aDg.verticalScrollPolicy = ScrollPolicy.AUTO;
addChild(aDg);
// Assign cellRenderers.
var col3:DataGridColumn = new DataGridColumn();
col3 = aDg.getColumnAt(2);
col3.cellRenderer = MultiLineCell;
```

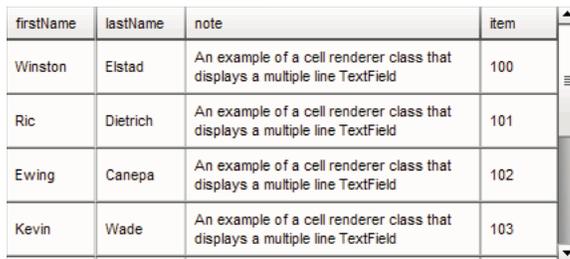
- 4 FLA ファイルに "MultiLineGrid.fla" という名前を付けて保存します。
- 5 ActionScript ファイルを新規作成します。
- 6 次の ActionScript コードをスクリプトウィンドウにコピーします。

```
package {  
  
    import fl.controls.listClasses.CellRenderer;  
  
    public class MultiLineCell extends CellRenderer  
    {  
  
        public function MultiLineCell()  
        {  
            textField.wordWrap = true;  
            textField.autoSize = "left";  
        }  
        override protected function drawLayout():void {  
            textField.width = this.width;  
            super.drawLayout();  
        }  
    }  
}
```

7 "MultiLineGrid.fla" を保存したフォルダーと同じフォルダーに、ActionScript ファイルを "MultiLineCell.as" という名前で保存します。

8 MultiLineGrid.fla アプリケーションに戻り、制御/ムービープレビューを選択します。

DataGrid は、次のように表示されます。



firstName	lastName	note	item
Winston	Elstad	An example of a cell renderer class that displays a multiple line TextField	100
Ric	Dietrich	An example of a cell renderer class that displays a multiple line TextField	101
Ewing	Canepa	An example of a cell renderer class that displays a multiple line TextField	102
Kevin	Wade	An example of a cell renderer class that displays a multiple line TextField	103

MultiLineGrid.fla アプリケーションの DataGrid

ヘッダースタイルの設定

ヘッダースタイルを設定するには、headerTextFormat スタイルを使用します。次の例では、TextFormat オブジェクトを使用して、Arial フォント、赤のカラー、フォントサイズ 14 およびイタリックを使用するように headerTextFormat スタイルを設定します。

- 1 新しい Flash ファイル (ActionScript 3.0) ドキュメントを作成します。
- 2 DataGrid コンポーネントをステージまでドラッグし、インスタンス名を **adg** にします。
- 3 アクションパネルを開いて、メインタイムライン内でフレーム 1 を選択し、次のコードを入力します。

```
import fl.data.DataProvider;
import fl.controls.dataGridClasses.DataGridColumn;

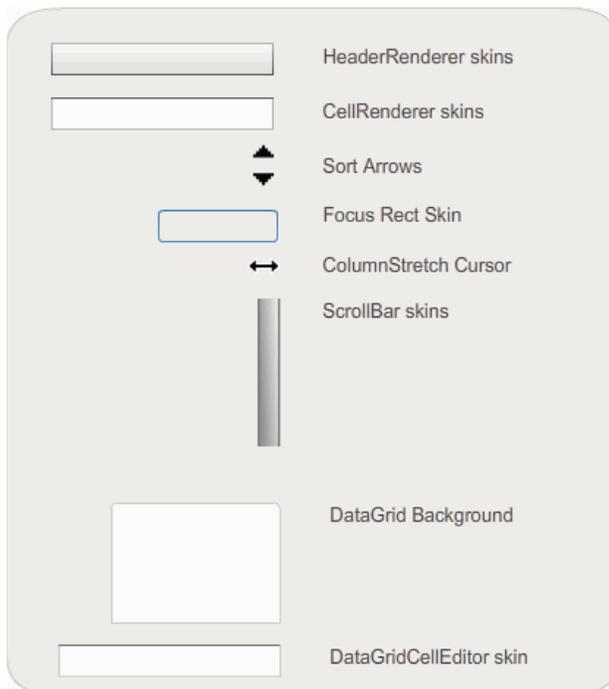
var myDP:Array = new Array();
myDP = [{FirstName:"Winston", LastName:"Elstad"},
        {FirstName:"Ric", LastName:"Dietrich"},
        {FirstName:"Ewing", LastName:"Canepa"},
        {FirstName:"Kevin", LastName:"Wade"},
        {FirstName:"Kimberly", LastName:"Dietrich"},
        {FirstName:"AJ", LastName:"Bilow"},
        {FirstName:"Chuck", LastName:"Yushan"},
        {FirstName:"John", LastName:"Roo"},
    ];

// Assign the data provider to the DataGrid to populate it.
// Note: This has to be done before applying the cellRenderers.
aDg.dataProvider = new DataProvider(myDP);
aDg.setSize(160,190);
aDg.move(40,40);
aDg.columns[0].width = 80;
aDg.columns[1].width = 80;
var tf:TextFormat = new TextFormat();
tf.size = 14;
tf.color = 0xff0000;
tf.italic = true;
tf.font = "Arial"
aDg.setStyle("headerTextFormat", tf);
```

- 4 制御/ムービープレビューを選択して、アプリケーションを実行します。

DataGrid コンポーネントでのスキンの使用

DataGrid コンポーネントでは、次のスキンを使用して表示状態を表します。



DataGrid のスキン

CellRenderer スキンは DataGrid の本文のセルに使用され、HeaderRenderer スキンはヘッダー行に使用されます。次の手順では、ヘッダー行の背景色を変更しますが、同じ手順を使用して CellRenderer スキンを編集し、DataGrid の本文のセルの背景色を変更できます。

- 1 新しい Flash ドキュメント (ActionScript 3.0) を作成します。
- 2 DataGrid コンポーネントをステージまでドラッグし、インスタンス名を **aDg** にします。
- 3 コンポーネントをダブルクリックしてスキンのパレットを開きます。
- 4 ズームコントロールを 400% にして、編集のためにアイコンを大きくします。
- 5 HeaderRenderer スキンをダブルクリックして、HeaderRenderer スキンのパレットを開きます。
- 6 Up_Skin をダブルクリックしてシンボル編集モードで開きます。次に、背景が選択されてプロパティインスペクターの塗りカラーピッカーが表示されるまで、背景をクリックします。
- 7 塗りカラーピッカーでカラー #00CC00 を選択し、Up_Skin HeaderRenderer スキンの背景に適用します。
- 8 ステージの上の編集バーの左側にある「戻る」ボタンをクリックし、ドキュメント編集モードに戻ります。
- 9 タイムラインのフレーム 1 のアクションパネルに次のコードを追加し、DataGrid にデータを追加します。

```
import fl.data.DataProvider;

bldRosterGrid(aDg);
var aRoster:Array = new Array();
aRoster = [
    {Name:"Wilma Carter",Home: "Redlands, CA"},
    {Name:"Sue Pennypacker",Home: "Athens, GA"},
    {Name:"Jill Smithfield",Home: "Spokane, WA"},
    {Name:"Shirley Goth", Home: "Carson, NV"},
    {Name:"Jennifer Dunbar",Home: "Seaside, CA"}
];
aDg.dataProvider = new DataProvider(aRoster);
function bldRosterGrid(dg:DataGrid){
    dg.setSize(400, 130);
    dg.columns = ["Name", "Home"];
    dg.move(50,50);
    dg.columns[0].width = 120;
    dg.columns[1].width = 120;
};
```

- 10 制御/ムービープレビューを選択してアプリケーションをテストします。
DataGrid によって、次の図のようにヘッダー行の背景が緑で表示されます。

Name	Home
Wilma Carter	Redlands, CA
Sue Pennypacker	Athens, GA
Jill Smithfield	Spokane, WA
Shirley Goth	Carson, NV
Jennifer Dunbar	Seaside, CA

ヘッダー行の背景がカスタマイズされた DataGrid

Label コンポーネントのカスタマイズ

Label コンポーネントは、オーサリング時および実行時に、縦横に変形できます。オーサリング時には、ステージ上でコンポーネントを選択して、自由変形ツールを使用するか、修正/変形でいずれかのコマンドを使用します。また、autoSize オーサリングパラメーターを設定することもできます。このパラメーターを設定してもライブプレビューに表示される境界

ボックスは変化しませんが、Label のサイズが調整されます。Label のサイズの調整は、wordwrap パラメーターに応じて行われます。パラメーターが true の場合、Label のサイズはテキストに合わせて垂直方向に調整されます。パラメーターが false の場合、Label のサイズは垂直方向に調整されます。実行時には、setSize() メソッドを使用します。詳しくは、『[ActionScript 3.0 Reference for the Adobe Flash Platform](#)』の Label.setSize() メソッドおよび Label.autoSize プロパティを参照してください。59 ページの「[Label コンポーネントを使用したアプリケーションの作成](#)」も参照してください。

Label コンポーネントでのスタイルの使用

一連のスタイルプロパティを設定して、Label インスタンスの外観を変更できます。1 つの Label コンポーネントインスタンス内では、テキスト全体に同じスタイルが適用されます。Label コンポーネントには textFormat スタイルがあります。このスタイルは TextFormat オブジェクトと同じ属性を持ち、通常の Flash TextField に対して行うように、Label.text のコンテンツに同じプロパティを設定できます。次の例では、ラベルのテキストのカラーを赤に設定します。

- 1 Label コンポーネントをコンポーネントパネルからステージまでドラッグし、インスタンス名を a_label にします。
- 2 「パラメーター」タブをクリックし、テキストプロパティの値を次のテキストに置き換えます。

Color me red

- 3 メインタイムライン内でフレーム 1 を選択し、アクションパネルを開いて、次のコードを入力します。

```
/* Create a new TextFormat object, which allows you to set multiple text properties at a time. */  
  
var tf:TextFormat = new TextFormat();  
tf.color = 0xFF0000;  
/* Apply this specific text format (red text) to the Label instance. */  
a_label.setStyle("textFormat", tf);
```

- 4 制御/ムービープレビューを選択します。

Label スタイルについて詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の Label クラスを参照してください。

スキンと Label

Label コンポーネントには、スキンを使用するビジュアルエレメントはありません。

List コンポーネントのカスタマイズ

List コンポーネントは、オーサリング時および実行時に、縦横に変形できます。オーサリング時には、ステージ上でコンポーネントを選択して、自由変形ツールを使用するか、修正/変形でいずれかのコマンドを使用します。実行時には、setSize() メソッドを使用するか、List クラスの適切なプロパティ (height、width、scaleX、scaleY など) を使用します。

リストのサイズを変更し、リストの行を水平方向に縮めると、行内のテキストの表示が切り詰められます。垂直方向のサイズを変更すると、必要に応じてリストの行数が増減します。スクロールバーは必要に応じて自動的に配置されます。

List コンポーネントでのスタイルの使用

一連のスタイルプロパティを設定して、List コンポーネントの外観を変更できます。スタイルは、コンポーネントが描画される時のスキンとバディングの値を指定します。

様々なスキんスタイルを使用することで、スキンに使用する様々なクラスを指定できます。スキんスタイルの使用について詳しくは、96 ページの「[スキンについて](#)」を参照してください。

次の手順では、List コンポーネントの `contentPadding` スタイルの値を設定します。この設定の値は、コンテンツの周囲のパディングを行うために List のサイズから減算されるので、List 内のテキストが切り詰めて表示されるのを防ぐには、List のサイズを大きくする必要がある場合があります。

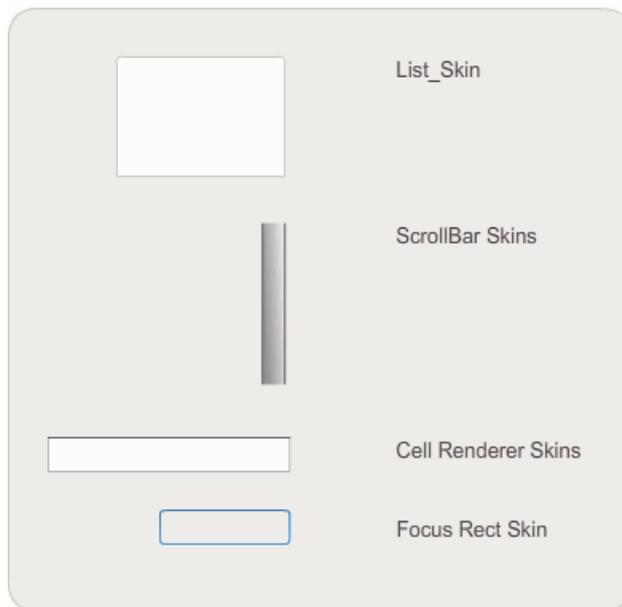
- 1 新しい Flash ファイル (ActionScript 3.0) ドキュメントを作成します。
- 2 List コンポーネントをコンポーネントパネルからステージまでドラッグし、インスタンス名を **aList** にします。
- 3 メインタイムフレーム内でフレーム 1 を選択し、アクションパネルを開いて、次のコードを入力します。このコードでは `contentPadding` スタイルを設定し、データを List に追加します。

```
aList.setStyle("contentPadding", 5);  
aList.setSize(145, 200);  
aList.addItem({label:"1956 Chevy (Cherry Red)", data:35000});  
aList.addItem({label:"1966 Mustang (Classic)", data:27000});  
aList.addItem({label:"1976 Volvo (Xc11nt Cond)", data:17000});  
aList.rowCount = aList.length;
```

- 4 制御/ムービープレビューを選択します。

List コンポーネントでのスキンの使用

List コンポーネントでは、次のスキンを使用して表示状態を表します。

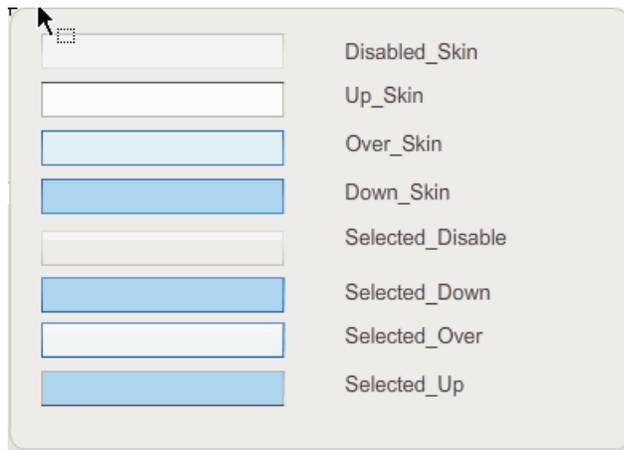


List のスキン

ScrollBar でのスキンの適用について詳しくは、125 ページの「[UIScrollBar コンポーネントのカスタマイズ](#)」を参照してください。Focus Rect スキンでのスキンの適用について詳しくは、120 ページの「[TextArea コンポーネントのカスタマイズ](#)」を参照してください。

注意：1 つのコンポーネントで ScrollBar スキンを変更すると、ScrollBar を使用するその他のすべてのコンポーネントでスキンが変更されます。

Cell Renderer スキンをダブルクリックして、List セルのさまざまな状態を示す 2 番目のスキンのパレットを開きます。



List の Cell Renderer のスキン

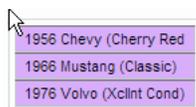
List のセルの外観は、これらのスキンを編集すると変更できます。次の手順では、Up スキンのカラーを変更し、通常の非アクティブ状態の List の外観を変更します。

- 1 新しい Flash ファイル (ActionScript 3.0) ドキュメントを作成します。
- 2 List コンポーネントをコンポーネントパネルからステージまでドラッグし、インスタンス名を **aList** にします。
- 3 List をダブルクリックしてスキンのパレットを開きます。
- 4 Cell Renderer スキンをダブルクリックして、そのスキンのパレットを開きます。
- 5 Up_Skin スキンをダブルクリックして、編集のために開きます。
- 6 スキンの塗り領域をクリックして選択します。プロパティインスペクターに塗りカラーピッカーが表示され、スキンの現在の塗りのカラーが表示されます。
- 7 塗りカラーピッカーでカラー #CC66FF を選択し、Up_Skin スキンの塗りに適用します。
- 8 ステージの上の編集バーの左側にある「戻る」ボタンをクリックし、ドキュメント編集モードに戻ります。
- 9 タイムラインのフレーム 1 のアクションパネルに次のコードを追加し、List にデータを追加します。

```
aList.setStyle("contentPadding", 5);  
aList.setSize(145, 200);  
aList.addItem({label:"1956 Chevy (Cherry Red)", data:35000});  
aList.addItem({label:"1966 Mustang (Classic)", data:27000});  
aList.addItem({label:"1976 Volvo (Xcllnt Cond)", data:17000});  
aList.rowCount = aList.length;
```

- 10 制御/ムービープレビューを選択します。

List の表示は次の図のようになります。



カスタムカラーの Up_Skin が適用された List セル

フレームは contentPadding スタイルの設定によって異なります。

NumericStepper コンポーネントのカスタマイズ

NumericStepper コンポーネントは、オーサリング時および実行時に、縦横に変形できます。オーサリング時には、ステージ上でコンポーネントを選択して、自由変形ツールを使用するか、修正／変形でいずれかのコマンドを使用します。実行時には、setSize() メソッドを使用するか、NumericStepper クラスの任意の適切なプロパティおよびメソッド (width、height、scaleX、scaleY など) を使用します。

NumericStepper コンポーネントのサイズを変更しても、上下の矢印ボタンの幅は変化しません。ステッパーの高さをデフォルトよりも大きくすると、デフォルトの動作により、矢印ボタンはコンポーネントの上端と下端に配置されます。それ以外の場合は、9 スライスの拡大 / 縮小によってボタンの描画方法が決定されます。矢印ボタンは常にテキストボックスの右側に表示されます。

スタイルと NumericStepper コンポーネント

NumericStepper コンポーネントのスタイルプロパティを設定すると、その外観を変更できます。スタイルは、コンポーネントが描画されるときにスキン、パディングおよびテキストフォーマットの値を指定します。textFormat スタイルを使用すると、NumericStepper の値のサイズと外観を変更できます。様々なスキンスタイルを使用することで、スキンに使用する様々なクラスを指定できます。スキンスタイルの使用について詳しくは、96 ページの「[スキンについて](#)」を参照してください。

この手順では、textFormat スタイルを使用して、NumericStepper で表示される値の外観を変更します。

- 1 新しい Flash ドキュメント (ActionScript 3.0) を作成します。
- 2 NumericStepper コンポーネントをコンポーネントパネルからステージまでドラッグし、インスタンス名を **myNs** にします。
- 3 メインタイムラインのフレーム 1 のアクションパネルに次のコードを追加します。

```
var tf:TextFormat = new TextFormat();
myNs.setSize(100, 50);
tf.color = 0x0000CC;
tf.size = 24;
tf.font = "Arial";
tf.align = "center";
myNs.setStyle("textFormat", tf);
```

- 4 制御／ムービープレビューを選択します。

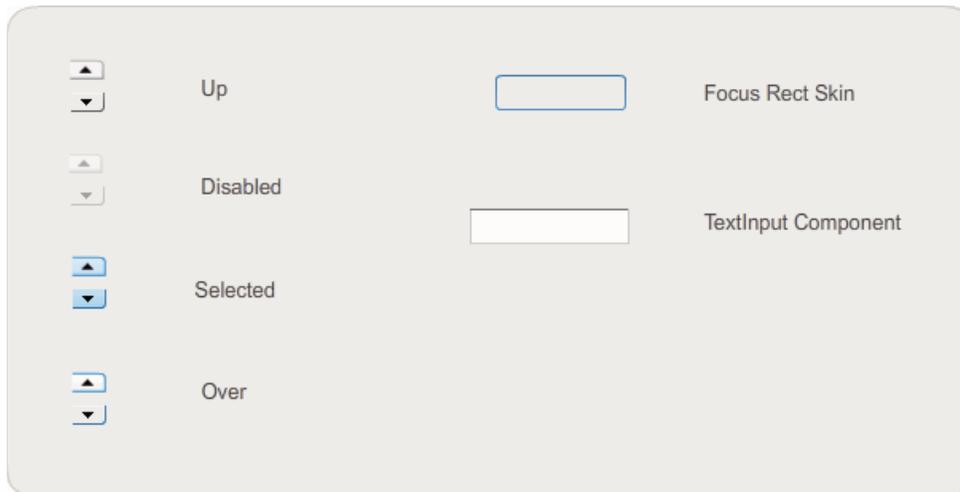
スキンと NumericStepper コンポーネント

NumericStepper コンポーネントには、ボタンのアップ、ダウン、無効、および選択の各状態を表すスキンがあります。

ステッパーが有効状態のとき、マウスポインターが上下の矢印ボタン上に移動すると、そのボタンはオーバー状態が表示されます。クリックすると、ボタンはダウン状態が表示されます。ユーザーがマウスボタンを離すと、ボタンはオーバー状態に戻ります。ボタン上でマウスボタンを押し下げ、そのままの状態でもマウスポインターをボタンの範囲外に移動すると、そのボタンは最初の状態に戻ります。

無効状態のステッパーは、ユーザーの操作に関係なく無効状態の外観で表示されます。

NumericStepper コンポーネントには次のスキンがあります。



NumericStepper のスキン

- 1 FLA ファイルを新規作成します。
- 2 NumericStepper コンポーネントをステージにドラッグします。
- 3 ズームコントロールを 400% にして、編集のためにイメージを大きくします。
- 4 「グループ」レベルまでドリルダウンし、プロパティインスペクターの塗りカラーピッカーに背景色が表示されるまで、スキンパネルの `TextInput` スキンの背景をダブルクリックします。
- 5 プロパティインスペクターの塗りカラーピッカーでカラー #9999FF を選択し、`TextInput` スキンの背景に適用します。
- 6 ステージの上の編集バーの左側にある「戻る」ボタンをクリックし、ドキュメント編集モードに戻ります。
- 7 再度 `NumericStepper` をダブルクリックしてスキンパネルをもう一度開きます。
- 8 背景が選択され、そのカラーがプロパティインスペクターの塗りカラーピッカーに表示されるまで、「Up」グループの上矢印ボタンの背景をクリックします。
- 9 カラー #9966FF を選択し、上矢印ボタンの背景に適用します。
- 10 「Up」グループの下矢印ボタンに対して手順 8 および 9 を繰り返します。
- 11 制御／ムービープレビューを選択します。

`NumericStepper` インスタンスは次の図のように表示されます。



ProgressBar コンポーネントのカスタマイズ

`ProgressBar` コンポーネントは、オーサリング時および実行時に、縦横に変形できます。オーサリング時には、ステージ上でコンポーネントを選択して、自由変形ツールを使用するか、修正／変形でいずれかのコマンドを使用します。実行時には、`setSize()` メソッドを使用するか、`ProgressBar` クラスの適切なプロパティ (`height`、`width`、`scaleX`、`scaleY` など) を使用します。

`ProgressBar` には、トラックスキン、バースキン、および無限スキンの 3 つのスキンがあります。9 スライスの拡大 / 縮小機能を使用してアセットが拡大または縮小されます。

スタイルと ProgressBar コンポーネント

一連のスタイルプロパティを設定して、ProgressBar インスタンスの外観を変更できます。ProgressBar のスタイルは、コンポーネントが描画されるときにスキンとパディングの値を指定します。次の例では、ProgressBar インスタンスのサイズを拡大し、その barPadding スタイルを設定します。

- 1 FLA ファイルを新規作成します。
- 2 ProgressBar コンポーネントをコンポーネントパネルからステージまでドラッグし、インスタンス名を **myPb** にします。
- 3 メインタイムラインのフレーム 1 で、アクションパネルに次のコードを入力します。

```
myPb.width = 300;  
myPb.height = 30;  
  
myPb.setStyle("barPadding", 3);
```

- 4 制御/ムービープレビューを選択します。

スキンスタイルの設定について詳しくは、96 ページの「[スキンについて](#)」を参照してください。

スキンと ProgressBar コンポーネント

次の図に示すように、ProgressBar コンポーネントはスキンを使用して、プログレスパートラック、完了したバー、および無限バーを表します。



ProgressBar のスキン

バーはトラックスキンに配置され、barPadding を使用して位置が決定されます。アセットは 9 スライスの拡大 / 縮小を使用して拡大または縮小されます。

無限バーは、ProgressBar インスタンスの indeterminate プロパティが true に設定されている場合に使用されます。スキンのサイズは、ProgressBar のサイズに合わせて垂直方向および水平方向に変更されます。

これらのスキンを編集して、ProgressBar の外観を変更できます。例えば、次の例では無限バーのカラーを変更します。

- 1 FLA ファイルを新規作成します。
- 2 ProgressBar コンポーネントをステージにドラッグし、ダブルクリックしてスキンアイコンのパネルを開きます。
- 3 無限バーのスキンをダブルクリックします。
- 4 ズームコントロールを 400% に設定して、編集のためにアイコンを大きくします。
- 5 対角線のバーの 1 つをダブルクリックし、Shift キーを押しながら他の各バーをクリックします。現在のカラーが、プロパティインスペクターの塗りカラーピッカーに表示されます。
- 6 プロパティインスペクターで塗りカラーピッカーをクリックして開き、カラー #00CC00 を選択して、選択した対角線のバーに適用します。
- 7 ステージの上の編集バーの左側にある「戻る」ボタンをクリックし、ドキュメント編集モードに戻ります。
- 8 制御/ムービープレビューを選択します。

ProgressBar は次の図のように表示されます。



RadioButton コンポーネントのカスタマイズ

RadioButton コンポーネントは、オーサリング時および実行時に、縦横に変形できます。オーサリング時には、ステージ上でコンポーネントを選択して、自由変形ツールを使用するか、修正／変形でいずれかのコマンドを使用します。実行時には、setSize() メソッドを使用します。

RadioButton コンポーネントの境界ボックスは不可視ですが、そのコンポーネントのヒット領域を表しています。コンポーネントのサイズを拡大すると、ヒット領域も広がります。

コンポーネントの境界ボックスが小さすぎてコンポーネントラベル全体が収まらない場合は、境界ボックスに合わせてラベルが切り詰められます。

RadioButton コンポーネントでのスタイルの使用

一連のスタイルプロパティを設定して、RadioButton の外観を変更できます。RadioButton のスタイルプロパティは、コンポーネントが描画される時のスキン、アイコン、テキストフォーマットおよびパディングの値を指定します。

RadioButton のスタイルは、コンポーネントが描画される時のレイアウトのスキンおよびパディングの値を指定します。

次の例では、CheckBox コンポーネントから textFormat スタイルを取得し、RadioButton に適用して、そのラベルのスタイルを同じにします。

- 1 新しい Flash ドキュメント (ActionScript 3.0) を作成します。
- 2 CheckBox コンポーネントをステージにドラッグし、プロパティインスペクターでインスタンス名を **myCh** にします。
- 3 RadioButton をステージにドラッグし、プロパティインスペクターでインスタンス名を **myRb** にします。
- 4 タイムラインのフレーム 1 のアクションパネルに次のコードを追加します。

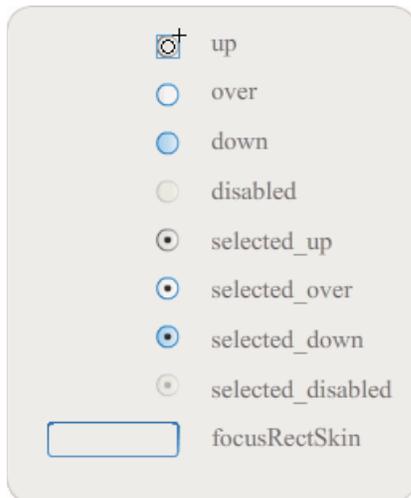
```
var tf:TextFormat = new TextFormat();  
tf.color = 0x00FF00;  
tf.font = "Georgia";  
tf.size = 18;  
myCh.setStyle("textFormat", tf);  
myRb.setStyle("textFormat", myCh.getStyle("textFormat"));
```

このコードは CheckBox に textFormat スタイルを設定し、CheckBox で getStyle() メソッドを呼び出して RadioButton に適用します。

- 5 制御／ムービープレビューを選択します。

スキンと RadioButton コンポーネント

RadioButton には次のスキンがあり、これらを編集して外観を変更することができます。



RadioButton のスキン

RadioButton が有効状態で選択されていない場合は、マウスポインターが RadioButton 上に移動すると、その RadioButton はオーバー状態のスキンで表示されます。クリックすると、RadioButton は入力フォーカスを取得し、selected_down スキンで表示されます。ユーザーがマウスボタンを離すと、selected_up スキンで表示されます。マウスボタンを押し下げながらマウスポインターを RadioButton のヒット領域外に移動すると、RadioButton はアップ状態のスキンで再表示されます。

無効状態の RadioButton は、ユーザーの操作に関係なく無効状態の外観で表示されます。

次の例では、選択状態を示す selected_up スキンを置き換えます。

- 1 新しい Flash ドキュメント (ActionScript 3.0) を作成します。
- 2 RadioButton コンポーネントをステージにドラッグし、ダブルクリックしてスキンのパレットを開きます。
- 3 ズームコントロールを 800% にして、編集のためにアイコンを大きくします。
- 4 selected_up スキンをダブルクリックして選択し、Delete キーを押して削除します。
- 5 ツールパネルから矩形ツールを選択します。
- 6 プロパティインスペクターで、線のカラーを赤 (#FF0000) に設定し、塗りのカラーを黒 (#000000) に設定します。
- 7 シンボルの基準点 (原点、ゼロ点) を示す十字カーソルから開始し、ポインターをクリックおよびドラッグして矩形を描画します。
- 8 ステージの上の編集バーの左側にある「戻る」ボタンをクリックし、ドキュメント編集モードに戻ります。
- 9 制御/ムービープレビューを選択します。
- 10 RadioButton をクリックして選択します。

選択状態の RadioButton は、次の図のように表示されます。



ScrollPane コンポーネントのカスタマイズ

ScrollPane コンポーネントは、オーサリング時および実行時に、縦横に変形できます。オーサリング時には、ステージ上でコンポーネントを選択して、自由変形ツールを使用するか、修正／変形でいずれかのコマンドを使用します。実行時には、setSize() メソッドを使用するか、ScrollPane クラスの任意の適切なプロパティまたはメソッド (height、width、scaleX、scaleY など) を使用します。

ScrollPane コンポーネントには、次のグラフィカル特性があります。

- コンテンツの基準点 (原点、ゼロ点) はペインの左上隅に表示されます。
- 水平スクロールバーがオフのときは、垂直スクロールバーはスクロールペインの右側に沿って上から下に表示されます。垂直スクロールバーがオフのときは、水平スクロールバーはスクロールペインの下部に沿って左から右に表示されます。両方のスクロールバーをオフにすることもできます。
- スクロールペインが小さすぎると、コンテンツが正しく表示されないことがあります。
- スクロールペインのサイズを変更すると、スクロールトラックおよびスクロールボックス (サム) が拡大または縮小し、ヒット領域のサイズが変更されます。ボタンのサイズは同じままです。

ScrollPane コンポーネントでのスタイルの使用

ScrollPane コンポーネントのスタイルプロパティにより、コンポーネントが描画されるときのスकिनの値およびレイアウトのパディングの値が指定されます。様々なスकिनスタイルを使用することで、コンポーネントのスकिनに使用する様々なクラスを指定できます。スकिनスタイルの使用について詳しくは、96 ページの「[スकिनについて](#)」を参照してください。

- 1 新しい Flash ドキュメント (ActionScript 3.0) を作成します。
- 2 ScrollPane コンポーネントをステージまでドラッグし、インスタンス名を **mySp** にします。
- 3 プロパティインスペクターの「パラメーター」タブをクリックし、source パラメーターに「<http://www.helpexamples.com/flash/images/image1.jpg>」と入力します。
- 4 メインタイムラインのフレーム 1 で、アクションパネルに次のコードを入力します。

```
mySp.setStyle("contentPadding", 5);
```

スクロールバーの外側で、コンポーネントの境界線とそのコンテンツの間にパディングが適用されます。

- 5 制御／ムービープレビューを選択します。

スकिनと ScrollPane

ScrollPane コンポーネントはスクロールアセットに対して境界線とスクロールバーを使用します。スクロールバーへのスकिनの適用について詳しくは、125 ページの「[UIScrollBar コンポーネントでのスकिनの使用](#)」を参照してください。

Slider コンポーネントのカスタマイズ

Slider コンポーネントは、オーサリング時および実行時に、横方向に変形できます。オーサリング時には、ステージ上でコンポーネントを選択して、自由変形ツールを使用するか、修正／変形でいずれかのコマンドを使用します。実行時には、setSize() メソッドを使用するか、Slider クラスの任意の適切なプロパティ (width、scaleX など) を使用します。

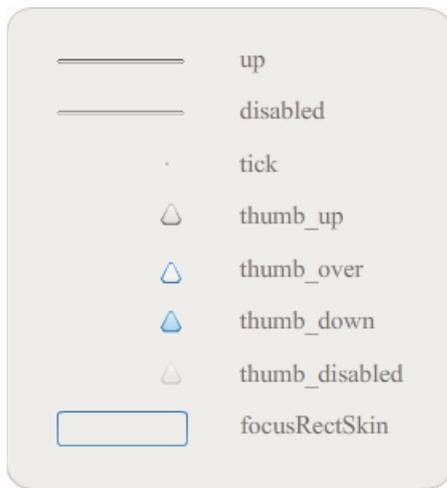
スライダーは、長くすることはできますが、高さを高くすることはできません。Flash では、setSize() メソッドの height プロパティおよび height パラメーターは無視されます。ただし、垂直方向のスライダーを作成し、垂直方向に長くすることはできます。

スタイルと Slider コンポーネント

Slider コンポーネントのスタイルは、そのスキンのクラスと、コンポーネントの境界ボックスと外側の境界の間のパディングに使用するピクセル数を指定する `FocusRectPadding` の値のみを指定します。スキンスタイルの使用については、96 ページの「[スキンについて](#)」を参照してください。

スキンと Slider コンポーネント

Slider コンポーネントでは次のスキンを使用します。これらのスキンは編集して外観を変更することができます。



Slider のスキン

次の例では、アップトラックを編集してそのカラーを青に変更します。

- 1 新しい Flash ドキュメント (ActionScript 3.0) を作成します。
- 2 Slider コンポーネントをコンポーネントパネルからステージまでドラッグします。
- 3 Slider コンポーネントをダブルクリックしてスキンパネルを開きます。
- 4 アップトラックを基準点でダブルクリックし、シンボル編集モードで開きます。
- 5 ズームコントロールを 800% に設定して、編集のためにアイコンを大きくします。Slider のトラックには 3 つのバーがあります。
- 6 一番上のバーをクリックして選択します。バーを選択すると、そのカラーがプロパティインスペクターの塗りカラーピッカーに表示されます。
- 7 プロパティインスペクターの塗りカラーピッカーを使用し、カラー #000066 を選択して Slider トラックの一番上のバーに適用します。
- 8 Slider トラックの中央部分のバーをクリックして選択します。バーを選択すると、そのカラーがプロパティインスペクターの塗りカラーピッカーに表示されます。
- 9 プロパティインスペクターの塗りカラーピッカーを使用し、カラー #0066FF を選択して Slider トラックの中央のバーに適用します。
- 10 Slider トラックの一番下のバーをクリックして選択します。バーを選択すると、そのカラーがプロパティインスペクターの塗りカラーピッカーに表示されます。
- 11 プロパティインスペクターの塗りカラーピッカーを使用し、カラー #00CCFF を選択して Slider トラックの一番下のバーに適用します。
- 12 ステージの上の編集バーの左側にある「戻る」ボタンをクリックし、ドキュメント編集モードに戻ります。

13 制御/ムービープレビューを選択します。

Slider の表示は次の図のようになります。



TextArea コンポーネントのカスタマイズ

TextArea コンポーネントは、オーサリング時および実行時に、縦横に変形できます。オーサリング時には、ステージ上でコンポーネントを選択して、自由変形ツールを使用するか、修正/変形でいずれかのコマンドを使用します。実行時には、setSize() メソッドを使用するか、TextArea クラスの任意の適切なプロパティ (height、width、scaleX、scaleY など) を使用します。

TextArea コンポーネントのサイズを変更すると、境界線が新しい境界ボックスのサイズに合わせて変更されます。スクロールバーが必要な場合は、下端と右端に配置されます。その後、残りの領域内でテキストエリアのサイズが変更されます。TextArea コンポーネントには、固定サイズの要素はありません。TextArea コンポーネントの幅が、テキストのサイズを表示するのに狭すぎる場合、テキストは切り詰められます。

スタイルと TextArea コンポーネント

TextArea コンポーネントのスタイルは、コンポーネントが描画される時のスキン、パディングおよびテキストフォーマットの値を指定します。textFormat スタイルおよび disabledTextFormat スタイルにより、TextArea が表示するテキストのスタイルが制御されます。スキンのスタイルプロパティについて詳しくは、120 ページの「[TextArea コンポーネントでのスキンの使用](#)」を参照してください。

次の例では、disabledTextFormat スタイルを設定し、TextArea が無効になっているが、有効になっている TextArea に対する textFormat スタイルの設定に同じプロセスが適用される場合のテキストの外観を変更します。

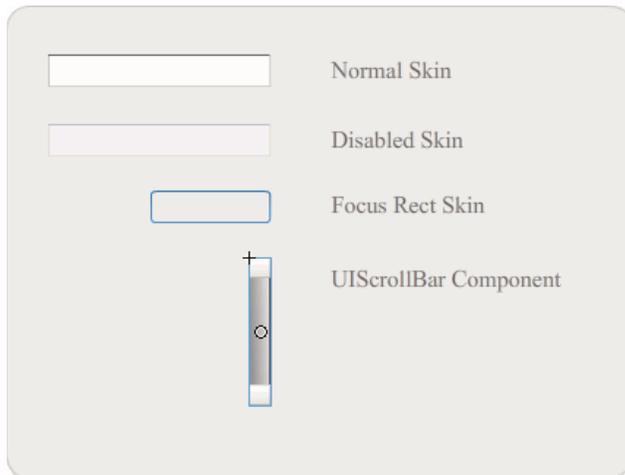
- 1 Flash ファイルを新規作成します。
- 2 TextArea コンポーネントをステージまでドラッグし、インスタンス名を **myTa** にします。
- 3 メインタimelineのフレーム 1 のアクションパネルに次のコードを追加します。

```
var tf:TextFormat = new TextFormat();
tf.color = 0xCC99FF;
tf.font = "Arial Narrow";
tf.size = 24;
myTa.setStyle("disabledTextFormat", tf);
myTa.text = "Hello World";
myTa.setSize(120, 50);
myTa.move(200, 50);
myTa.enabled = false;
```

- 4 制御/ムービープレビューを選択します。

TextArea コンポーネントでのスキンの使用

TextArea コンポーネントでは次のスキンを使用します。これらのスキンは編集して外観を変更することができます。



TextArea のスキン

注意：1つのコンポーネントで ScrollBar スキンを変更すると、ScrollBar を使用するその他のすべてのコンポーネントでスキンが変更されます。

次の手順では、TextArea にフォーカスがある場合に表示される Focus Rect スキンの境界線色と、Normal スキンを変更します。

- 1 Flash ファイルを新規作成します。
- 2 TextArea コンポーネントをステージにドラッグし、ダブルクリックしてスキンアイコンのパネルを開きます。
- 3 Focus Rect スキンをダブルクリックします。
- 4 Focus Rect スキンの境界線をクリックして選択します。バーを選択すると、現在のカラーがプロパティインスペクターの塗りカラーピッカーに表示されます。
- 5 プロパティインスペクターで塗りのカラーピッカーをクリックして開き、カラー #CC0000 を選択して、境界線に適用します。
- 6 ステージ上の編集バーの左側にある「戻る」ボタンをクリックし、ドキュメント編集モードに戻ります。
- 7 TextArea コンポーネントをダブルクリックして、スキンアイコンのパネルを開きます。
- 8 Normal スキンをダブルクリックします。
- 9 Normal スキンの境界線の端を1つずつ選択し、そのカラーを #990099 に設定します。
- 10 ステージ上の編集バーの左側にある「戻る」ボタンをクリックし、ドキュメント編集モードに戻ります。
- 11 制御/ムービープレビューを選択します。

TextArea を選択してテキストの入力を開始すると、その境界線は次の図のように表示されます。



外側の境界線が Focus Rect スキンであり、内側の境界線が Normal スキンの境界線です。

UIScrollBar スキンの編集について詳しくは、125 ページの「[UIScrollBar コンポーネントのカスタマイズ](#)」を参照してください。

TextInput コンポーネントのカスタマイズ

オーサリング時および実行時には、TextInput インスタンスのサイズを変更できます。オーサリング時には、ステージ上でコンポーネントを選択して、自由変形ツールを使用するか、修正／変形でいずれかのコマンドを使用します。実行時には、setSize() メソッドを使用するか、TextInput クラスの適切なプロパティ (height、width、scaleX、scaleY など) を使用します。

TextInput コンポーネントのサイズを変更すると、境界線は新しい境界ボックスのサイズに合わせて変更されます。TextInput コンポーネントでは、スクロールバーは使用されませんが、ユーザーがテキストを操作すると挿入ポインタが自動的にスクロールします。テキストフィールドのサイズは残りの領域内で変更されます。TextInput コンポーネントには、固定サイズのエレメントはありません。TextInput コンポーネントが小さすぎてテキスト全体を表示できない場合は、テキストが切り詰められます。

スタイルと TextInput コンポーネント

TextInput コンポーネントのスタイルは、コンポーネントが描画されるときにスキン、パディングおよびテキストフォーマットの値を指定します。textFormat スタイルおよび disabledTextFormat スタイルにより、コンポーネントに表示されるテキストのスタイルが制御されます。スキンのスタイルプロパティについて詳しくは、122 ページの「[スキンと TextInput コンポーネント](#)」を参照してください。

次の例では、textFormat スタイルを設定して、TextInput コンポーネントに表示されるテキストのフォント、サイズおよびカラーを設定します。同じプロセスが、コンポーネントが無効になってきたときに適用される disabledTextFormat スタイルの設定に適用されます。

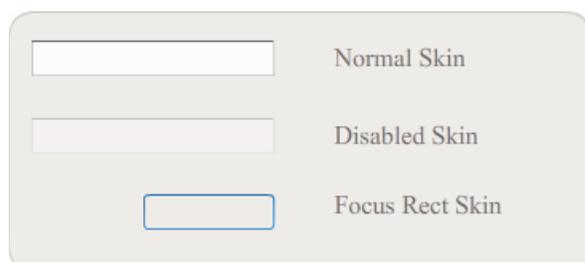
- 1 新しい Flash ドキュメント (ActionScript 3.0) を作成します。
- 2 TextInput コンポーネントをステージまでドラッグし、インスタンス名を **myTi** にします。
- 3 メインタイムラインのフレーム 1 のアクションパネルに次のコードを追加します。

```
var tf:TextFormat = new TextFormat();  
tf.color = 0x0000FF;  
tf.font = "Verdana";  
tf.size = 30;  
tf.align = "center";  
tf.italic = true;  
myTi.setStyle("textFormat", tf);  
myTi.text = "Enter your text here";  
myTi.setSize(350, 50);  
myTi.move(100, 50);
```

- 4 制御／ムービープレビューを選択します。

スキンと TextInput コンポーネント

TextInput コンポーネントでは次のスキンを使用します。これらのスキンは編集して外観を変更することができます。

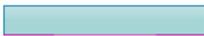


TextInput のキャプション

次の手順では、TextInput コンポーネントの境界線と背景のカラーを変更します。

- 1 Flash ファイルを新規作成します。
- 2 TextInput コンポーネントをステージにドラッグし、ダブルクリックしてスキンのパネルを開きます。
- 3 Normal スキンをダブルクリックします。
- 4 ズームコントロールを 800% に設定して、編集のためにアイコンを大きくします。
- 5 Normal スキンの境界線の端を 1 つずつ選択し、そのカラーを #993399 に設定して適用します。
- 6 プロパティインスペクターの塗りカラーピッカーにカラーが表示されるまで、背景をダブルクリックします。カラー #99CCCC を選択して背景に適用します。
- 7 ステージ上の編集バーの左側にある「戻る」ボタンをクリックし、ドキュメント編集モードに戻ります。
- 8 制御/ムービープレビューを選択します。

TextInput コンポーネントは次の図のように表示されます。



TileList コンポーネントのカスタマイズ

TileList コンポーネントは、オーサリング時および実行時に、縦横に変形できます。オーサリング時には、ステージ上でコンポーネントを選択して、自由変形ツールを使用するか、修正/変形でいずれかのコマンドを使用します。実行時には、setSize() メソッドを使用するか、適切なプロパティ (width、height、columnCount、rowCount、scaleX、scaleY など) を使用します。TileList に含まれる ScrollBar は、リストボックスに対応します。

スタイルと TileList コンポーネント

TileList のスタイルは、コンポーネントが描画されるときのスキン、パディングおよびテキストフォーマットの値を指定します。textFormat スタイルおよび disabledTextFormat スタイルにより、コンポーネントに表示されるテキストのスタイルが制御されます。スキんスタイルについて詳しくは、124 ページの「[TileList コンポーネントでのスキンの使用](#)」を参照してください。

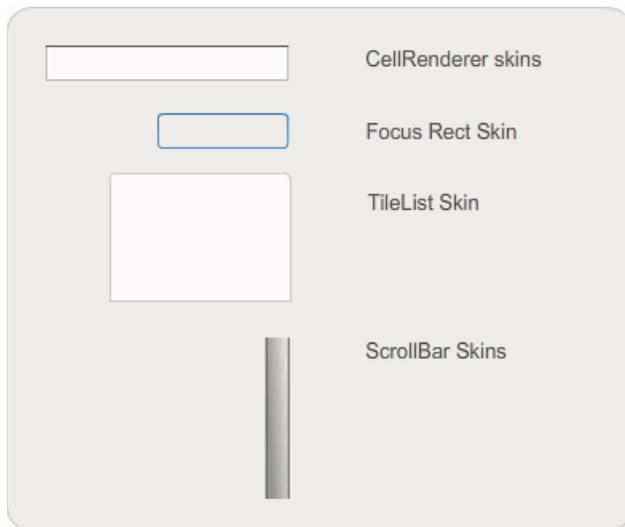
次の例では、textFormat スタイルを使用して setRendererStyle() メソッドを呼び出し、TileList インスタンスを表示するラベルのフォント、サイズ、カラーおよびテキストの各属性を設定します。同じプロセスが、enabled プロパティを false に設定したときに適用される disabledTextFormat スタイルの設定にも適用されます。

- 1 新しい Flash ドキュメント (ActionScript 3.0) を作成します。
- 2 TileList コンポーネントをステージまでドラッグし、インスタンス名を **myTl** にします。
- 3 タイムラインのフレーム 1 のアクションパネルに次のコードを追加します。

```
myTl.setSize(100, 100);  
myTl.addItem({label:"#1"});  
myTl.addItem({label:"#2"});  
myTl.addItem({label:"#3"});  
myTl.addItem({label:"#4"});  
var tf:TextFormat = new TextFormat();  
tf.font = "Arial";  
tf.color = 0x00FF00;  
tf.size = 16;  
tf.italic = true;  
tf.bold = true;  
tf.underline = true;  
tf.align = "center";  
myTl.setRendererStyle("textFormat", tf);
```

TileList コンポーネントでのスキンの使用

TileList コンポーネントには、TileList スキン、CellRenderer スキン、および ScrollBar スキンがあります。これらのスキンを編集して、TileList の外観を変更できます。



TileList のスキン

注意：1つのコンポーネントで ScrollBar スキンを変更すると、ScrollBar を使用するその他のすべてのコンポーネントでスキンが変更されます。

次の手順では、TileList の CellRenderer Selected_Up スキンのカラーを変更します。

- 1 Flash ドキュメント (ActionScript 3.0) を作成します。
- 2 TileList コンポーネントをステージまでドラッグし、ダブルクリックしてスキンのパネルを開きます。
- 3 CellRenderer スキンをダブルクリックし、Selected_Up スキンをダブルクリックして、矩形の背景をクリックします。
- 4 プロパティインスペクターの塗りカラーピッカーを使用してカラー #99FFFF を選択し、Selected_Up スキンに適用します。
- 5 ステージ上の編集バーの左側にある「戻る」ボタンをクリックし、ドキュメント編集モードに戻ります。
- 6 プロパティインスペクターの「パラメーター」タブで、dataProvider 行の 2 番目の列をダブルクリックして値ダイアログボックスを開きます。1st item、2nd item、3rd item、および 4th item の各ラベルのアイテムを追加します。
- 7 制御/ムービープレビューを選択します。

- 8 TileList のいずれかのセルをクリックして選択し、選択したセルからマウスポインターを離します。
選択したセルの表示は次の図のようになります。



変更された Selected_Up スキンカラーの TileList コンポーネント

UILoader コンポーネントのカスタマイズ

UILoader コンポーネントは、オーサリング時および実行時に、縦横に変形できます。オーサリング時には、ステージ上でコンポーネントを選択して、自由変形ツールを使用するか、修正／変形でいずれかのコマンドを使用します。実行時には、setSize() メソッドを使用するか、適切なプロパティ (width、height、scaleX、scaleY など) を使用します。

UILoader コンポーネントのサイズ変更動作は scaleContent プロパティで制御します。scaleContent が true のときは、Loader の境界線内に収まるようにコンテンツが拡大 / 縮小されます (setSize() を呼び出すと、拡大 / 縮小が再度行われます)。scaleContent が false のときは、コンポーネントのサイズがコンテンツのサイズに合わせて調整され、setSize() およびサイズ調整のプロパティが無効になります。

UILoader コンポーネントには、スタイルまたはスキンを適用できるユーザーインターフェイスエレメントがありません。

UIScrollBar コンポーネントのカスタマイズ

UIScrollBar コンポーネントは、オーサリング時および実行時に、縦横に変形できます。ただし、垂直方向の UIScrollBar の幅、水平方向の UIScrollBar の高さを変更することはできません。オーサリング時には、ステージ上でコンポーネントを選択して、自由変形ツールを使用するか、修正／変形でいずれかのコマンドを使用します。実行時には、setSize() メソッドを使用するか、UIScrollBar クラスの任意の適切なプロパティ (width、height、scaleX、scaleY など) を使用します。

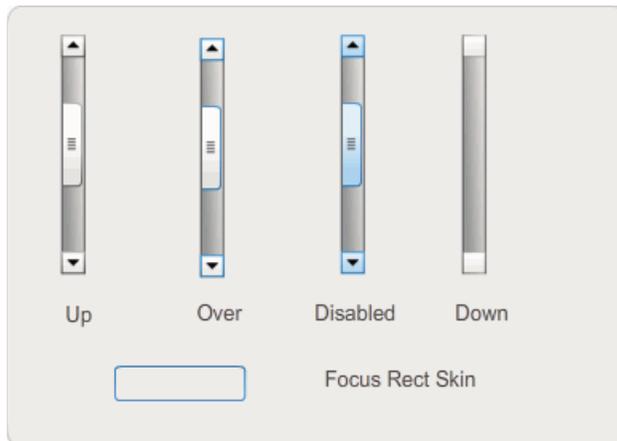
注意：setSize() メソッドを使用する場合は、水平スクロールバーの幅、または垂直スクロールバーの高さのみを変更できます。オーサリング時には、水平スクロールバーの高さまたは垂直スクロールバーの幅を設定できますが、ムービーをバブルリッシュするときには値はリセットされます。スクロールバーの長さ方向に対応するサイズのみ変更できます。

UIScrollBar コンポーネントでのスタイルの使用

UIScrollBar コンポーネントのスタイルは、そのスキンのクラスと、コンポーネントの境界ボックスと外側の境界の間のパディングに使用するピクセル数を指定する FocusRectPadding の値のみを指定します。スキンスタイルの使用について詳しくは、96 ページの「[スキンについて](#)」を参照してください。

UIScrollBar コンポーネントでのスキンの使用

UIScrollBar コンポーネントでは次のスキンが使用されます。



UIScrollBar のスキン

水平スクロールバーと垂直スクロールバーはいずれも同じスキンを使用し、水平スクロールバーの表示時に UIScrollBar コンポーネントが必要に応じてスキンを回転します。

注意：1つのコンポーネントで ScrollBar スキンを変更すると、ScrollBar を使用するその他のすべてのコンポーネントでスキンが変更されます。

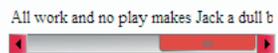
次の例は、UIScrollBar のサムと矢印ボタンのカラーを変更する方法を示しています。

- 1 新しい Flash ドキュメント (ActionScript 3.0) を作成します。
- 2 UIScrollBar コンポーネントをステージまでドラッグし、インスタンス名を **mySb** にします。「パラメーター」タブで、「direction」を「horizontal」に設定します。
- 3 スクロールバーをダブルクリックしてスキンのパネルを開きます。
- 4 Up スキンをクリックして選択します。
- 5 ズームコントロールを 400% に設定して、編集のためにアイコンを大きくします。
- 6 背景が選択され、カラーがプロパティインスペクターの塗りカラーピッカーに表示されるまで、右矢印 (または垂直スクロールバーの上矢印) の背景をダブルクリックします。
- 7 カラー #CC0033 を選択してボタンの背景に適用します。
- 8 ステージの上の編集バーの左側にある「戻る」ボタンをクリックし、ドキュメント編集モードに戻ります。
- 9 サムおよび左矢印 (または垂直スクロールバーの下矢印) エレメントに対して、手順 6、7、および 8 を繰り返します。
- 10 タイムラインのフレーム 1 のアクションパネルに次のコードを追加し、スクロールバーを TextField に割り当てます。

```
var tf:TextField = new TextField();
addChild(tf);
tf.x = 150;
tf.y = 100;
mySb.width = tf.width = 200;
tf.height = 22;
tf.text = "All work and no play makes Jack a dull boy. All work and no play makes Jack a dull boy. All
. . .";
mySb.y = tf.y + tf.height;
mySb.x = tf.x + tf.width;x
mySb.scrollTarget = tf;
```

11 制御/ムービープレビューを選択します。

UIScrollBar コンポーネントの表示は次の図のようになります。



赤いサムと左右の矢印の付いた水平 ScrollBar

第6章：FLVPlayback コンポーネントの使用

FLVPlayback コンポーネントを使用すると、Adobe Flash CS5 Professional アプリケーションにビデオプレーヤーを簡単に組み込むことができます。また、HTTP 経由でプログレッシブダウンロードしたビデオファイルの再生、および Adobe の Macromedia Flash Media Server や Flash Video Streaming Service (FVSS) からのストリーミングビデオファイルの再生も可能です。

Adobe Flash Player 9 Update 3 (バージョン 9.0.115.0 以降) のリリースにより、Flash Player でのビデオコンテンツの再生が大幅に改善されました。この更新には、エンドユーザーのシステムビデオハードウェアを利用してビデオ再生のパフォーマンスを向上させる、FLVPlayback コンポーネントの変更が含まれています。また、FLVPlayback コンポーネントの変更により、フルスクリーンモードで表示されるビデオファイルの忠実度も向上しました。

さらに Flash Player 9 Update 3 では、業界標準の H.264 エンコーディングを利用する高解像度の MPEG-4 ビデオ形式がサポートされ、FLVPlayback コンポーネントの機能が向上しています。サポートされる形式は、MP4、M4A、MOV、MP4V、3GP および 3G2 です。

注意：ただし、Apple® iTunes® からダウンロードしたファイルや FairPlay® でデジタル暗号化されたファイルなど、保護された MP4 ファイルはサポートされません。

FLVPlayback コンポーネントは使いやすく、次のような特徴と長所があります。

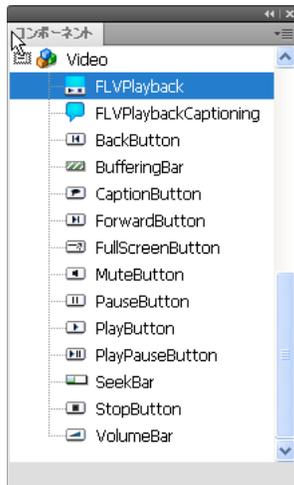
- ステージへのドラッグによって、すばやく確実に実装できる
- フルスクリーンサイズをサポート
- あらかじめ用意されているデザイン済みスキンのコレクションを使用して、再生コントロールの外観をカスタマイズできる
- デザイン済みスキンの色とアルファ値を選択できる
- 上級ユーザーは独自スキンを作成できる
- オーサリング時にライブプレビューを使用できる
- レイアウトプロパティによって、サイズ変更時もビデオファイルを中央に配置できる
- ビデオファイルのプログレッシブダウンロードがある程度進んだ時点で再生を開始できる
- キューポイントを使用して、ビデオをテキスト、グラフィック、およびアニメーションと同期できる
- 適度なサイズの SWF ファイルが保持される

FLVPlayback コンポーネントの使用

FLVPlayback コンポーネントを使用するには、ステージにコンポーネントを配置し、そのコンポーネントで再生するビデオファイルを指定する必要があります。さらに、動作を制御したりビデオファイルを指定したりする様々なパラメーターを設定することもできます。

FLVPlayback コンポーネントには、ActionScript アプリケーションプログラミングインターフェイス (API) も含まれます。API には、CuePointType、FLVPlayback、FLVPlaybackCaptioning、NCManager、NCManagerNative、VideoAlign、VideoError、VideoPlayer、VideoState の各クラス、およびイベントクラスとして AutoLayoutEvent、LayoutEvent、MetadataEvent、SkinErrorEvent、SoundEvent、VideoEvent および VideoProgressEvent が含まれています。詳しくは、**Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド** を参照してください。

FLVPlayback コンポーネントには、FLV Playback カスタム UI コンポーネントが含まれています。FLVPlayback コンポーネントは、ビデオファイルを表示するための表示領域（ビデオプレーヤー）と、その操作に使用できるコントロールを組み合わせたものです。FLV Playback カスタム UI コンポーネントには、ビデオファイルの再生、停止、一時停止およびその他の制御に使用できるコントロールボタンとメカニズムが備わっています。これらのコントロールには、BackButton、BufferingBar、CaptionButton (FLVPlaybackCaptioning 用)、ForwardButton、FullScreenButton、MuteButton、PauseButton、PlayButton、PlayPauseButton、SeekBar、StopButton、および VolumeBar があります。次の図に示すように、FLVPlayback コンポーネントと FLV Playback カスタム UI コントロールはコンポーネントパネルに表示されます。



コンポーネントパネルの FLVPlayback コンポーネント

FLVPlayback コンポーネントに再生コントロールを追加するプロセスを、スキンの適用と呼びます。FLVPlayback コンポーネントには、最初のデフォルトスキンとして、SkinOverAll.swf があります。このスキンには、再生、停止、戻る、早送り、シークバー、ミュート、ボリューム、フルスクリーン、およびキャプションの各コントロールがあります。このスキンを変更するには、次の方法を使用できます。

- デザイン済みスキンのコレクションから選択する
- カスタムスキンを作成し、デザイン済みスキンのコレクションにそれを追加する
- FLV Playback カスタム UI コンポーネントから個々のコントロールを選択してカスタマイズする

デザイン済みスキンを選択するときには、オーサリング時または実行時のいずれかに、スキンの色とアルファ値を個別に選択できます。詳しくは、146 ページの「[デザイン済みスキンの選択](#)」を参照してください。

別のスキンを選択すると、選択したスキンが新しいデフォルトスキンになります。

スキンを FLVPlayback コンポーネントに選択したり、作成したりする方法について詳しくは、146 ページの「[FLVPlayback コンポーネントのカスタマイズ](#)」を参照してください。

FLVPlayback コンポーネントを使用したアプリケーションの作成

アプリケーションで FLVPlayback コンポーネントを使用するには、次のような方法があります。

- コンポーネントパネルからステージに FLVPlayback コンポーネントをドラッグし、source パラメーターの値を指定する
- ビデオの読み込みウィザードを使用して、ステージ上にコンポーネントを作成し、スキンを選択してカスタマイズする
- コンポーネントがライブラリ内にあることを前提として、FLVPlayback() コンストラクターを使用してステージ上に FLVPlayback インスタンスを動的に作成する

注意：ActionScript で FLVPlayback のインスタンスを作成した場合、ActionScript で skin プロパティを設定してスキンを割り当てる必要もあります。この方法で適用するスキンは、SWF ファイルに自動的にパブリッシュされません。アプリケーションの SWF ファイルとスキンの SWF ファイルの両方をアプリケーションサーバーにコピーする必要があります。そうしないと、アプリケーションを実行するときにスキンの SWF ファイルを利用できません。

コンポーネントパネルからの FLVPlayback コンポーネントのドラッグ

- 1 コンポーネントパネルで、プラス「+」ボタンをクリックしてビデオエントリを開きます。
- 2 FLVPlayback コンポーネントをステージにドラッグします。
- 3 ステージ上で FLVPlayback コンポーネントを選択した状態で、コンポーネントインスペクターの「パラメーター」タブにある source パラメーターの「値」セルに、次のいずれかを指定するストリングを入力します。
 - ビデオファイルのローカルパス
 - ビデオファイルの URL
 - ビデオファイルの再生方法を指定する SMIL (Synchronized Multimedia Integration Language) ファイルの URL
SMIL ファイルを作成して FLV ファイルを指定する方法については、156 ページの「[SMIL ファイルの使用](#)」を参照してください。
- 4 ステージ上で FLVPlayback コンポーネントを選択した状態で、コンポーネントインスペクターの「パラメーター」タブで、skin パラメーターの「値」セルをクリックします。
- 5 拡大鏡アイコンをクリックして、スキンの選択ダイアログボックスを開きます。
- 6 次のオプションから 1 つを選択します。
 - 「スキン」ドロップダウンリストから、デザイン済みスキンを選択し、再生コントロールのセットをコンポーネントに割り当てます。
 - カスタムスキンを作成してある場合は、ポップアップメニューから「カスタムスキンの URL」を選択し、スキンが含まれる SWF ファイルの URL を「URL」ボックスに入力します。
 - 「なし」を選択し、個々の FLV Playback カスタム UI コンポーネントをステージにドラッグして再生コントロールを追加します。

注意：最初の 2 つのオプションでは、ポップアップメニューの上にある表示ペインにスキンが表示されます。カラーピッカーを使用して、スキンの色を変更できます。

カスタム UI コントロールの色を変更するには、カスタマイズする必要があります。カスタム UI コントロールの使用方法について詳しくは、147 ページの「[FLV Playback カスタム UI コンポーネントへの個々のスキンの適用](#)」を参照してください。
- 7 「OK」をクリックして、スキンの選択ダイアログボックスを閉じます。
- 8 制御/ムービープレビューを選択し、SWF ファイルを実行してビデオを再生します。
ビデオの読み込みウィザードを使用して FLVPlayback コンポーネントを追加する手順を次に示します。

ビデオの読み込みウィザードの使用：

- 1 ファイル/読み込み/ビデオの読み込みを選択します。
- 2 次のいずれかのオプションを選択して、ビデオファイルの場所を示します。
 - ローカルマシン上
 - Web サーバー、Flash Video Streaming Service、または Flash Media Server に既に展開済み
- 3 選択内容によって、ビデオファイルの場所を示すパスまたは URL を入力し、「次へ」をクリックします。

4 ファイルパスを選択した場合は、展開ダイアログボックスが表示され、次のいずれかのオプションを選択してビデオの展開方法を指定できます。

- 標準 Web サーバーからのプログレッシブダウンロード
- Flash Video Streaming Service (FVSS) からのストリーム
- Flash Media Server からのストリーム
- SWF ファイルにビデオを埋め込みタイムラインで再生

重要：ビデオの埋め込みオプションは選択しないでください。FLVPlayback コンポーネントでは、外部のストリーミングビデオだけを再生します。このオプションを選択すると、FLVPlayback コンポーネントはステージに配置されません。

5 「次へ」をクリックします。

6 次のオプションから 1 つを選択します。

- 「スキン」ドロップダウンリストから、デザイン済みスキンを選択し、再生コントロールのセットをコンポーネントに割り当てます。
- コンポーネントのカスタムスキンを作成した場合は、ポップアップメニューから「カスタムスキンの URL」を選択し、スキンを含む SWF ファイルの URL を「URL」ボックスに入力します。
- 「なし」を選択し、個々の FLV Playback カスタム UI コンポーネントをステージにドラッグして再生コントロールを追加します。

注意：最初の 2 つのオプションでは、ポップアップメニューの上にある表示ペインにスキンが表示されます。

7 「OK」をクリックして、スキンの選択ダイアログボックスを閉じます。

8 ビデオの読み込みの終了ダイアログボックスに表示される情報を読み、次の手順について理解してから「終了」をクリックします。

9 FLA ファイルを保存していない場合は、名前を付けて保存ダイアログボックスが表示されます。

10 制御/ムービープレビューを選択し、SWF ファイルを実行してビデオを再生します。

ActionScript を使用して FLVPlayback コンポーネントを追加する手順を次に示します。

ActionScript を使用したインスタンスの動的な作成：

1 コンポーネントパネルから FLVPlayback コンポーネントをライブラリパネル（ウィンドウ/ライブラリ）にドラッグします。

2 タイムラインのフレーム 1 のアクションパネルに次のコードを追加します。install_drive の部分に Flash のインストール先ドライブを指定し、実際の Skins フォルダの場所に合わせてパスを変更します。

Windows コンピューターの場合：

```
import fl.video.*;
var my_FLVPlybk = new FLVPlayback();
my_FLVPlybk.x = 100;
my_FLVPlybk.y = 100;
addChild(my_FLVPlybk);
my_FLVPlybk.skin = "file:///install_drive|Program Files/Adobe/Adobe Flash
CS5/en/Configuration/FLVPlayback Skins/ActionScript 3.0/SkinOverPlaySeekMute.swf"
my_FLVPlybk.source = "http://www.helpexamples.com/flash/video/water.flv";
```

Macintosh コンピューターの場合：

```
import fl.video.*;
var my_FLVPlybk = new FLVPlayback();
my_FLVPlybk.x = 100;
my_FLVPlybk.y = 100;
addChild(my_FLVPlybk);
my_FLVPlybk.skin = "file:///Macintosh HD:Applications:Adobe Flash CS5:Configuration:FLVPlayback
Skins:ActionScript 3.0SkinOverPlaySeekMute.swf"
my_FLVPlybk.source = "http://www.helpexamples.com/flash/video/water.flv";
```

注意：source プロパティと skin プロパティを設定しないと、生成されるムービークリップには何も表示されません。

- 3 制御/ムービープレビューを選択し、SWF ファイルを実行してビデオファイルを再生します。

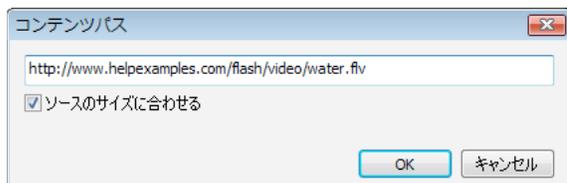
FLVPlayback コンポーネントのパラメーター

FLVPlayback コンポーネントの各インスタンスについて、コンポーネントインスペクターまたはプロパティインスペクターでパラメーターとして align、autoPlay、cuePoints、preview、scaleMode、skin、skinAutoHide、skinBackgroundAlpha、skinBackgroundColor、source および volume を設定できます。これらの各パラメーターには、対応する同名の ActionScript プロパティがあります。これらのパラメーターに値を割り当てると、アプリケーション内でプロパティの初期状態が設定されます。ActionScript で設定したプロパティは、パラメーターで設定した値よりも優先されます。これらのパラメーターに設定できる値については、「**Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド**」の FLVPlayback クラスを参照してください。

FLVPlayback source パラメーターの指定

source パラメーターを使用すると、ビデオファイルの名前と場所を指定できます。この情報により、ファイルの再生方法が Flash に指示されます。

コンポーネントインスペクターで source パラメーターの「値」セルをダブルクリックすると、コンテンツパスダイアログボックスが開きます。



FLVPlayback のコンテンツパスダイアログボックス

コンテンツパスダイアログボックスには、ステージ上の FLVPlayback インスタンスをソースビデオファイルのサイズに合わせるかどうかを指定する「ソースとなる FLV のサイズに合わせる」チェックボックスがあります。ソースビデオファイルには、再生に適した推奨の高さと幅が指定されています。このオプションを選択した場合、FLVPlayback インスタンスのサイズは、推奨のサイズに合わせて変更されます。

ソース

ビデオファイルの再生方法を指定したビデオファイルまたは XML ファイルの URL またはローカルパスを入力します。ビデオファイルの正確な保管場所がわからない場合は、フォルダーアイコンをクリックすると、参照ダイアログボックスが開き、ビデオファイルの場所を特定できます。ビデオファイルを参照する際に、ビデオファイルがターゲット SWF ファイルと同じ場所またはその下位にある場合は、Web サーバーで使用できるように、その場所への相対パスが自動的に作成されます。それ以外の場合、パスは Windows または Macintosh の絶対パスとなります。ローカル XML ファイルの名前を入力するには、パスと名前を入力します。

HTTP URL を指定すると、ビデオファイルはプログレッシブダウンロードとして再生されます。RTMP URL を指定すると、ビデオファイルは Flash Media Server または FVSS からストリーミングされます。XML ファイルの URL を指定しても、ビデオファイルが Flash Media Server または FVSS からストリーミングされる場合があります。

重要：

複数の帯域幅で複数のビデオファイルストリームを再生する方法を指定した SMIL ファイルの場所を指定することもできます。このファイルでは、SMIL (Synchronized Multimedia Integration Language) を使用して FLV ファイルを記述します。SMIL ファイルについては、156 ページの「[SMIL ファイルの使用](#)」を参照してください。

ActionScript の FLVPlaybacksource プロパティ、FLVPlayback.play() メソッドおよび FLVPlayback.load() メソッドを使用して、ビデオファイルの名前と場所を指定することもできます。これらの 3 つの方法は、コンポーネントインスペクターの source パラメーターよりも優先されます。詳しくは、FLVPlayback.source、FLVPlayback.play() および FLVPlayback.load() の各エントリ（「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の FLVPlayback クラス）を参照してください。

フルスクリーンのサポート

FLVPlayback コンポーネントの ActionScript 3.0 バージョンでは、フルスクリーンモードをサポートしています。このモードには、Flash Player 9.0.28.0 以降が必要であり、フルスクリーンで表示されるように HTML を正しく設定する必要があります。一部のデザイン済みスキンには、フルスクリーンモードのオンとオフを切り替えるためのトグルボタンが用意されています。FullScreenButton アイコンは、次の図のコントロールバーの右側に表示されています。



コントロールバーのフルスクリーンアイコン

フルスクリーンモードは、fullScreenTakeOver プロパティが true（デフォルト値）に設定されている場合にのみサポートされます。

フルスクリーンのサポートは、ハードウェアアクセラレーションがサポートされているかどうかに関係なく、実装できます。ハードウェアアクセラレーションのサポートについては、135 ページの「[ハードウェアアクセラレーション](#)」を参照してください。

FLVPlayback にフルスクリーンのサポートを実装するには：

- 1 FLVPlayback コンポーネントをアプリケーションに追加し、ビデオファイルを割り当てます。
- 2 フルスクリーンのボタン（例えば、SkinUnderPlaySeekFullscreen.swf）を含むスキンを FLVPlayback コンポーネントに選択するか、コンポーネントパネルの「ビデオ」セクションから FLVPlayback コンポーネントに FullScreenButton ユーザーインターフェイスコンポーネントを追加します。
- 3 ファイル/パブリッシュ設定を選択します。
- 4 パブリッシュ設定ダイアログボックスで、「HTML」タブをクリックします。
- 5 「HTML」タブで、テンプレートポップアップメニューから「Flash でのフルスクリーンサポート」を選択します。
- 6 また、「HTML」タブで、「フラッシュのバージョンを検出」チェックボックスを選択し、使用している Flash Player のバージョンに応じて、9.0.28 以降のバージョンを指定します。
- 7 「フォーマット」タブを選択し、「Flash (.swf)」オプションと「HTML (.html)」オプションの両方が選択されていることを確認します。デフォルトのファイル名を置き換えることができます。
- 8 「パブリッシュ」をクリックし、「OK」をクリックします。

手順 7 の代わりに「OK」をクリックし、ファイル/パブリッシュプレビュー/デフォルト - (HTML) を選択すると、書き出した HTML ファイルがデフォルトのブラウザに表示されます。または、書き出した HTML ファイルをブラウザで表示して、フルスクリーンオプションをテストします。

フルスクリーンモードがサポートされた FLVPlayback コンポーネントを Web ページに追加するには、書き出した HTML ファイルを開き、SWF ファイルを Web ページの HTML ファイルに埋め込むコードをコピーします。このコードは次の例のようになります。

```
//from the <head> section

<script language="javascript"> AC_FL_RunContent = 0; </script>
<script language="javascript"> DetectFlashVer = 0; </script>
<script src="AC_RunActiveContent.js" language="javascript"></script>
<script language="JavaScript" type="text/javascript">
<!--
// -----
// Globals
// Major version of Flash required
var requiredMajorVersion = 9;
// Minor version of Flash required
var requiredMinorVersion = 0;
// Revision of Flash required
var requiredRevision = 28;
// -----
// -->
</script>

//and from the <body> section

<script language="JavaScript" type="text/javascript">
<!--
if (AC_FL_RunContent == 0 || DetectFlashVer == 0) {
    alert("This page requires AC_RunActiveContent.js.");
} else {
    var hasRightVersion = DetectFlashVer(requiredMajorVersion,
        requiredMinorVersion, requiredRevision);
    if (hasRightVersion) { // if we've detected an acceptable version
        // embed the Flash movie
        AC_FL_RunContent(
            &apos;codebase&apos;, &apos;http://download.macromedia.com/pub/
                shockwave/cabs/flash/swflash.cab#version=9,0,28,0&apos;,
            &apos;width&apos;, &apos;550&apos;,
            &apos;height&apos;, &apos;400&apos;,
            &apos;src&apos;, &apos;fullscreen&apos;,
            &apos;quality&apos;, &apos;high&apos;,
            &apos;pluginspage&apos;, &apos;http://www.macromedia.com/go/
                getflashplayer&apos;,
            &apos;align&apos;, &apos;middle&apos;,
            &apos;play&apos;, &apos>true&apos;,
            &apos;loop&apos;, &apos>true&apos;,
            &apos;scale&apos;, &apos;showall&apos;,
            &apos;wmode&apos;, &apos>window&apos;,
            &apos;devicefont&apos;, &apos>false&apos;,
            &apos;id&apos;, &apos;fullscreen&apos;,
            &apos;bgcolor&apos;, &apos;#ffffff&apos;,
            &apos;name&apos;, &apos;fullscreen&apos;,
            &apos;menu&apos;, &apos>true&apos;,
            &apos;allowScriptAccess&apos;, &apos;sameDomain&apos;,
```

```

    &apos;allowFullScreen&apos;, &apos;true&apos;,
    &apos;movie&apos;, &apos;fullscreen&apos;,
    &apos;align&apos;, &apos;&apos; ); //end AC code
  } else { // Flash is too old or we can&apos;t detect the plug-in.
    var alternateContent = &apos;Alternative HTML content should be placed
      here.&apos;
    + &apos;This content requires Adobe Flash Player.&apos;
    + &apos;<a href=http://www.macromedia.com/go/getflash/>Get Flash</a>
      &apos;;
    document.write(alternateContent); // Insert non-Flash content.
  }
}
// -->
</script>
<noscript>
  // Provide alternative content for browsers that do not support scripting
  // or for those that have scripting disabled.
  Alternative HTML content should be placed here. This content requires Adobe Flash Player.
  <a href="http://www.macromedia.com/go/getflash/">Get Flash</a>
</noscript>

```

また、書き出した HTML ファイルを Web ページのテンプレートとして使用し、他のコンテンツを追加することもできます。ただしその場合は、後で Flash から FLVPlayback の HTML ファイルを書き出すときに間違っファイルを上書きしないように、HTML ファイルの名前を変更します。

いずれの場合も、HTML ファイルと同じフォルダーに書き出した AC_RunActiveContent.js ファイルを、Web サーバーにアップロードする必要があります。

ActionScript によるフルスクリーンモードのサポートには、fullScreenBackgroundColor プロパティ、fullScreenSkinDelay プロパティ、fullScreenTakeOver プロパティおよび enterFullScreenDisplayState() メソッドが用意されています。これらの ActionScript エレメントについては、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」を参照してください。

enterFullScreenDisplayState() の使用

次の例に示すように、ActionScript の enterFullScreenDisplayState() メソッドを呼び出して、フルスクリーンモードを起動することもできます。

```

function handleClick(e:MouseEvent):void {
    myFLVPlaybk.enterFullScreenDisplayState();
}
myButton.addEventListener(MouseEvent.CLICK, handleClick);

```

この例では、FLVPlayback スキンのフルスクリーンコントロールボタンをクリックしてフルスクリーンモードを起動するのではなく、Web ページの作成者がフルスクリーンモードを起動するために挿入したボタン (MyButton) をクリックして起動します。このボタンをクリックすると、handleClick イベントハンドラーがトリガーされ、enterFullScreenDisplayState() メソッドが呼び出されます。

enterFullScreenDisplayState() メソッドによって Stage.displayState プロパティが StageDisplayState.FULL_SCREEN に設定され、displayState プロパティと同じ制限が適用されます。enterFullScreenDisplayState() メソッドと Stage.displayState プロパティについて詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」を参照してください。

ハードウェアアクセラレーション

Flash Player 9.0.115.0 以降のバージョンには、使用可能なビデオハードウェアを利用して、FLVPlayback がフルスクリーンモードで再生する FLV ファイルのパフォーマンスと忠実度を改善するコードが組み込まれています。前提条件が満たされており、fullScreenTakeOver プロパティが true に設定されている場合、Flash Player ではソフトウェアを使用してビデオ

ファイルを拡大するのではなく、ハードウェアアクセラレーションを使用してビデオファイルを拡大します。FLVPlayback コンポーネントを以前のバージョンの Flash Player で実行する場合、またはハードウェアアクセラレーションに関する前提条件が満たされていない場合は、以前と同様にビデオファイル自体を拡大します。

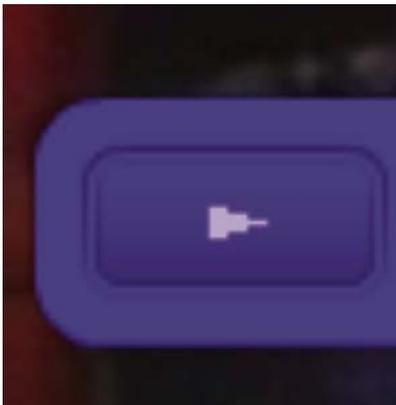
フルスクリーンモードのサポートのためにハードウェアアクセラレーションを利用するには、4 MB 以上の VRAM (ビデオ RAM) を備えた DirectX 7 対応のビデオカードがコンピューターに搭載されている必要があります。このハードウェアサポートは、Windows 2000 または Mac OS X 10.2、およびこれらのオペレーティングシステムのそれ以降のバージョンで利用できます。DirectX® は、ソフトウェアとビデオハードウェアの間でインターフェイスとして機能する API を提供し、特に 2 次元および 3 次元グラフィックの処理を高速化します。

また、ハードウェアアクセラレーションモードを利用するには、次のいずれかの方法でフルスクリーンモードを起動する必要があります。

- FLVPlayback スキン上のフルスクリーンボタンを使用する。
- FullScreenButton ビデオコントロールを使用する。
- enterFullScreenDisplayState() メソッドを使用する。詳しくは、135 ページの「[enterFullScreenDisplayState\(\) の使用](#)」を参照してください。

Stage.displayState プロパティを StageDisplayState.FULLSCREEN に設定してフルスクリーンモードを起動する場合、ビデオハードウェアおよびメモリが使用できる場合でも、FLVPlayback ではハードウェアアクセラレーションが使用されません。

フルスクリーンモードをサポートするためにハードウェアアクセラレーションを使用すると、ビデオプレーヤーおよびビデオファイルに合わせて、FLVPlayback のスキンが拡大されます。次の図は、ハードウェアアクセラレーションを使用したフルスクリーンモードで FLVPlayback のスキンを再生した結果 (フル解像度で表示されたイメージの詳細) を示しています。



解像度 1600 x 1200 のモニターでのフルスクリーンモード (幅 320 ピクセル、高さ 240 ピクセル)

このイメージは、解像度が 1600 x 1200 のモニターでフルスクリーンモードを使用して、幅 320 ピクセル、高さ 240 ピクセル (FLVPlayback コンポーネントのデフォルトのサイズ) のビデオファイルを再生した結果です。拡大されたスキンの歪みは、FLV ファイルのサイズが小さいほど、または表示するモニターが大きいほど目立ちます。これに対して、FLV ファイルが大きい場合、またはモニターが小さい場合、歪みはあまり目立ちません。例えば、640 x 480 を 1600 x 1200 に変更すると、スキンのサイズは大きくなりますが、歪みは小さくなります。

skinScaleMaximum プロパティを設定して、FLVPlayback スキンの拡大を制限できます。デフォルト値は 4.0 (400%) です。ただし、スキンの拡大を制限するには、FLV ファイルを拡大するためのハードウェアとソフトウェアの組み合わせが必要になります。場合によっては、スキンの拡大を制限すると、高いビットレートでエンコードした大きいサイズの FLV ファイルのパフォーマンスに悪影響を与えることがあります。ビデオが大きい場合 (例えば、幅が 640 ピクセル以

上で高さが 480 ピクセル以上の場合)、skinScaleMaximum プロパティを小さい値に設定しないでください。大きいモニターでは、パフォーマンスに甚大な問題が生じる可能性があります。skinScaleMaximum プロパティを使用すると、大きいスキンのパフォーマンス、画質および外観のトレードオフを解決できます。

フルスクリーンモードの終了

フルスクリーンモードを終了するには、フルスクリーンボタンをもう一度クリックするか、Esc キーを押します。

height、registrationHeight、registrationWidth、registrationX、registrationY、scaleX、scaleY、width、x、y、setScale()、setSize() などのプロパティやメソッドを設定したり、呼び出したりすると、レイアウトが変更され、その結果、FLVPlayback コンポーネントではフルスクリーンモードが終了します。

align プロパティまたは scaleMode プロパティを設定すると、フルスクリーンモードが終了するまで FLVPlayback コンポーネントではこれらのプロパティが center および maintainAspectRatio に設定されます。

フルスクリーンモードを使用しているときに、fullScreenTakeOver プロパティの値を true から false に変更しても、フルスクリーンモードが終了します。

複数のビデオファイルの再生でのレイアウトの配置

ActionScript 3.0 の FLVPlayback には align プロパティがあります。これは、ビデオファイルのサイズを変更したときに中央に配置したり、コンポーネントの上、下、左、または右に配置するように指定するプロパティです。コンポーネントの x、y、width および height の各プロパティに加えて、ActionScript 3.0 コンポーネントには registrationX、registrationY、registrationWidth および registrationHeight の各プロパティもあります。これらのプロパティは、最初は x、y、width および height の各プロパティと同じです。続いて別のビデオファイルをロードすると、自動再レイアウト機能によって変更されないで、新しいビデオファイルを同じ場所の中央に配置できます。scaleMode = VideoScaleMode.MAINTAIN_ASPECT_RATIO の場合、コンポーネントの幅と高さを変更するのではなく、後からロードした FLV ファイルをコンポーネントの元のサイズに合わせるすることができます。

プログレッシブダウンロードされるビデオファイルの自動再生

プログレッシブダウンロードされるビデオファイルをロードする場合、FLVPlayback で再生が開始されるのは、ビデオファイルのダウンロードが十分に進み、ビデオファイルを最初から最後まで再生できるようになってからです。

ダウンロードの途中でビデオファイルを再生するには、パラメーターを付けずに play() メソッドを呼び出します。

ビデオファイルが十分にダウンロードされるまで待機する状態に戻るには、pause() メソッドを呼び出し、playWhenEnoughDownloaded() メソッドを呼び出します。

キューポイントの使用

キューポイントは、ビデオファイルの再生中にビデオプレーヤーが cuePoint イベントを送出するポイントです。キューポイントは、Web ページ上の別のエレメントでアクションを発生させるときに、FLV ファイルに追加することができます。例えばテキストやグラフィックを表示する、Flash アニメーションと同期する、FLV ファイルを一時停止して再生を操作する、ビデオの別のポイントをシークする、別の FLV ファイルに切り替えることなどがあります。キューポイントを使用すると、ActionScript コードで制御を受け取り、FLV ファイルのそれらのポイントを Web ページ上の他のアクションと同期することができます。

キューポイントには、ナビゲーションキューポイント、イベントキューポイント、および ActionScript キューポイントの 3 種類があります。ナビゲーションキューポイントおよびイベントキューポイントは、FLV ファイルストリームおよび FLV ファイルのメタデータパケットに埋め込まれるので、埋め込みキューポイントとも呼ばれます。

ナビゲーションキューポイントを使用すると、FLV ファイルの特定のフレームをシークできます。これは、FLV ファイル内で、指定する時間に可能な限り近い時間にキーフレームが作成されるからです。キーフレームは、FLV ファイルストリーム内のイメージフレーム間で発生するデータセグメントです。ナビゲーションキューポイントを使用すると、コンポーネントはキーフレームをシークし、cuePoint イベントを開始します。

イベントキューポイントを使用すると、FLV ファイル内の時点と、Web ページ上の外部イベントと同期できます。cuePoint イベントは、指定した時間に正確に発生します。FLV ファイルにナビゲーションキューポイントおよびイベントキューポイントを埋め込むには、ビデオの読み込みウィザードまたは Flash Video Encoder を使用します。ビデオの読み込みウィザードと Flash Video Encoder については、『Flash ユーザーガイド』の第 16 章「ビデオの操作」を参照してください。

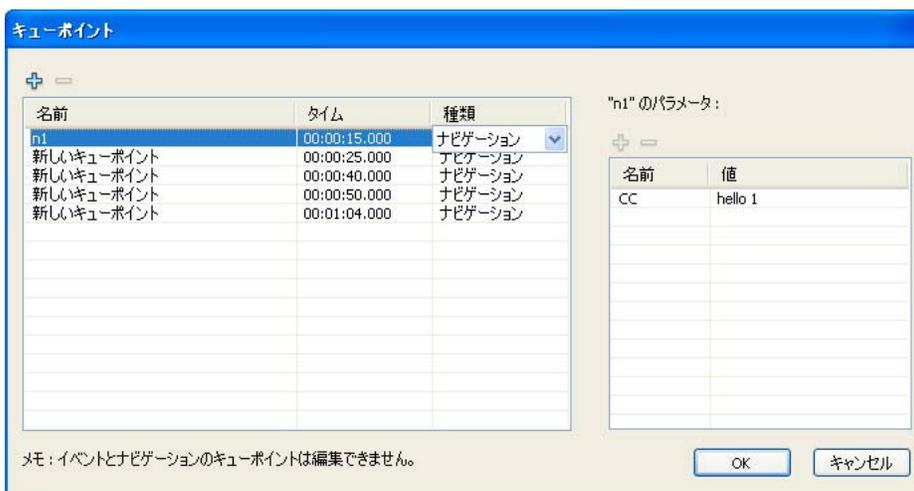
ActionScript キューポイントは、コンポーネントの Flash Video キューポイントダイアログボックスまたは FLVPlayback.addASCuePoint() メソッドを使用して追加できる外部キューポイントです。コンポーネントは FLV ファイルとは別に ActionScript キューポイントを格納および追跡します。その結果、最終的な正確さは埋め込みキューポイントよりも劣ります。ActionScript キューポイントは、1/10 秒単位までは正確です。ActionScript キューポイントの精度を上げるには、playheadUpdateInterval プロパティの値を小さくします。これは、再生ヘッドが更新されるたびに、コンポーネントによって ActionScript キューポイントに対して cuePoint イベントが生成されるからです。詳しくは、「**Adobe Flash Professional CS5** 用 **ActionScript 3.0** リファレンスガイド」の FLVPlayback.playheadUpdateInterval プロパティを参照してください。

ActionScript および FLV ファイルのメタデータ内では、キューポイントは name、time、type および parameters の各プロパティを持つオブジェクトとして表されます。name プロパティは、キューポイントに割り当てられた名前を含んでいるストリングです。time プロパティは、キューポイントが発生する時、分、秒、およびミリ秒 (HH:MM:SS.mmm) を表す数値です。type プロパティは、作成したキューポイントの種類に応じて値が "navigation"、"event" または "actionscript" となるストリングです。parameters プロパティは、指定した名前と値のペアの配列です。

cuePoint イベントが発生すると、info プロパティを通じてイベントオブジェクトでキューポイントオブジェクトを使用できるようになります。

Flash Video キューポイントダイアログボックスの使用

Flash Video キューポイントダイアログボックスを開くには、コンポーネントインスペクターで cuePoints パラメーターの「値」セルをダブルクリックします。このダイアログボックスの外観を次の図に示します。



キューポイントダイアログボックス

このダイアログボックスには、埋め込みキューポイントおよび ActionScript キューポイントが表示されます。このダイアログボックスを使用すると、ActionScript キューポイントおよびキューポイントパラメーターの追加と削除ができます。埋め込みキューポイントを有効または無効にすることもできます。ただし、埋め込みキューポイントを追加、変更、または削除することはできません。

ActionScript キューポイントの追加：

- 1 コンポーネントインスペクターで cuePoints パラメーターの「値」セルをダブルクリックし、Flash Video キューポイントダイアログボックスを開きます。
- 2 キューポイントのリストの上の左上隅にあるプラス記号 (+) をクリックして、デフォルトの ActionScript キューポイント項目を追加します。
- 3 「名前」列の「新しいキューポイント」というテキストをクリックし、テキストを編集してキューポイントに名前を付けます。
- 4 時間値「00:00:00:000」をクリックして編集し、キューポイントが発生する時間を割り当てます。時間は、時、分、秒、およびミリ秒 (HH:MM:SS.mmm) で指定します。
複数のキューポイントが存在する場合、新しいキューポイントはリスト内で時系列順に移動されます。
- 5 選択したキューポイントのパラメーターを追加するには、「パラメーター」セクションの上のあるプラス記号 (+) をクリックして、「名前」列および「値」列に値を入力します。パラメーターごとに、この手順を繰り返します。
- 6 さらに多くの ActionScript キューポイントを追加するには、パラメーターごとに手順 2 ~ 5 を繰り返します。
- 7 「OK」をクリックして変更を保存します。

ActionScript キューポイントの削除：

- 1 コンポーネントインスペクターで cuePoints パラメーターの「値」セルをダブルクリックし、Flash キューポイントダイアログボックスを開きます。
- 2 削除するキューポイントを選択します。
- 3 キューポイントのリストの上の左上隅にあるマイナス記号 (-) をクリックして、キューポイントを削除します。
- 4 削除するキューポイントごとに手順 2 および 3 を繰り返します。
- 5 「OK」をクリックして変更を保存します。

FLV ファイルの埋め込みキューポイントを有効または無効にするには：

- 1 コンポーネントインスペクターで cuePoints パラメーターの「値」セルをダブルクリックし、Flash キューポイントダイアログボックスを開きます。
- 2 有効または無効にするキューポイントを選択します。
- 3 「タイプ」列で値をクリックしてポップアップメニューを開くか、下矢印ボタンをクリックします。
- 4 キューポイントを有効にするには、そのタイプの名前 (例えば「イベント」または「ナビゲーション」) をクリックします。キューポイントを無効にするには、「無効」をクリックします。
- 5 「OK」をクリックして変更を保存します。

ActionScript でのキューポイントの使用

ActionScript を使用すると、ActionScript キューポイントの追加、cuePoint イベントのリスン、任意のタイプまたは特定のタイプのキューポイントの検索、ナビゲーションキューポイントのシーク、キューポイントの有効化または無効化、キューポイントが有効であるかどうかの確認、およびキューポイントの削除を行うことができます。

この節の例では、「cuepoints.flv」という名前の FLV ファイルを使用します。このファイルには、次の 3 つのキューポイントが含まれています。

名前	時刻	型
point1	00:00:00.418	ナビゲーション
point2	00:00:07.748	ナビゲーション
point3	00:00:16.020	ナビゲーション

ActionScript キューポイントの追加

ActionScript キューポイントを FLV ファイルに追加するには、`addASCuePoint()` メソッドを使用します。次の例では、再生の準備ができたときに 2 つの ActionScript キューポイントを FLV ファイルに追加します。キューポイントオブジェクトを使用して最初のキューポイントを追加します。キューポイントのプロパティで、その時間、名前、およびタイプを指定します。2 番目の呼び出しでは、メソッドの `time` パラメーターおよび `name` パラメーターを使用して時間と名前を指定します。

```
// Requires an FLVPlayback instance called my_FLVPlaybk on Stage
import fl.video.*;
import fl.video.MetadataEvent;
my_FLVPlaybk.source = "http://www.helpexamples.com/flash/video/cuepoints.flv"
var cuePt:Object = new Object(); //create cue point object
cuePt.time = 2.02;
cuePt.name = "ASpt1";
cuePt.type = "actionscript";
my_FLVPlaybk.addASCuePoint(cuePt);//add AS cue point
// add 2nd AS cue point using time and name parameters
my_FLVPlaybk.addASCuePoint(5, "ASpt2");
```

詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の `FLVPlayback.addASCuePoint()` メソッドを参照してください。

cuePoint イベントのリスン

`cuePoint` イベントを使用すると、`cuePoint` イベントが発生したときに、ActionScript コードで制御を受け取ることができません。次の例でキューポイントが発生したときに、`cuePoint` リスナーは `playheadTime` プロパティの値およびキューポイントの名前とタイプを表示するイベントハンドラー関数を呼び出します。前の節の「ActionScript キューポイントの追加」の例と共にこの例を使用して、結果を確認してください。

```
my_FLVPlaybk.addEventListener(MetadataEvent.CUE_POINT, cp_listener);
function cp_listener(eventObject:MetadataEvent):void {
    trace("Elapsed time in seconds: " + my_FLVPlaybk.playheadTime);
    trace("Cue point name is: " + eventObject.info.name);
    trace("Cue point type is: " + eventObject.info.type);
}
```

`cuePoint` イベントについて詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の `FLVPlayback.cuePoint` イベントを参照してください。

キューポイントの検索

ActionScript を使用すると、任意のタイプのキューポイントの検索、時間に対して最も近いキューポイントの検索、または指定した名前を持つ次のキューポイントの検索を行うことができます。

次の例の `ready_listener()` イベントハンドラーは、`findCuePoint()` メソッドを呼び出してキューポイント `ASpt1` を検索し、`findNearestCuePoint()` メソッドを呼び出して、キューポイント `ASpt1` の時間に最も近いナビゲーションキューポイントを検索します。

```
import fl.video.FLVPlayback;
import fl.video.CuePointType;
import fl.video.VideoEvent;
my_FLVPlybk.source = "http://www.helpexamples.com/flash/video/cuepoints.flv"
var rtn_obj:Object; //create cue point object
my_FLVPlybk.addASCuePoint(2.02, "ASpt1");//add AS cue point
function ready_listener(eventObject:VideoEvent):void {
    rtn_obj = my_FLVPlybk.findCuePoint("ASpt1", CuePointType.ACTIONSCRIPT);
    traceit(rtn_obj);
    rtn_obj = my_FLVPlybk.findNearestCuePoint(rtn_obj.time, CuePointType.NAVIGATION);
    traceit(rtn_obj);
}
my_FLVPlybk.addEventListener(VideoEvent.READY, ready_listener);
function traceit(cuePoint:Object):void {
    trace("Cue point name is: " + cuePoint.name);
    trace("Cue point time is: " + cuePoint.time);
    trace("Cue point type is: " + cuePoint.type);
}
```

次の例では、`ready_listener()` イベントハンドラーはキューポイント `ASpt` を検索し、`findNextCuePointWithName()` メソッドを呼び出して、同じ名前を持つ次のキューポイントを検索します。

```
import fl.video.*;
my_FLVPlybk.source = "http://www.helpexamples.com/flash/video/cuepoints.flv"
var rtn_obj:Object; //create cue point object
my_FLVPlybk.addASCuePoint(2.02, "ASpt");//add AS cue point
my_FLVPlybk.addASCuePoint(3.4, "ASpt");//add 2nd ASpt
my_FLVPlybk.addEventListener(VideoEvent.READY, ready_listener);
function ready_listener(eventObject:VideoEvent):void {
    rtn_obj = my_FLVPlybk.findCuePoint("ASpt", CuePointType.ACTIONSCRIPT);
    traceit(rtn_obj);
    rtn_obj = my_FLVPlybk.findNextCuePointWithName(rtn_obj);
    traceit(rtn_obj);
}
function traceit(cuePoint:Object):void {
    trace("Cue point name is: " + cuePoint.name);
    trace("Cue point time is: " + cuePoint.time);
    trace("Cue point type is: " + cuePoint.type);
}
```

キューポイントの検索方法について詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の `FLVPlayback.findCuePoint()`、`FLVPlayback.findNearestCuePoint()`、および `FLVPlayback.findNextCuePointWithName()` の各メソッドを参照してください。

ナビゲーションキューポイントのシーク

ナビゲーションキューポイントのシーク、指定した時間から後の次のナビゲーションキューポイントのシーク、および指定した時間から前のナビゲーションキューポイントのシークを行うことができます。次の例では、FLV ファイル `cuepoints.flv` を再生し、`ready` イベントが発生したときに、7.748 にあるキューポイントをシークします。`cuePoint` イベントが発生したときに、`seekToPrevNavCuePoint()` メソッドを呼び出して最初のキューポイントをシークします。`cuePoint` イベントが発生したとき、`seekToNextNavCuePoint()` メソッドを呼び出して、現在のキューポイントの時間である `eventObject.info.time` に 10 秒を追加して、最後のキューポイントをシークします。

```
import fl.video.*;

my_FLVPlybk.addEventListener(VideoEvent.READY, ready_listener);
function ready_listener(eventObject:Object):void {
    my_FLVPlybk.seekToNavCuePoint("point2");
}
my_FLVPlybk.addEventListener(MetadataEvent.CUE_POINT, cp_listener);
function cp_listener(eventObject:MetadataEvent):void {
    trace(eventObject.info.time);
    if(eventObject.info.time == 7.748)
        my_FLVPlybk.seekToPrevNavCuePoint(eventObject.info.time - .005);
    else
        my_FLVPlybk.seekToNextNavCuePoint(eventObject.info.time + 10);
}
my_FLVPlybk.source = "http://helpexamples.com/flash/video/cuepoints.flv";
```

詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の `FLVPlayback.seekToNavCuePoint()`、`FLVPlayback.seekToNextNavCuePoint()`、および `FLVPlayback.seekToPrevNavCuePoint()` の各メソッドを参照してください。

FLV ファイルの埋め込みキューポイントの有効化と無効化

FLV ファイルの埋め込みキューポイントを有効または無効にするには、`setFLVCuePointEnabled()` メソッドを使用します。無効になっているキューポイントは、`cuePoint` イベントをトリガーせず、`seekToCuePoint()`、`seekToNextNavCuePoint()` および `seekToPrevNavCuePoint()` の各メソッドに対して機能しません。ただし、`findCuePoint()`、`findNearestCuePoint()` および `findNextCuePointWithName()` の各メソッドを使用して、無効になっているキューポイントを検索できます。

FLV ファイルの埋め込みキューポイントが有効になっているかどうかは、`isFLVCuePointEnabled()` メソッドを使用してテストできます。次の例では、ビデオの再生準備ができたときに、埋め込みキューポイント `point2` および `point3` を無効にします。ただし、最初の `cuePoint` イベントが発生すると、イベントハンドラーはキューポイント `point3` が無効になっているかどうかを確認し、無効の場合はそれを有効にします。

```
import fl.video.*;
my_FLVPlybk.source = "http://www.helpexamples.com/flash/video/cuepoints.flv";
my_FLVPlybk.addEventListener(VideoEvent.READY, ready_listener);
function ready_listener(eventObject:VideoEvent):void {
    my_FLVPlybk.setFLVCuePointEnabled(false, "point2");
    my_FLVPlybk.setFLVCuePointEnabled(false, "point3");
}
my_FLVPlybk.addEventListener(MetadataEvent.CUE_POINT, cp_listener);
function cp_listener(eventObject:MetadataEvent):void {
    trace("Cue point time is: " + eventObject.info.time);
    trace("Cue point name is: " + eventObject.info.name);
    trace("Cue point type is: " + eventObject.info.type);
    if (my_FLVPlybk.isFLVCuePointEnabled("point2") == false) {
        my_FLVPlybk.setFLVCuePointEnabled(true, "point2");
    }
}
}
```

詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の `FLVPlayback.isFLVCuePointEnabled()` メソッドおよび `FLVPlayback.setFLVCuePointEnabled()` メソッドを参照してください。

ActionScript キューポイントの削除

ActionScript キューポイントを削除するには、`removeASCuePoint()` メソッドを使用します。次の例では、キューポイント `ASpt1` が発生したときに、キューポイント `ASpt2` を削除します。

```
import fl.video.*;
my_FLVPlayback.source = "http://www.helpexamples.com/flash/video/cuepoints.flv"
my_FLVPlayback.addASCuePoint(2.02, "ASpt1");//add AS cue point
my_FLVPlayback.addASCuePoint(3.4, "ASpt2");//add 2nd Aspt
my_FLVPlayback.addEventListener(MouseEvent.CLICK, cp_listener);
function cp_listener(eventObject:MouseEvent):void {
    trace("Cue point name is: " + eventObject.info.name);
    if (eventObject.info.name == "ASpt1") {
        my_FLVPlayback.removeASCuePoint("ASpt2");
        trace("Removed cue point ASpt2");
    }
}
```

詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の `FLVPlayback.removeASCuePoint()` を参照してください。

複数のビデオファイルの再生

FLVPlayback インスタンスでビデオファイルを連続して再生するには、前のビデオファイルの再生が終了したときに、`source` プロパティに新しい URL をロードします。例えば、次の ActionScript コードでは、`complete` イベントをリスンします。このイベントは、ビデオファイルが再生を終了したときに発生します。このイベントが発生したら、新しいビデオファイルの名前と場所を `source` プロパティに設定し、`play()` メソッドを呼び出して新しいビデオを再生します。

```
import fl.video.*;
my_FLVPlayback.source = "http://www.helpexamples.com/flash/video/clouds.flv";
my_FLVPlayback.addEventListener(Event.COMPLETE, complete_listener);
// listen for complete event; play new FLV
function complete_listener(eventObject:Event):void {
    if (my_FLVPlayback.source == "http://www.helpexamples.com/flash/video/clouds.flv") {
        my_FLVPlayback.play("http://www.helpexamples.com/flash/video/water.flv");
    }
};
```

複数のビデオプレーヤーの使用

FLVPlayback コンポーネントの 1 つのインスタンス内で複数のビデオプレーヤーを開き、複数のビデオを再生して、再生中にそれらを切り替えることもできます。

FLVPlayback コンポーネントをステージにドラッグすると、最初のビデオプレーヤーを作成できます。最初のビデオプレーヤーには自動的に数値 0 が割り当てられ、これがデフォルトプレーヤーとなります。追加のビデオプレーヤーを作成するには、`activeVideoPlayerIndex` プロパティを新しい数値に設定します。`activeVideoPlayerIndex` プロパティを設定すると、指定したビデオプレーヤーがアクティブなビデオプレーヤーになります。これは、FLVPlayback クラスのプロパティとメソッドによって制御されるビデオプレーヤーです。ただし、`activeVideoPlayerIndex` プロパティを設定しても、ビデオプレーヤーは表示されません。ビデオプレーヤーを表示するには、`visibleVideoPlayerIndex` プロパティにビデオプレーヤーの番号を設定します。これらのプロパティと、FLVPlayback クラスのメソッドとプロパティとの関連について詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の `FLVPlayback.activeVideoPlayerIndex` プロパティおよび `FLVPlayback.visibleVideoPlayerIndex` プロパティを参照してください。

次の ActionScript コードでは、`source` プロパティをロードしてデフォルトのビデオプレーヤーでビデオファイルを再生し、そのキューポイントを追加します。`ready` イベントが発生したら、イベントハンドラーは `activeVideoPlayerIndex` プロパティを値 1 に設定して、2 番目のビデオプレーヤーを開きます。イベントハンドラーは、2 番目のビデオプレーヤーの FLV ファイルとキューポイントを指定して、デフォルトのプレーヤー (0) を再びアクティブなビデオプレーヤーにします。

```

/**
    Requires:
    - FLVPlayback component on the Stage with an instance name of my_FLVPlybk
*/
// add a cue point to the default player
import fl.video.*;
my_FLVPlybk.source = "http://www.helpexamples.com/flash/video/clouds.flv";
my_FLVPlybk.addASCuePoint(3, "1st_switch");
my_FLVPlybk.addEventListener(VideoEvent.READY, ready_listener);
function ready_listener(eventObject:VideoEvent):void {
    // add a second video player and create a cue point for it
    my_FLVPlybk.activeVideoPlayerIndex = 1;
    my_FLVPlybk.source = "http://www.helpexamples.com/flash/video/water.flv";
    my_FLVPlybk.addASCuePoint(3, "2nd_switch");
    my_FLVPlybk.activeVideoPlayerIndex = 0;
};

```

1 つの FLV ファイルが再生中に別の FLV ファイルに切り替えるには、ActionScript コードで切り替えを行う必要があります。キューポイントを使用すると、cuePoint イベントを使用して、FLV ファイルで特定のポイントに介入することができます。次のコードでは、cuePoint イベントのリスナーを作成します。次に、アクティブなビデオプレーヤー (0) を一時停止し、2 番目のプレーヤー (1) に切り替え、その FLV ファイルを再生するハンドラー関数を呼び出します。

```

import fl.video.*;
// add listener for a cuePoint event
my_FLVPlybk.addEventListener(MetadataEvent.CUE_POINT, cp_listener);
// add the handler function for the cuePoint event
function cp_listener(eventObject:MetadataEvent):void {
    // display the no. of the video player causing the event
    trace("Hit cuePoint event for player: " + eventObject.vp);
    // test for the video player and switch FLV files accordingly
    if (eventObject.vp == 0) {
        my_FLVPlybk.pause(); //pause the first FLV file
        my_FLVPlybk.activeVideoPlayerIndex = 1; // make the 2nd player active
        my_FLVPlybk.visibleVideoPlayerIndex = 1; // make the 2nd player visible
        my_FLVPlybk.play(); // begin playing the new player/FLV
    } else if (eventObject.vp == 1) {
        my_FLVPlybk.pause(); // pause the 2nd FLV
        my_FLVPlybk.activeVideoPlayerIndex = 0; // make the 1st player active
        my_FLVPlybk.visibleVideoPlayerIndex = 0; // make the 1st player visible
        my_FLVPlybk.play(); // begin playing the 1st player
    }
}
my_FLVPlybk.addEventListener(VideoEvent.COMPLETE, complete_listener);
function complete_listener(eventObject:VideoEvent):void {
    trace("Hit complete event for player: " + eventObject.vp);
    if (eventObject.vp == 0) {
        my_FLVPlybk.activeVideoPlayerIndex = 1;
        my_FLVPlybk.visibleVideoPlayerIndex = 1;
        my_FLVPlybk.play();
    } else {
        my_FLVPlybk.closeVideoPlayer(1);
    }
};

```

新しいビデオプレーヤーを作成すると、FLVPlayback インスタンスはそのプロパティにデフォルトビデオプレーヤーの値を設定します。ただし、source、totalTime および isLive の各プロパティは例外であり、これらに関しては常にデフォルト値である空の文字列、0 および false にそれぞれ設定されます。また autoPlay プロパティも設定されます。このプロパティはデフォルトで true であり、デフォルトのビデオプレーヤーの場合は false になります。cuePoints プロパティには効果がなく、デフォルトビデオプレーヤーへのそれ以降のロードに対して無効です。

ボリューム、位置、サイズ、可視性およびユーザーインターフェイスコントロールを制御するメソッドとプロパティは常にグローバルであり、そのビヘイビアは `activeVideoPlayerIndex` プロパティを設定しても影響を受けません。これらのメソッドおよびプロパティと、`activeVideoPlayerIndex` プロパティを設定した影響については、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の `FLVPlayback.activeVideoPlayerIndex` プロパティを参照してください。その他のプロパティとメソッドは、`activeVideoPlayerIndex` プロパティの値で指定されたビデオプレーヤーをターゲットとします。

ただし、サイズを制御するプロパティとメソッドは、`visibleVideoPlayerIndex` プロパティによって影響を受けます。詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の `FLVPlayback.visibleVideoPlayerIndex` プロパティを参照してください。

Flash Media Server からの FLV ファイルのストリーミング

Flash Media Server から FLV ファイルをストリーミング再生するための要件は、Flash Video Streaming Service プロバイダーからネイティブの帯域幅検出を利用できるかどうかによって異なります。ネイティブの帯域幅検出とは、帯域幅検出がストリーミングサーバーに組み込まれているということであり、パフォーマンスに優れています。ネイティブの帯域幅検出を利用できるかどうかについては、プロバイダーに確認してください。

Flash Media Server 上の FLV ファイルにアクセスするには、`rtmp://my_servername/my_application/stream.flv` のような URL を使用します。

Flash Media Server を使用してライブストリームを再生する場合は、`FLVPlayback` の `isLive` プロパティを `true` に設定する必要があります。詳しくは、「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の `FLVPlayback.isLive` プロパティを参照してください。

ライブストリームの設定方法を含む Flash Media Server の管理については、Flash Media Server のドキュメント (www.adobe.com/support/documentation/jp/flashmediaserver/) を参照してください。

ネイティブの帯域幅検出の場合と帯域幅検出なしの場合

`NCManagerNative` クラスは `NCManager` のサブクラスで、一部の Flash Video Streaming Service プロバイダーがサポートするネイティブな帯域幅検出に対応しています。`NCManagerNative` を使用するとき、Flash Media Server 上に特別なファイルは不要です。また、`NCManagerNative` では、帯域幅検出が不要な場合、どのバージョンの Flash Media Server にも `main.asc` ファイルなしで接続できます。

デフォルトの `NCManager` クラスの代わりに `NCManagerNative` を使用するには、FLA ファイルの最初のフレームに次のコードを追加します。

```
import fl.video*;  
VideoPlayer.inCMManagerClass = fl.video.NCManagerNative;
```

非ネイティブの帯域幅検出の場合

Flash Video Streaming Service プロバイダーでネイティブの帯域幅検出を利用できないが、帯域幅検出が必要な場合には、Flash Media Server の FLV アプリケーションに `main.asc` ファイルを追加する必要があります。`main.asc` ファイルは www.adobe.com/go/learn_fl_samples_jp からオンラインで取得できます。`Samples¥ComponentsAS2¥FLVPlayback` ディレクトリにある `Samples.zip` ファイルに含まれています。

FLV ファイルをストリーミングするため Flash Media Server をセットアップするには

- 1 Flash Media Server アプリケーションフォルダー内にフォルダーを作成し、`my_application` などの名前を付けます。
- 2 "main.asc" ファイルを "my_application" フォルダーにコピーします。
- 3 `streams` という名前のフォルダーを `my_application` フォルダー内に作成します。
- 4 `_definst_` という名前のフォルダーを `streams` フォルダー内に作成します。

5 FLV ファイルを `_definst_` フォルダーに配置します。

FLVPlayback コンポーネントのカスタマイズ

ここでは、FLVPlayback コンポーネントをカスタマイズする方法について説明します。なお、他のコンポーネントのカスタマイズに使用されるほとんどの方法は、FLVPlayback コンポーネントでは機能しません。FLVPlayback コンポーネントをカスタマイズするには、この節で説明しているテクニックのみを使用します。

FLVPlayback コンポーネントのカスタマイズでは、デザイン済みスキンの選択、FLV Playback カスタム UI コンポーネントへの個々のスキンの適用、または新規スキンの作成の選択肢があります。FLVPlayback プロパティを使用してスキンのビヘイビアを変更することもできます。

注意：スキンが FLVPlayback コンポーネントで機能するためには、スキン SWF ファイルをアプリケーション SWF ファイルと共に Web サーバーにアップロードする必要があります。

デザイン済みスキンの選択

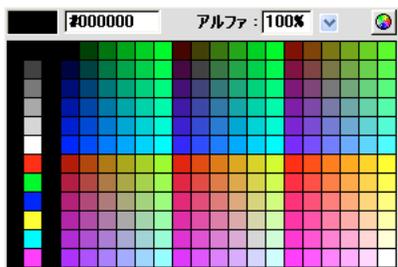
FLVPlayback コンポーネントのスキンを選択するには、コンポーネントインスペクターで `skin` パラメーターの「値」セルをクリックします。次に、拡大鏡アイコンをクリックして、次に示すスキンの選択ダイアログボックスを開きます。このダイアログボックスでは、スキンを選択するか、スキン SWF ファイルの場所を示す URL を指定します。



FLVPlayback のスキンの選択ダイアログボックス

スキンポップアップメニューに表示されるスキンは、Flash アプリケーションフォルダー \Flash Configuration\FLVPlayback Skins\ActionScript 3.0 ディレクトリにあります。このダイアログボックスに使用できる新しいスキンを作成するには、スキンを作成して SWF ファイルをこのフォルダーに配置します。名前が `.swf` 拡張子と共にポップアップメニューに表示されます。スキンセットの作成方法について詳しくは、152 ページの「[スキンの新規作成](#)」を参照してください。

`skin` プロパティを設定して割り当てるスキン（オーサリング時に `skin` パラメーターを設定するか、または実行時に ActionScript で割り当てるスキン）については、スキンの選択とは別に、色およびアルファ値（透明度）を割り当てることができます。オーサリング時に色およびアルファ値を割り当てるには、スキンの選択ダイアログボックスで、次のようなカラーピッカーを開きます。



スキンの選択ダイアログボックスのカラーピッカー

色を選択するには、パネルの色見本をクリックするか、またはテキストボックスに数値を入力します。アルファ値を選択するには、スライダーを使用するか、「アルファ」テキストボックスにパーセントの値を入力します。

実行時に色およびアルファ値を割り当てるには、`skinBackgroundColor` プロパティと `skinBackgroundAlpha` プロパティを設定します。`skinBackgroundColor` プロパティは `0xRRGGBB` (赤、緑、青) の値に設定します。`skinBackgroundAlpha` プロパティは、`0.0` ~ `1.0` の範囲の値に設定します。次の例では、`skinBackgroundColor` を `0xFF0000` (赤) に設定し、`skinBackgroundAlpha` を `0.5` に設定しています。

```
my_FLVPlaybk.skinBackgroundColor = 0xFF0000;  
my_FLVPlaybk.skinBackgroundAlpha = .5;
```

デフォルト値は、ユーザーが最後に選択した値です。

FLV Playback カスタム UI コンポーネントを使用して FLVPlayback コンポーネントにスキンを適用する場合は、ポップアップメニューから「なし」を選択します。

FLV Playback カスタム UI コンポーネントへの個々のスキンの適用

FLV Playback カスタム UI コンポーネントを使用すると、FLA ファイル内で FLVPlayback コントロールの外観をカスタマイズし、Web ページをプレビューしたときに結果を表示することができます。ただし、これらのコンポーネントは、拡大/縮小するように設計されていません。ムービークリップとそのコンテンツは特定のサイズとなるように編集する必要があります。このため、通常は FLVPlayback コンポーネントを目的のサイズでステージに配置し、`scaleMode` を `exactFit` に設定することをお勧めします。

最初に、コンポーネントパネルから FLV Playback カスタム UI コンポーネントをドラッグし、ステージ上の目的の場所に配置して、インスタンス名を付けます。

これらのコンポーネントは `ActionScript` なしでも機能します。これらを FLVPlayback コンポーネントと同じタイムラインおよびフレームに配置し、FLVPlayback コンポーネントにスキンが設定されていない場合、FLVPlayback コンポーネントはこれらのコンポーネントに自動的に接続します。ステージ上に複数の FLVPlayback コンポーネントがある場合、またはカスタムコントロールと FLVPlayback インスタンスが同じタイムラインにない場合には、`ActionScript` が必要です。

ステージにコンポーネントを配置したら、他のシンボルと同じように編集します。コンポーネントを開くと、各コンポーネントの設定は他のコンポーネントと少し異なることがわかります。

ボタンコンポーネント

ボタンコンポーネントは同じような構造を持っています。ボタンには、`BackButton`、`ForwardButton`、`MuteButton`、`PauseButton`、`PlayButton`、`PlayPauseButton`、および `StopButton` があります。ほとんどのボタンには、`placeholder_mc` というインスタンス名を持つフレーム 1 に 1 つのムービークリップがあります。通常、これはボタンの通常状態のインスタンスですが、必ずしもそうであるとは限りません。フレーム 2 では、各表示状態 (通常、オーバー、ダウン、無効) に対して、ステージ上に 4 つのムービークリップがあります。実行時には、コンポーネントが実際にフレーム 2 に移動

することはありません。ムービークリップがここに配置されるのは、編集を簡単にし、シンボルプロパティダイアログボックスの「最初のフレームに書き出し」チェックボックスをオンにすることなく SWF ファイルにロードするためです。ただし、「最初のフレームに書き出し」チェックボックスはオンにする必要があります。

ボタンにスキンを適用するには、各ムービークリップを編集します。そのサイズと外観を変更することができます。

通常、一部の ActionScript はフレーム 1 に表示されます。このスクリプトを編集する必要はありません。フレーム 1 で再生ヘッドが停止され、どのムービークリップをどの状態で使用するのかが指定されます。

PlayPauseButton ボタン、MuteButton ボタン、FullScreenButton ボタン、および CaptionButton ボタン

PlayPauseButton ボタン、MuteButton ボタン、FullScreenButton ボタン、および CaptionButton ボタンは、他のボタンとは設定が異なり、これらのボタンには、2つのレイヤーがありスクリプトのない1つのフレームのみが含まれます。このフレームには2つのボタンがあり、上下に重ねて配置されています。PlayPauseButton の場合は「再生」ボタンと「一時停止」ボタン、MuteButton の場合は「ミュートオン」ボタンと「ミュートオフ」ボタン、FullScreenButton の場合は「フルスクリーンオン」ボタンと「フルスクリーンオフ」ボタン、CaptionButton の場合は「キャプションオン」ボタンと「キャプションオフ」ボタンです。これらのボタンにスキンを適用するには、147ページの「[FLV Playback カスタム UI コンポーネントへの個々のスキンの適用](#)」の説明に従って、これら2つの内部ボタンそれぞれにスキンを適用します。それ以外の操作は不要です。

CaptionButton は FLVPlaybackCaptioning コンポーネント用であり、このコンポーネントに接続する必要があります。FLVPlayback コンポーネントに接続しないでください。

BackButton ボタンおよび ForwardButton ボタン

BackButton ボタンおよび ForwardButton ボタンも、他のボタンとは設定が異なります。フレーム 2 には、1つまたは両方のボタンの周囲のフレームとして使用できる追加のムービークリップがあります。このムービークリップは必須ではなく、特別な機能はありません。便宜上、用意されています。このムービークリップを使用するには、ライブラリパネルからステージにドラッグし、目的の場所に配置します。このムービークリップが不要な場合は、使用しないか、ライブラリパネルから削除します。

用意されているほとんどのボタンは、共通のムービークリップセットに基づいているので、すべてのボタンの外観を一度に変更することができます。この機能を使用するか、共通クリップを置き換えて、すべてのボタンの外観を変更できます。

BufferingBar コンポーネント

バッファリングバーコンポーネントは単純です。このコンポーネントは、コンポーネントがバッファリング状態に移行したときに表示されるアニメーションで構成されており、設定するのに特別な ActionScript は必要ありません。デフォルトは、矩形のマスクを持ち左から右に移動するストライプのバーで、「理髪店の看板柱」のようなエフェクトですが、この設定について特に説明の必要なことはありません。

スキン SWF ファイルのバッファリングバーでは、実行時に拡大 / 縮小する必要があるため 9 スライスの拡大 / 縮小が使用されますが、BufferingBar FLV カスタム UI コンポーネントにはネストされたムービークリップが含まれるため、9 スライスの拡大 / 縮小を使用することはできません。BufferingBar の幅や高さを大きくする場合は、拡大するのではなくコンテンツを変更してください。

SeekBar コンポーネントおよび VolumeBar コンポーネント

SeekBar コンポーネントと VolumeBar コンポーネントは似ていますが、機能は異なります。各コンポーネントはハンドルを備え、同じハンドル追跡メカニズムを使用し、内部にネストされたクリップの再生状況とロード状況を追跡する機能をサポートしています。

FLVPlayback コンポーネントの ActionScript コードでは、SeekBar コンポーネントまたは VolumeBar コンポーネントの基準点（原点またはゼロ点）はコンテンツの左上隅にあると見なされる場合が多いので、この規則を守ることが重要です。守らない場合、ハンドル、再生状況ムービークリップ、およびロード状況ムービークリップに関して問題が発生する可能性があります。

スキン SWF ファイルのシークバーでは、実行時に拡大 / 縮小する必要があるので 9 スライスの拡大 / 縮小が使用されますが、SeekBar FLV カスタム UI コンポーネントにはネストされたムービークリップが含まれるので、9 スライスの拡大 / 縮小を使用することはできません。SeekBar の幅や高さを大きくする場合は、拡大するのではなくコンテンツを変更してください。

ハンドル

ハンドルムービークリップのインスタンスはフレーム 2 にあります。BackButton コンポーネントおよび ForwardButton コンポーネントと同様に、コンポーネントが実際にフレーム 2 に移動することはありません。ムービークリップがここに配置されるのは、編集を簡単にし、シンボルプロパティダイアログボックスの「最初のフレームに書き出し」チェックボックスをオンにすることなく SWF ファイルにロードするためです。ただし、「最初のフレームに書き出し」チェックボックスはオンにする必要があります。

ハンドルムービークリップの背景に、アルファが 0 に設定された矩形がある場合があります。この矩形によって、ハンドルのヒット領域が大きくなるので、ボタンのヒット状態のように、その外観を変えずにハンドルがつかみやすくなります。ハンドルは実行時に動的に作成されるので、ボタンではなくムービークリップである必要があります。アルファが 0 に設定されたこの矩形を必要とする他の理由はなく、通常、ハンドルの内部は任意のイメージで置き換えることができます。ただし、ハンドルムービークリップの中央で基準点を水平方向の中央に維持することをお勧めします。

次の ActionScript コードは SeekBar コンポーネントのフレーム 1 にあり、ハンドルを管理します。

```
stop();  
handleLinkageID = "SeekBarHandle";  
handleLeftMargin = 2;  
handleRightMargin = 2;  
handleY = 11;
```

stop() 関数の呼び出しは、フレーム 2 のコンテンツのために必要です。

2 行目は、ハンドルとして使用するシンボルを指定します。フレーム 2 のハンドルムービークリップインスタンスのみを編集する場合、この指定を変更する必要はありません。FLVPlayback コンポーネントは実行時に、指定したムービークリップのインスタンスを Bar コンポーネントインスタンスの子としてステージ上に作成します。これにより、同じ親ムービークリップを持つことになります。そのため、バーがルートレベルにある場合は、ハンドルもルートレベルにある必要があります。

変数 handleLeftMargin はハンドルの元の場所 (0%) を指定し、変数 handleRightMargin は最後の場所 (100%) を指定します。数値はバーコントロールの左右の端からのオフセットを示し、正の数値はバー内部に限度をマークし、負の数値はバー外部に限度をマークします。これらのオフセットは、基準点に基づいてハンドルが移動できる場所を示します。ハンドルの中央に基準点を配置した場合、ハンドルの左右の端はマージンを越えます。シークバームービークリップが正常に機能するには、そのコンテンツの左上隅に基準点がある必要があります。

変数 handleY は、バーインスタンスを基準にしたハンドルの y 位置を指定します。これは、各ムービークリップの基準点に基づいています。サンプルハンドルの基準点は、表示される部分に相対的に配置されるように三角形の頂点にあります。非表示のヒット状態矩形は無視されます。また、バームービークリップが正常に機能するには、そのコンテンツの左上隅に基準点を維持する必要があります。

そのため、これらの限度を設定すると、バーコントロールが (100, 100) に設定され、幅が 100 ピクセルである場合、ハンドルは水平方向では 102 ~ 198 の範囲となり、垂直方向では 111 の位置となります。handleLeftMargin および handleRightMargin を -2 に変更し、handleY を -11 に変更した場合、ハンドルの水平方向の範囲は 98 ~ 202 となり、垂直方向の位置は 89 となります。

再生状況ムービークリップとロード状況ムービークリップ

SeekBar コンポーネントには再生状況ムービークリップがあり、VolumeBar コンポーネントにはロード状況ムービークリップがあります。ただし、実際の個々の SeekBar コンポーネントや VolumeBar コンポーネントには、いずれか一方または両方のムービークリップがある場合も、どちらもない場合もあります。これらのムービークリップは構造的には同じであ

り、同じように動作しますが、異なる値を追跡します。再生状況ムービークリップは FLV ファイルがダウンロードされると塗りつぶされ (FMS からのストリーミングの場合は常に塗りつぶされるので、HTTP ダウンロードのみに有効です)、ロード状況ムービークリップはハンドルが左から右に移動すると塗りつぶされます。

FLVPlayback コンポーネントは特定のインスタンス名でこれらのムービークリップインスタンスを検索するので、再生状況ムービークリップインスタンスは、その親としてパームービークリップを持ち、インスタンス名が `progress_mc` である必要があります。ロード状況ムービークリップインスタンスのインスタンス名は `fullness_mc` である必要があります。

再生状況ムービークリップおよびロード状況ムービークリップは、内側にネストした `fill_mc` ムービークリップインスタンスを使用して設定することも、使用しないで設定することもできます。VolumeBar `fullness_mc` ムービークリップは、`fill_mc` ムービークリップを使用する方法を示し、SeekBar `progress_mc` ムービークリップは `fill_mc` ムービークリップを使用しない方法を示します。

内側にネストした `fill_mc` ムービークリップを使用する方法は、拡大 / 縮小すると外観が湾曲してしまう塗りを使用する場合に便利です。

VolumeBar `fullness_mc` ムービークリップでは、ネストされた `fill_mc` ムービークリップインスタンスがマスクされます。ムービークリップの作成時にマスクするか、または実行時に動的にマスクが作成されます。ムービークリップを使用してマスクする場合は、インスタンスに `mask_mc` という名前を付け、パーセンテージが 100% のときに `fill_mc` が表示されるように設定します。`fill_mc` をマスクしない場合、動的に作成されるマスクは矩形になり、`fill_mc` (100%) と同じサイズとなります。

`fill_mc` ムービークリップは、`fill_mc.slideReveal` が `true` または `false` のいずれであるかに応じて、2 つの方法のいずれかで表示されます。

`fill_mc.slideReveal` が `true` の場合、`fill_mc` は左から右に移動し、マスクを通じて表示されます。0% では左端に配置されるので、マスクを通じて何も表示されません。パーセンテージが増えると 100% になるまで右に移動し、ステージ上で作成された場所に戻ります。

`fill_mc.slideReveal` が `false` または `undefined` (デフォルトのビヘイビア) の場合、左から右に移動すると共にマスクのサイズが変わり、`fill_mc` のより多くの部分が表示されます。0% の場合、マスクは水平方向に 05 まで拡大され、パーセントが増えるにつれて、100% で `fill_mc` がすべて表示されるまで、`scaleX` は増大します。`mask_mc` が作成時に拡大 / 縮小されている可能性があるため、これは `scaleX = 100` であるとは限りません。

`fill_mc` を使用しない方法は `fill_mc` を使用する方法よりも簡単ですが、塗りは水平方向に湾曲します。この湾曲を望まない場合は、`fill_mc` を使用する必要があります。SeekBar `progress_mc` で、この方法を示します。

再生状況ムービークリップまたはロード状況ムービークリップは、パーセンテージに基づいて水平方向に拡大 / 縮小されます。0% では、インスタンスの `scaleX` は 0 に設定され、非表示になります。パーセンテージが大きくなると `scaleX` は調整され、100% で、クリップのサイズは作成時のステージ上でのサイズと同じになります。クリップインスタンスが作成時に拡大 / 縮小されている可能性があるため、これは `scaleX = 100` であるとは限りません。

FLV Playback カスタム UI コンポーネントの接続

カスタム UI コンポーネントを FLVPlayback コンポーネントと同じタイムラインおよびフレームに配置し、`skin` プロパティを設定していない場合、FLVPlayback はそれらのコンポーネントに自動的に接続され、ActionScript は必要ありません。

ステージ上に複数の FLVPlayback コンポーネントがある場合、またはカスタムコントロールと FLVPlayback が同じタイムラインにない場合には、ActionScript コードを記述して、カスタム UI コンポーネントを FLVPlayback コンポーネントのインスタンスに接続する必要があります。最初に、FLVPlayback インスタンスに名前を割り当て、次に ActionScript を使用して FLV Playback カスタム UI コンポーネントインスタンスに対応する FLVPlayback プロパティに割り当てる必要があります。次の例では、FLVPlayback インスタンスは `my_FLVPlybk` であり、ピリオド (.) の後に FLVPlayback プロパティ名が続きます。また、FLV Playback カスタム UI コントロールインスタンスは等号 (=) の右側になります。

```
//FLVPlayback instance = my_FLVPlaybk  
my_FLVPlaybk.playButton = playbtn; // set playButton prop. to playbtn, etc.  
my_FLVPlaybk.pauseButton = pausebtn;  
my_FLVPlaybk.playPauseButton = playpausebtn;  
my_FLVPlaybk.stopButton = stopbtn;  
my_FLVPlaybk.muteButton = mutebtn;  
my_FLVPlaybk.backButton = backbtn;  
my_FLVPlaybk.forwardButton = forbtn;  
my_FLVPlaybk.volumeBar = volbar;  
my_FLVPlaybk.seekBar = seekbar;  
my_FLVPlaybk.bufferingBar = bufbar;
```

次の手順では、StopButton、PlayPauseButton、MuteButton、および SeekBar の各コントロールを作成します。

- 1 FLVPlayback コンポーネントをステージにドラッグし、インスタンス名を **my_FLVPlaybk** にします。
- 2 コンポーネントインスペクターで source パラメーターを <http://www.helpexamples.com/flash/video/cuepoints.flv> に設定します。
- 3 Skin パラメーターを「None」に設定します。
- 4 StopButton、PlayPauseButton、および MuteButton をステージにドラッグし、FLVPlayback インスタンスの上に配置して、左側に垂直に重ねます。プロパティインスペクターで、各ボタンにインスタンス名を付けます (**my_stopbtn**、**my_playpausebtn**、**my_mutebtn** など)。
- 5 ライブラリパネルで "FLVPlayback Skins" フォルダーを開き、その下位にある "SquareButton" フォルダーを開きます。
- 6 SquareBgDown ムービークリップを選択し、ダブルクリックしてステージ上で開きます。
- 7 右クリック (Windows)、または Control キーを押しながらクリック (Macintosh) し、ショートカットメニューから「すべて選択」を選択してシンボルを削除します。
- 8 楕円ツールを選択し、同じ場所に楕円を描画して、塗りを青 (#0033FF) に設定します。
- 9 プロパティインスペクターで、「W」(幅) を **40** に設定し、「H」(高さ) を **20** に設定します。「X」(x 座標) を **0.0** に設定し、「Y」(y 座標) を **0.0** に設定します。
- 10 SquareBgNormal に対して手順 6 ~ 8 を繰り返します。ただし、塗りは黄色 (#FFFF00) に設定します。
- 11 SquareBgOver に対して手順 6 ~ 8 を繰り返します。ただし、塗りは緑 (#006600) に設定します。
- 12 ボタン内のさまざまなシンボルアイコン (PauseIcon、PlayIcon、MuteOnIcon、MuteOffIcon、および StopIcon) のムービークリップを編集します。これらのムービークリップは、ライブラリパネルで FLV Playback Skins < ラベル > Button \ Assets の下位にあります。ここで、< ラベル > は再生、一時停止などのボタン名を表します。それぞれについて次の手順を実行します。
 - a 「すべて選択」オプションを選択します。
 - b 色を赤 (#FF0000) に変更します。
 - c 300% に拡大します。
 - d コンテンツの X 位置を **7.0** に変更し、各ボタンの状態のアイコンの水平方向の配置を変更します。

注意: このように場所を変更することで、各ボタンの状態を開き、アイコンのムービークリップインスタンスを移動せずに済みます。
- 13 タイムラインの上にある青の戻る矢印をクリックして、シーン 1、フレーム 1 に戻ります。
- 14 SeekBar コンポーネントをステージにドラッグし、FLVPlayback インスタンスの右下隅に配置します。
- 15 ライブラリパネルで SeekBar をダブルクリックしてステージ上で開きます。
- 16 400% に拡大します。
- 17 アウトラインを選択し、色を赤 (#FF0000) に設定します。

- 18 FLVPlayback Skins\Seek Bar フォルダーで SeekBarProgress をダブルクリックして、色を黄色 (#FFFF00) に設定します。
- 19 FLVPlayback Skins\Seek Bar フォルダーで SeekBarHandle をダブルクリックして、色を赤 (#FF0000) に設定します。
- 20 タイムラインの上にある青の戻る矢印をクリックして、シーン 1、フレーム 1 に戻ります。
- 21 ステージ上の SeekBar インスタンスを選択し、インスタンス名を **my_seekbar** にします。
- 22 次の例に示すように、タイムラインのフレーム 1 のアクションパネルで、ビデオクラスの読み込みステートメントを追加し、対応する FLVPlayback プロパティにボタン名とシークバー名を割り当てます。

```
import fl.video.*;
my_FLVPlaybk.stopButton = my_stopbbtn;
my_FLVPlaybk.playPauseButton = my_plypausbbtn;
my_FLVPlaybk.muteButton = my_mutebbtn;
my_FLVPlaybk.seekBar = my_seekbar;
```
- 23 Ctrl + Enter を押してムービーをテストします。

スキンの新規作成

スキン SWF ファイルを作成する最善の方法は、Flash に付属のスキンファイルをコピーし、それを土台として使用することです。これらのスキンの FLA ファイルは、"Configuration/FLVPlayback Skins/FLA/ActionScript 3.0/" の Flash アプリケーションフォルダーにあります。完成したスキン SWF ファイルをスキンの選択ダイアログボックスのオプションとして使用できるようにするには、Configuration\FLVPlayback Skins\ActionScript 3.0 フォルダーにそのファイルを配置します。このフォルダーは、Flash アプリケーションフォルダー、またはユーザーのローカルの Configuration\FLVPlayback Skins\ActionScript 3.0 フォルダーにあります。

スキンの色はスキンの選択とは別個に設定できるので、FLA ファイルを編集して色を変更する必要はありません。特定の色を持つスキンを作成した場合に、スキンの選択ダイアログボックスで編集できないようにするには、スキンの FLA ファイルの ActionScript コードで `this.border_mc.colorMe = false;` に設定します。スキンの色の設定方法については、146 ページの「[デザイン済みスキンの選択](#)」を参照してください。

インストールされた Flash スキンの FLA ファイルを見ると、ステージ上の特定のものが不要に思われることもあります。これらのうち多くのものはガイドレイヤーに配置されています。Scale-9 を使用したライブプレビューでは、SWF ファイルの実際の表示をすばやく確認できます。

以降の節では、SeekBar、BufferingBar、および VolumeBar の各ムービークリップに対する、より複雑なカスタマイズと変更について説明します。

スキンレイアウトの使用

Flash のスキンの FLA ファイルを開くと、スキンのムービークリップがメインタイムラインにレイアウトされています。これらのクリップと、同じフレームにある ActionScript コードにより、実行時にコントロールがどのようにレイアウトされるかが定義されます。

レイアウトレイヤーの外観は実行時のスキンの外観と似ていますが、このレイヤーのコンテンツは実行時には表示されません。これは、コントロールを配置する場所を計算するためだけに使用されます。実行時にはステージ上の他のコントロールが使用されます。

レイアウトレイヤー内には、`video_mc` という名前の FLVPlayback コンポーネント用のプレースホルダーがあります。その他のすべてのコントロールは、`video_mc` を基準にしてレイアウトされます。Flash FLA ファイルのいずれかを使用して作業を開始し、コントロールのサイズを変更する場合は、これらのプレースホルダークリップを移動するとレイアウトを修正することができます。

各プレースホルダークリップには、特定のインスタンス名があります。プレースホルダークリップの名前は、playpause_mc、play_mc、pause_mc、stop_mc、captionToggle_mc、fullScreenToggle_mc、back_mc、bufferingBar_mc、bufferingBarFill_mc、seekBar_mc、seekBarHandle_mc、seekBarProgress_mc、volumeMute_mc、volumeBar_mc、および volumeBarHandle_mc です。スキンの色を選択したときに再度色が設定される部分は border_mc といいます。

コントロールに対してどのクリップが使用されるかは重要ではありません。通常、ボタンには通常状態クリップが使用されます。他のコントロールに対しては、そのコントロールのクリップが使用されますが、これは便宜上の目的のみです。重要なのは、プレースホルダーの x (水平) 位置、y (垂直) 位置、幅および高さです。

標準のコントロール以外にも、いくつでもクリップを追加できます。これらのクリップで必要なことは、そのライブラリシンボルで、リンケージダイアログボックスの「ActionScript 用」に書き出し」チェックボックスがオンになっていることだけです。レイアウトレイヤーのカスタムクリップは、前述の予約済みインスタンス名以外の任意のインスタンス名を持つことができます。インスタンス名は、クリップで ActionScript を設定してレイアウトを決定するためにのみ必要です。

border_mc クリップは特別なクリップです。FlvPlayback.skinAutoHide プロパティを true に設定すると、マウスが border_mc クリップの上にあるときにスキンが表示されます。これは、ビデオプレーヤーの境界の外側に表示されるスキンにとって重要です。skinAutoHide プロパティについては、156 ページの「スキンのビヘイビアの変更」を参照してください。

Flash FLA ファイルでは、クロムと、「早送り」および「戻る」ボタンの周囲の境界に border_mc が使用されます。

border_mc クリップはスキンの一部でもあり、そのアルファ値と色は skinBackgroundAlpha プロパティと skinBackgroundColor プロパティによって変更されます。カスタマイズ可能な色とアルファ値を使用するには、スキンの FLA ファイルの ActionScript に次のものがが必要です。

```
border_mc.colorMe = true;
```

ActionScript とスキンレイアウト

次の ActionScript コードは、通常はすべてのコントロールに適用されます。一部のコントロールには、追加のビヘイビアを定義する特定の ActionScript がありますが、これについては、そのコントロールのセクションで説明します。

最初の ActionScript は、各コンポーネントの各状態のクラス名を示す大きなセクションです。これらのすべてのクラス名は、SkinOverAll.fla ファイルで確認できます。例えば、「再生」ボタンおよび「一時停止」ボタンについては、コードは次のようになります。

```
this.pauseButtonDisabledState = "fl.video.skin.PauseButtonDisabled";  
this.pauseButtonDownState = "fl.video.skin.PauseButtonDown";  
this.pauseButtonNormalState = "fl.video.skin.PauseButtonNormal";  
this.pauseButtonOverState = "fl.video.skin.PauseButtonOver";  
this.playButtonDisabledState = "fl.video.skin.PlayButtonDisabled";  
this.playButtonDownState = "fl.video.skin.PlayButtonDown";  
this.playButtonNormalState = "fl.video.skin.PlayButtonNormal";  
this.playButtonOverState = "fl.video.skin.PlayButtonOver";
```

これらのクラス名は、実際の外部クラスファイルを持ちません。ライブラリのすべてのムービークリップについて、リンケージダイアログボックスで指定されているだけです。

ActionScript 2.0 コンポーネントでは、実行時に実際に使用されるムービークリップがステージ上にありました。ActionScript 3.0 コンポーネントでも、これらのムービークリップは FLA ファイルに含まれていますが、編集の便宜のためだけに用意されています。現在では、これらはすべてガイドレイヤーにあり、書き出されません。ライブラリのすべてのスキンアセットは最初のフレームに書き出されるように設定されており、例えば次のようなコードで動的に作成されます。

```
new fl.video.skin.PauseButtonDisabled();
```

続くセクションの ActionScript コードでは、スキンの最小の幅と高さを定義します。スキンの選択ダイアログボックスにはこれらの値が表示されます。これらの値は、実行時にスキンが最小サイズ以下に縮小されないようにするために使用されます。最小サイズを指定しない場合は、undefined のままにしておくか、ゼロ以下にします。

```
// minimum width and height of video recommended to use this skin,  

// leave as undefined or <= 0 if there is no minimum  

this.minWidth = 270;  

this.minHeight = 60;
```

各プレースホルダーには、次のプロパティを適用することができます。

プロパティ	説明
anchorLeft	ブール値。FLVPlayback インスタンスの左側を基準にしてコントロールを配置します。anchorRight が明示的に true に設定されていない限り、デフォルトは true です。それ以外の場合、デフォルトは false です。
anchorRight	ブール値。FLVPlayback インスタンスの右側を基準にしてコントロールを配置します。デフォルトは false です。
anchorBottom	ブール値。FLVPlayback インスタンスの下を基準にしてコントロールを配置します。anchorTop が明示的に true に設定されていない限り、デフォルトは true です。それ以外の場合、デフォルトは false です。
anchorTop	ブール値。FLVPlayback インスタンスの上を基準にしてコントロールを配置します。デフォルトは false です。

anchorLeft プロパティおよび anchorRight プロパティが true の場合、コントロールは実行時に水平方向に拡大 / 縮小されません。anchorTop プロパティおよび anchorBottom プロパティが true の場合、コントロールは実行時に垂直方向に拡大 / 縮小されます。

これらのプロパティの効果を表示するには、Flash スキンでの使用方法を確認します。BufferingBar コントロールおよび SeekBar コントロールのみが拡大 / 縮小するコントロールであり、これらは重ねてレイアウトされます。また、両方のコントロールとも、anchorLeft プロパティおよび anchorRight プロパティが true に設定されます。BufferingBar および SeekBar の左側にあるすべてのコントロールでは、anchorLeft が true に設定され、右側にあるすべてのコントロールでは、anchorRight が true に設定されます。すべてのコントロールで、anchorBottom が true に設定されます。

レイアウトレイヤーでムービークリップの編集を試みて、コントロールが下ではなく上に位置するようにスキンを設定することができます。video_mc を基準にしてコントロールを上に移動し、すべてのコントロールについて anchorTop を true にするだけです。

バッファリングバー

バッファリングバーには、bufferingBar_mc および bufferingBarFill_mc の 2 つのムービークリップがあります。ステージ上におけるこれらのクリップの相対位置は重要です。この相対的な位置関係が維持されるからです。バッファリングバーは 2 つのクリップを別々に使用します。これはコンポーネントによって bufferingBar_mc は拡大 / 縮小されますが、bufferingBarFill_mc は拡大 / 縮小されないためです。

bufferingBar_mc クリップには 9 スライスの拡大 / 縮小が割り当てられているので、拡大 / 縮小しても境界は湾曲しません。bufferingBarFill_mc クリップは非常に幅が広く、常に十分な幅があるので、拡大する必要はありません。実行時に自動的にマスクされ、伸縮された bufferingBar_mc の上の部分のみが表示されます。マスクのサイズでは、デフォルトで、bufferingBar_mc および bufferingBarFill_mc の x (水平方向) の位置の差に基づいて、bufferingBar_mc 内の左右の同じマージンが維持されます。ActionScript コードを使用して位置をカスタマイズすることができます。

バッファリングバーを拡大 / 縮小する必要がない場合や、9 スライスの拡大 / 縮小を使用しない場合は、FLV Playback カスタム UI BufferingBar コンポーネントのように設定できます。詳しくは、148 ページの「[BufferingBar コンポーネント](#)」を参照してください。

バッファリングバーには、次の追加のプロパティがあります。

プロパティ	説明
fill_mc:MovieClip	バッファリングバーの塗りのインスタンス名を指定します。デフォルトは bufferingBarFill_mc です。

シークバーとボリュームバー

シークバーにも、seekBar_mc および seekBarProgress_mc の 2 つのムービークリップがあります。レイアウトレイヤー上におけるこれらのクリップの相対位置は重要です。この相対的な位置関係が維持されるからです。どちらのクリップも拡大 / 縮小されますが、seekBarProgress_mc は seekBar_mc 内にネストできません。これは seekBar_mc が 9 スライスの拡大 / 縮小を使用するためで、この拡大 / 縮小はネストされたムービークリップで適切に動作しません。

seekBar_mc クリップには 9 スライスの拡大 / 縮小が割り当てられているので、拡大 / 縮小しても境界は湾曲しません。seekBarProgress_mc クリップも拡大 / 縮小されますが、湾曲します。このクリップに 9 スライスの拡大 / 縮小を使用しない理由は、塗りであり、湾曲しても外観に問題がないためです。

seekBarProgress_mc クリップは fill_mc がなくても機能します。これは、progress_mc クリップが FLV Playback カスタム UI コンポーネントで機能するのと同様です。つまり、マスクされずに水平方向に拡大 / 縮小されます。100% での seekBarProgress_mc の正確なサイズは、seekBarProgress_mc クリップの左右のマージンによって定義されます。このサイズはデフォルトで等しく、seekBar_mc および seekBarProgress_mc の x (水平方向) 位置の差に基づきます。次の例のように、シークバームービークリップのサイズは、ActionScript を使用してカスタマイズすることができます。

```
this.seekBar_mc.progressLeftMargin = 2;  
this.seekBar_mc.progressRightMargin = 2;  
this.seekBar_mc.progressY = 11;  
this.seekBar_mc.fullnessLeftMargin = 2;  
this.seekBar_mc.fullnessRightMargin = 2;  
this.seekBar_mc.fullnessY = 11;
```

このコードは、SeekBar ムービークリップのタイムラインか、またはメインタイムラインの他の ActionScript コードのいずれかに配置できます。レイアウトの変更ではなくコードでカスタマイズした場合、ステージ上に塗りは必要ありません。ライブラリに含まれていて、正しいクラス名でフレーム 1 の ActionScript に書き出すよう設定されていれば十分です。

FLV Playback カスタム UI SeekBar コンポーネントと同じように、シークバーのロード状況ムービークリップを作成することができます。シークバーを拡大 / 縮小する必要がない場合や、拡大 / 縮小しても 9 スライスの拡大 / 縮小は使用しない場合は、FLV Playback カスタム UI コンポーネントに使用される任意の方法を使用して progress_mc または fullness_mc を設定することができます。詳しくは、を参照してください。

Flash スキンのボリュームバーは拡大 / 縮小されないので、VolumeBar FLV Playback カスタム UI コンポーネントと同じ方法で作成されます。詳しくは、148 ページの「SeekBar コンポーネントおよび VolumeBar コンポーネント」を参照してください。例外は、ハンドルの実装が異なることです。

SeekBar ハンドルと VolumeBar ハンドル

SeekBar ハンドルおよび VolumeBar ハンドルは、レイアウトレイヤー上でバーの横に配置されます。デフォルトでは、ハンドルの左マージン、右マージン、および y 軸の値は、バームービークリップを基準とした相対的な位置で設定されます。左マージンはハンドルの x (水平方向) 位置とバーの x (水平方向) 位置の差によって設定され、右マージンは左マージンと等しくなります。これらの値は、SeekBar ムービークリップまたは VolumeBar ムービークリップで ActionScript を使用してカスタマイズできます。次の例は、FLV Playback カスタム UI コンポーネントと共に使用される同じ ActionScript コードです。

```
this.seekBar_mc.handleLeftMargin = 2;  
this.seekBar_mc.handleRightMargin = 2;  
this.seekBar_mc.handleY = 11;
```

このコードは、SeekBar ムービークリップのタイムラインか、またはメインタイムラインの他の ActionScript コードのいずれかに配置できます。レイアウトの変更ではなくコードでカスタマイズした場合、ステージ上にハンドルは必要ありません。ライブラリに含まれていて、正しいクラス名でフレーム 1 の ActionScript に書き出すよう設定されていれば十分です。

これらのプロパティ以外については、ハンドルは単純なムービークリップであり、FLV Playback カスタム UI コンポーネントと同じ方法で設定されます。どちらにも、alpha プロパティが 0 に設定された矩形の背景があります。これらの背景はヒット領域を大きくするためのものであり、必須ではありません。

背景クリップと前景クリップ

ムービークリップ `chrome_mc` および `forwardBackBorder_mc` は背景クリップとして実装されます。

ステージ上の `ForwardBackBorder`、`ForwardBorder` および `BackBorder` の各ムービークリップ、およびプレースホルダーの「早送り」ボタンと「戻る」ボタンのうち、ガイドレイヤー上にないものは `ForwardBackBorder` だけです。これは、実際に「早送り」ボタンおよび「戻る」ボタンを使用するスキン上にあります。

これらのクリップで必要なのは、ライブラリでフレーム 1 の `ActionScript` に書き出されることだけです。

スキンのビヘイビアの変更

`bufferingBarHidesAndDisablesOthers` プロパティおよび `skinAutoHide` プロパティを使用すると、`FLVPlayback` スキンのビヘイビアをカスタマイズできます。

`bufferingBarHidesAndDisablesOthers` プロパティを `true` に設定すると、`FLVPlayback` コンポーネントがバッファリング状態に移行したときに `SeekBar` とそのハンドルを非表示にし、「再生」ボタンおよび「一時停止」ボタンを無効にします。これは、`bufferTime` プロパティに高い値（例えば 10）を設定し、`FLV` ファイルを低速接続で `FMS` からストリーミングする場合に便利です。この状況では、一部のユーザーが「再生」ボタンおよび「一時停止」ボタンをクリックしてシークを開始しようとするのが考えられます。この場合は、さらにファイルの再生が遅れます。このような操作を防止するには、`bufferingBarHidesAndDisablesOthers` を `true` に設定し、コンポーネントがバッファリング状態になっている間は、`SeekBar` エレメント、「再生」ボタンおよび「一時停止」ボタンを無効にします。

`skinAutoHide` プロパティは、デザイン済みスキンの `SWF` ファイルのみに影響を与え、`FLV Playback` カスタム UI コンポーネントによって作成されたコントロールには影響を与えません。このプロパティが `true` に設定されている場合、マウスが表示領域の上にはないときは、スキンが非表示になります。このプロパティのデフォルト値は `false` です。

SMIL ファイルの使用

`VideoPlayer` クラスは、複数の帯域幅の複数のストリームを処理するために、`SMIL` のサブセットをサポートするヘルパークラス (`NCManager`) を使用します。`SMIL` はビデオストリームの位置、`FLV` ファイルのレイアウト（幅と高さ）、および異なる帯域幅に対応するソース `FLV` ファイルを識別するために使用されます。`FLV` ファイルのビットレートと継続時間を指定するためにも使用されます。

`source` パラメーターまたは `FLVPlayback.source` プロパティ (`ActionScript`) を使用して、`SMIL` ファイルの場所を指定します。詳しくは、および「[Adobe Flash Professional CS5 用 ActionScript 3.0 リファレンスガイド](#)」の `FLVPlayback.source` プロパティを参照してください。

次の例では、`RTMP` を使用して `FMS` から複数の帯域幅の `FLV` ファイルをストリーミングする `SMIL` ファイルを示します。

```
<smil>
<head>
<meta base="rtmp://myserver/myapp/" />
<layout>
<root-layout width="240" height="180" />
</layout>
</head>
<body>
    <switch>
        <ref src="myvideo_cable.flv" dur="3:00.1"/>
        <video src="myvideo_isdn.flv" system-bitrate="128000" dur="3:00.1"/>
        <video src="myvideo_mdm.flv" system-bitrate="56000" dur="3:00.1"/>
    </switch>
</body>
</smil>
```

<head> タグには、<meta> タグおよび <layout> タグを含めることができます。<meta> タグは base 属性のみをサポートします。この属性は、ストリーミングビデオ（FMS からの RTMP）の URL を指定するために使用されます。

<layout> タグは root-layout エlementのみをサポートします。このElementは height 属性および width 属性を設定するために使用されるので、FLV ファイルをレンダリングするウィンドウのサイズが決定されます。これらの属性では、パーセンテージではなくピクセル値のみを使用できます。

SMIL ファイルの本体内では、ソース FLV ファイルへの 1 つのリンクを含めるか、<switch> タグを使用してソースファイルを一覧にすることができます（前述の例のように FMS から複数の帯域幅用に複数のファイルをストリーミングする場合）。

<switch> タグ内の video タグおよび ref タグは同義であり、いずれのタグも src 属性を使用して FLV ファイルを指定できます。さらに、いずれのタグも region、system-bitrate および dur の各属性を使用して、領域、必要な最低の帯域幅、および FLV ファイルの継続時間を指定できます。

<body> タグ内では、<video>、<src> または <switch> の各タグのいずれかのインスタンスを 1 つのみ使用できます。

次の例では、帯域幅の検出を使用しない、1 つの FLV ファイル用のプログレッシブダウンロードを示します。

```
<smil>
  <head>
    <layout>
      <root-layout width="240" height="180" />
    </layout>
  </head>
  <body>
    <video src="myvideo.flv" />
  </body>
</smil>
```

<smil>

利用可能なデバイス
Flash Professional 8

使用方法

```
<smil>
...
child tags
...
</smil>
```

属性

なし。

子タグ

<head>, <body>

親タグ

なし。

説明

SMIL ファイルを識別する最上位のタグ。

例

次の例では、3 つの FLV ファイルを指定する SMIL ファイルを示します。

```
<smil>
<head>
<meta base="rtmp://myserver/myapp/" />
<layout>
<root-layout width="240" height="180" />
</layout>
</head>
<body>
<switch>
    <ref src="myvideo_cable.flv" dur="3:00.1"/>
    <video src="myvideo_isdn.flv" system-bitrate="128000" dur="3:00.1"/>
    <video src="myvideo_mdm.flv" system-bitrate="56000" dur="3:00.1"/>
</switch>
</body>
</smil>
```

<head>

利用可能なデバイス
Flash Professional 8

使用方法

```
<head>
...
child tags
...
</head>
```

属性

なし。

子タグ

<meta>, <layout>

親タグ

<smil>

説明

<meta> タグおよび <layout> タグをサポートし、ソース FLV ファイルの場所とデフォルトレイアウト（高さと幅）を指定します。

例

次の例では、ルートレイアウトを 240 × 180 ピクセルに設定します。

```
<head>
    <meta base="rtmp://myserver/myapp/" />
    <layout>
        <root-layout width="240" height="180" />
    </layout>
</head>
```

<meta>

利用可能なデバイス
Flash Professional 8

使用方法
<meta/>

属性
base

子タグ
<layout>

親タグ
なし。

説明
ソース FLV ファイルの場所 (RTMP URL) を指定する base 属性を含みます。

例
次の例では、myserver 上のベース位置の meta タグを示します。

```
<meta base="rtmp://myserver/myapp/" />
```

<layout>

利用可能なデバイス
Flash Professional 8

使用方法
<layout>
...
child tags
...
</layout>

属性
なし。

子タグ
<root-layout>

親タグ
<meta>

説明
FLV ファイルの幅と高さを指定します。

例

次の例では、レイアウトを 240 × 180 ピクセルに設定します。

```
<layout>  
  <root-layout width="240" height="180" />  
</layout>
```

<root-layout>

利用可能なデバイス

Flash Professional 8

使用方法

```
<root-layout...attributes.../>
```

属性

Width、height

子タグ

なし。

親タグ

```
<layout>
```

説明

FLV ファイルの幅と高さを指定します。

例

次の例では、レイアウトを 240 × 180 ピクセルに設定します。

```
<root-layout width="240" height="180" />
```

<body>

利用可能なデバイス

Flash Professional 8

使用方法

```
<body>  
  ...  
  child tags  
  ...  
</body>
```

属性

なし。

子タグ

```
<video>, <ref>, <switch>
```

親タグ

```
<smil>
```

説明

<video>、<ref> および <switch> の各タグを含みます。これらのタグは、ソース FLV ファイルの名前、最低の帯域幅および FLV ファイルの継続時間を指定します。system-bitrate 属性は、<switch> タグを使用する場合のみサポートされます。<body> タグ内では、<switch>、<video> または <ref> の各タグのいずれかのインスタンスを 1 つのみ使用できます。

例

次の例では、3 つの FLV ファイルを指定します。そのうち 2 つは video タグを使用し、1 つは ref タグを使用します。

```
<body>
  <switch>
    <ref src="myvideo_cable.flv" dur="3:00.1"/>
    <video src="myvideo_isdn.flv" system-bitrate="128000" dur="3:00.1"/>
    <video src="myvideo_mdm.flv" system-bitrate="56000" dur="3:00.1"/>
  </switch>
</body>
```

<video>

利用可能なデバイス

Flash Professional 8

使用方法

```
<video...attributes.../>
```

属性

src, system-bitrate, dur

子タグ

なし。

親タグ

```
<body>
```

説明

<ref> タグと同義です。src 属性および dur 属性をサポートします。これらの属性は、ソース FLV ファイルの名前と継続時間を指定します。dur 属性は完全な時間形式 (00:03:00:01) と部分的な時間形式 (03:00:01) をサポートします。

例

次の例では、ビデオのソースと継続時間を設定します。

```
<video src="myvideo_mdm.flv" dur="3:00.1"/>
```

<ref>

利用可能なデバイス
Flash Professional 8

使用方法

```
<ref...attributes.../>
```

属性

src, system-bitrate, dur

子タグ

なし。

親タグ

<body>

説明

<video> タグと同義です。src 属性および dur 属性をサポートします。これらの属性は、ソース FLV ファイルの名前と継続時間を指定します。dur 属性は完全な時間形式 (00:03:00:01) と部分的な時間形式 (03:00:01) をサポートします。

例

次の例では、ビデオのソースと継続時間を設定します。

```
<ref src="myvideo_cable.flv" dur="3:00.1"/>
```

<switch>

利用可能なデバイス
Flash Professional 8

使用方法

```
<switch>  
...  
child tags  
...  
</switch/>
```

属性

なし。

子タグ

<video>, <ref>

親タグ

<body>

説明

<video> 子タグまたは <ref> 子タグと共に使用し、複数の帯域幅のビデオストリーミング用の FLV ファイルを列挙します。
<switch> タグは system-bitrate 属性をサポートします。この属性は、最低の帯域幅、src 属性および dur 属性を指定します。

例

次の例では、3 つの FLV ファイルを指定します。そのうち 2 つは video タグを使用し、1 つは ref タグを使用します。

```
<switch>
  <ref src="myvideo_cable.flv" dur="3:00.1"/>
  <video src="myvideo_isdn.flv" system-bitrate="128000" dur="3:00.1"/>
  <video src="myvideo_mdm.flv" system-bitrate="56000" dur="3:00.1" />
</switch>
```

第7章：FLVPlayback キャプションコンポーネントの使用

FLVPlayback コンポーネントを使用すると、Adobe Flash Professional CS5 アプリケーションにビデオプレーヤーを組み込んで、ダウンロードした Adobe Flash Video (FLV または F4V) ファイルやストリーミング FLV または F4V ファイルを再生できます。FLVPlayback について詳しくは、128 ページの「[FLVPlayback コンポーネントの使用](#)」を参照してください。

FLVPlaybackCaptioning コンポーネントを使用すると、ビデオでクローズドキャプションをサポートできるようになります。キャプションコンポーネントは、W3C 標準の XML 形式、Timed Text をサポートしており、次のような機能があります。

埋め込まれたイベントキューポイントを使用したキャプション Timed Text XML ファイルを使用するのではなく、FLV ファイルに埋め込まれたイベントキューポイントを XML と関連付けてキャプションを付けます。

複数の FLVPlayback キャプション 複数の FLVPlayback インスタンス用に複数の FLVPlayback キャプションインスタンスを作成します。

トグルボタンコントロール ユーザーはキャプショントグルボタンを使用してキャプションを付けることができます。

FLVPlaybackCaptioning コンポーネントの使用

FLVPlaybackCaptioning コンポーネントは、1 つまたは複数の FLVPlayback コンポーネントと共に使用します。最も単純なシナリオでは、FLVPlayback コンポーネントをステージ上にドラッグし、FLVPlaybackCaptioning コンポーネントを同じステージ上にドラッグして、キャプションの URL を識別し、表示するキャプションを設定します。さらに、FLVPlayback キャプションをカスタマイズするさまざまなパラメーターも設定できます。

FLVPlayback コンポーネントへのキャプションの追加

FLVPlaybackCaptioning コンポーネントを任意の FLVPlayback コンポーネントに追加できます。アプリケーションに対する FLVPlayback コンポーネントの追加については、129 ページの「[FLVPlayback コンポーネントを使用したアプリケーションの作成](#)」を参照してください。

コンポーネントパネルからの FLVPlaybackCaptioning コンポーネントの追加：

- 1 コンポーネントパネルで "Video" フォルダを開きます。
- 2 FLVPlaybackCaptioning コンポーネントをドラッグ (またはダブルクリック) し、これを FLVPlayback コンポーネントと同じ、キャプションを追加するステージに追加します。

注意：FLVPlaybackCaptioning コンポーネントの学習に役立つ 2 つのファイル、caption_video.flv (FLVPlayback サンプル) と caption_video.xml (キャプションのサンプル) を用意しています。これらのファイルは、www.helpexamples.com/flash/video/caption_video.flv および www.helpexamples.com/flash/video/caption_video.xml で入手できます。

- 3 (オプション) CaptionButton コンポーネントを、FLVPlayback および FLVPlaybackCaptioning コンポーネントと同じステージにドラッグします。CaptionButton コンポーネントを使用すると、キャプションのオンとオフを切り替えることができます。

注意： CaptionButton コンポーネントを有効にするには、このコンポーネントを FLVPlayback および FLVPlaybackCaptioning コンポーネントと同じステージにドラッグする必要があります。

- 4 ステージ上で FLVPlaybackCaptioning コンポーネントを選択したまま、プロパティインスペクターの「パラメーター」タブで次の必要な情報を指定します。

- showCaptions を true に設定します。
- ダウンロードする Timed Text XML ファイルの source を指定します。

 Flash でのキャプションのテスト中は、showCaptions プロパティを true に設定する必要があります。ただし、ユーザーがキャプションのオンとオフを切り替えることができるように CaptionButton コンポーネントを含めるには、showCaptions プロパティを false に設定する必要があります。

FLVPlaybackCaptioning コンポーネントのカスタマイズに役立つその他のパラメーターもあります。詳しくは、173 ページの「[FLVPlaybackCaptioning コンポーネントのカスタマイズ](#)」および『[ActionScript 3.0 Reference for the Adobe Flash Platform](#)』を参照してください。

- 5 制御/ムービープレビューを選択して、ビデオを再生します。

ActionScript を使用したインスタンスの動的な作成：

- 1 コンポーネントパネルから FLVPlayback コンポーネントをライブラリパネルにドラッグします (ウィンドウ/ライブラリ)。
- 2 FLVPlaybackCaptioning コンポーネントをコンポーネントパネルからライブラリパネルにドラッグします。
- 3 タイムラインのフレーム 1 のアクションパネルに次のコードを追加します。

```
import fl.video.*;
var my_FLVPlaybk = new FLVPlayback();
my_FLVPlaybk.x = 100;
my_FLVPlaybk.y = 100;
addChild(my_FLVPlaybk);
my_FLVPlaybk.skin = "install_drive:/Program Files/Adobe/Adobe Flash CS5/en/Configuration/FLVPlayback
Skins/ActionScript 3.0/SkinUnderPlaySeekCaption.swf";
my_FLVPlaybk.source = "http://www.helpexamples.com/flash/video/caption_video.flv";
var my_FLVPlaybkcap = new FLVPlaybackCaptioning();
addChild(my_FLVPlaybkcap);
my_FLVPlaybkcap.source = "http://www.helpexamples.com/flash/video/caption_video.xml";
my_FLVPlaybkcap.showCaptions = true;
```

- 4 install_drive の部分に Flash のインストール先ドライブを指定し、実際の Skins フォルダの場所に合わせてパスを変更します。

注意： ActionScript で FLVPlayback インスタンスを作成する場合は、ActionScript を使用してスキンプロパティを設定し、スキンも割り当てる必要があります。ActionScript を使用してスキンを適用しても、SWF ファイルと共に自動的にパブリッシュされることはありません。スキン SWF ファイルとアプリケーション SWF ファイルをサーバーにコピーします。コピーしないと、実行したときにスキン SWF ファイルを使用できません。

FLVPlaybackCaptioning コンポーネントパラメーターの設定

FLVPlaybackCaptioning コンポーネントの各インスタンスで、プロパティインスペクターまたはコンポーネントインスペクターの次のパラメーターを設定し、コンポーネントをさらにカスタマイズできます。次のリストでは、プロパティを挙げ、それぞれについて簡単に説明します。

autoLayout FLVPlaybackCaptioning コンポーネントがキャプション領域のサイズを制御するかどうかを指定します。デフォルト値は true です。

`captionTargetName` キャプションが含まれる `TextField` または `MovieClip` インスタンス名を識別します。デフォルト値は `auto` です。

`flvPlaybackName` キャプションを付ける `FLVPlayback` インスタンス名を識別します。デフォルト値は `auto` です。

`simpleFormatting` `true` に設定されている場合は、`Timed Text XML` ファイルのフォーマット手順が制限されます。デフォルト値は `false` です。

`showCaptions` キャプションを表示するかどうかを指定します。デフォルト値は `true` です。

`source` `Timed Text XML` ファイルの場所を指定します。

すべての `FLVPlaybackCaptioning` パラメーターについては、『[ActionScript 3.0 Reference for the Adobe Flash Platform](#)』を参照してください。

source パラメーターの指定

`source` パラメーターを使用して、ムービーのキャプションが含まれる `Timed Text XML` ファイルの名前と場所を指定します。コンポーネントインスペクターの `source` セルに URL パスを直接入力します。

キャプションの表示

キャプションを表示するには、`showCaptions` パラメーターを `true` に設定します。

すべての `FLVPlaybackCaptioning` コンポーネントのパラメーターについては、『[ActionScript 3.0 Reference for the Adobe Flash Platform](#)』を参照してください。

前の例では、キャプションを表示する `FLVPlaybackCaptioning` コンポーネントを作成し、有効にする方法を学習しました。キャプションに使用できるソースには、(1) キャプションが含まれる `Timed Text XML` ファイルと、(2) 埋め込まれたイベントキューポイントと関連付けるキャプションテキストを持つ `XML` ファイルの 2 つがあります。

Timed Text キャプションの使用

`FLVPlaybackCaptioning` コンポーネントを使用すると、`Timed Text (TT) XML` ファイルをダウンロードすることにより、関連付けられた `FLVPlayback` コンポーネントのキャプション付けが可能になります。`Timed Text` 形式については、www.w3.org で `AudioVideo Timed Text` の情報を参照してください。

この節では、サポートされている `Timed Text` タグの概要、必要なキャプションファイルタグ、`Timed Text XML` ファイルの例を説明します。サポートされているすべての `Timed Text` タグについては、168 ページの「[Timed Text タグ](#)」を参照してください。

`FLVPlaybackCaptioning` コンポーネントは次の `Timed Text` タグをサポートしています。

カテゴリ	タスク
段落の書式情報のサポート	段落を右揃え、左揃え、または中央揃えにします。
テキストの書式情報のサポート	<ul style="list-style-type: none"> • テキストのサイズを、絶対ピクセルサイズまたは Delta スタイル (+2、-4 など) で設定します。 • テキストカラーとフォントを設定します。 • テキストをボールドまたはイタリックにします。 • テキストの位置揃えを設定します。
その他の書式情報のサポート	<ul style="list-style-type: none"> • キャプションの TextField の背景色を設定します。 • キャプションの TextField の背景色を透明 (アルファ 0) に設定します。 • キャプションの TextField の折り返しを設定します (オンまたはオフ)。

FLVPlaybackCaptioning コンポーネントは FLV ファイルの時間コードに一致します。すべてのキャプションには、キャプションを表示する時刻を指定する begin 属性が必要です。キャプションに dur または end 属性がない場合は、次のキャプションが表示されたとき、または FLV ファイルが終了したときにキャプションが消去されます。

次に Timed Text XML ファイルの例を示します。このファイル (caption_video.xml) は "caption_video.flv" ファイルのキャプションを指定します。これらのファイルは、www.helpexamples.com/flash/video/caption_video.flv および www.helpexamples.com/flash/video/caption_video.xml で入手できます。

```
<?xml version="1.0" encoding="UTF-8"?>
  <tt xml:lang="en"
xmlns="http://www.w3.org/2006/04/ttaf1"xmlns:tts="http://www.w3.org/2006/04/ttaf1#styling">
<head>
  <styling>
<style id="1" tts:textAlign="right"/>
<style id="2" tts:color="transparent"/>
<style id="3" style="2" tts:backgroundColor="white"/>
<style id="4" style="2 3" tts:fontSize="20"/>
  </styling>
</head>
<body>
  <div xml:lang="en">
<p begin="00:00:00.00" dur="00:00:03.07">I had just joined <span
tts:fontFamily="monospaceSansSerif,proportionalSerif,TheOther"tts:fontSize="+2">Macromedia</span> in
1996,</p>
<p begin="00:00:03.07" dur="00:00:03.35">and we were trying to figure out what to do about the internet.</p>
<p begin="00:00:06.42" dur="00:00:03.15">And the company was in dire straights at the time.</p>
<p begin="00:00:09.57" dur="00:00:01.45">We were a CD-ROM authoring company,</p>
<p begin="00:00:11.42" dur="00:00:02.00">and the CD-ROM business was going away.</p>
<p begin="00:00:13.57" dur="00:00:02.50">One of the technologies I remember seeing was Flash.</p>
<p begin="00:00:16.47" dur="00:00:02.00">At the time, it was called <span tts:fontWeight="bold"
tts:color="#ccc333">FutureSplash</span>.</p>
<p begin="00:00:18.50" dur="00:00:01.20">So this is where Flash got its start.</p>
<p begin="00:00:20.10" dur="00:00:03.00">This is smart sketch running on the <span
tts:fontStyle="italic">EU-pin computer</span>,</p>
<p begin="00:00:23.52" dur="00:00:02.00">which was the first product that FutureWave did.</p>
<p begin="00:00:25.52" dur="00:00:02.00">So our vision for this product was to</p>
<p begin="00:00:27.52" dur="00:00:01.10">make drawing on the computer</p>
<p begin="00:00:29.02" dur="00:00:01.30" style="1">as <span tts:color="#ccc333">easy</span> as drawing on
paper.</p>
</div>
</body>
</tt>
```

Timed Text タグ

FLVPlaybackCaptioning コンポーネントは、XML ファイルのキャプション用に Timed Text タグをサポートしています。オーディオビデオの Timed Text タグについて詳しくは、www.w3.org の情報を参照してください。次の表に、サポートされているタグとサポートされていないタグを示します。

機能	タグ/値	用途/説明	例
無視されるタグ	metadata	無視されるか、ドキュメントのいずれかのレベルで許可されます。	
	set	無視されるか、ドキュメントのいずれかのレベルで許可されます。	
	xml:lang	無視されます。	
	xml:space	無視されるか、ビヘイビアが <code>xml:space="default"</code> より優先されます。	
	layout	無視されるか、レイアウトタグセクションに任意の領域タグを含みます。	
	br タグ	すべての属性とコンテンツが無視されます。	
キャプションのメディアのタイミング	begin 属性	p タグだけで使用されます。キャプションのメディアに時刻を展開するために必要です。	<p begin="3s">
	dur 属性	p タグだけで使用されます。推奨されます。この属性が含まれていないと、キャプションは FLV ファイル、または別のキャプションの開始と共に終了します。	
	end 属性	p タグだけで使用されます。推奨されます。この属性が含まれていないと、キャプションは FLV ファイル、または別のキャプションの開始と共に終了します。	
キャプションのクロックタイミング	00:03:00.1	フルクロック形式	
	03:00.1	部分クロック形式	
	10	単位の無いオフセット時間。オフセットは秒を表します。	
	00:03:00:05 00:03:00:05.1 30f 30t	サポートされません。フレームや目盛を含む時間形式はサポートされません。	
本文タグ	body	必須、または 1 つの body タグだけがサポートされます。	<body><div>...</div></body>
コンテンツタグ	div タグ	指定しないことも、1 つ以上指定することもできます。最初のタグが使用されます。	
	p タグ	指定しないことも、1 つ以上指定することもできます。	
	span タグ	連続したテキストコンテンツの単位の論理コンテナ。ネストされた span はサポートされません。属性スタイルタグをサポートします。	
	br タグ	明示的な改行を示します。	

機能	タグ/値	用途/説明	例
スタイル指定タグ (p タグ内部ではすべてのスタイルタグが使用される)	style	1 つまたは複数のスタイルエレメントを参照します。タグや属性として使用できます。タグとして使用する場合は ID 属性が必要です (style はドキュメントで再使用できます)。style タグ内部で 1 つまたは複数の style タグをサポートします。	
	tts:background Color	領域の背景色を定義するスタイルプロパティを指定します。ゼロに設定して (アルファ 0) 背景を透明にしない限り、アルファは無視されます。カラー形式は #RRGGBBAA です。	
	tts:color	前景色を定義するスタイルプロパティを指定します。どの色についてもアルファはサポートされません。値 transparent は黒に変換されます。	<pre><style id="3" style="2" tts:backgroundColor="white"/> "transparent" = #00000000 "black" = #000000FF "silver" = #C0C0C0FF "grey" = #808080FF "white" = #FFFFFFFF "maroon" = #800000FF "red" = #FF0000FF "purple" = #800080FF "fuchsia" ("magenta") = #FF00FFFF "green" = #008000FF "lime" = #00FF00FF "olive" = #808000FF "yellow" = #FFFF00FF "navy" = #000080FF "blue" = #0000FFFF "teal" = #008080FF "aqua" ("cyan") = #00FFFFFF</pre>
	tts:fontFamily	フォントファミリーを定義するスタイルプロパティを指定します。	<pre>"default" = _serif "monospace" = _typewriter "sansSerif" = _sans "serif" = _serif "monospaceSansSerif" = _typewriter "monospaceSerif" = _typewriter "proportionalSansSerif" = _sans</pre>
	tts:fontSize	フォントサイズを定義するスタイルプロパティを指定します。2 つの値が指定されている場合は、最初の値 (垂直) が使用されます。パーセント値と単位は無視されます。絶対ピクセル (12 など) と相対スタイル (+2 など) のサイズがサポートされています。	

機能	タグ/値	用途/説明	例
	tts:fontStyle	フォントスタイルを定義するスタイルプロパティを指定します。	"normal" "italic" "inherit"* * デフォルトのピヘイピア。スタイルは、囲んでいるタグから継承されます。
	tts:fontWeight	フォントの太さを定義するスタイルプロパティを指定します。	"normal" "bold" "inherit"* * デフォルトのピヘイピア。スタイルは、囲んでいるタグから継承されます。
	tts:textAlign	含まれるブロック領域内部でインライン領域を揃える方法を定義するスタイルプロパティを指定します。	"left" "right" "center" "start" ("=left") "end" ("=right") "inherit"* * スタイルは、囲んでいるタグから継承されません。textAlign タグを設定しない場合、デフォルト値は "left" です。
	tts:wrapOption	影響を受けるエレメントのコンテキスト内で行の自動的な折り返し (改行) を適用するかどうかを定義するスタイルプロパティを指定します。この設定は、キャプションエレメントのすべての段落に影響を与えます。	"wrap" "noWrap" "inherit"* * スタイルは、囲んでいるタグから継承されません。wrapOption タグを設定しない場合、デフォルト値は "wrap" です。
サポートされていない属性	tts:direction tts:display tts:displayAlign tts:dynamicFlow tts:extent tts:lineHeight tts:opacity tts:origin tts:overflow tts:padding tts:showBackground tts:textOutline tts:unicodeBidi tts:visibility tts:writingMode tts:zIndex		

キャプションでのキューポイントの使用

キューポイントを使用するとビデオを操作できます。例えば、FLV ファイルの再生を制御したり、特定の時刻にビデオにテキストを表示したりできます。FLV ファイルに使用する Timed Text XML ファイルがない場合は、イベントキューポイントを FLV ファイルに埋め込み、これらのキューポイントをテキストと関連付けることができます。このセクションでは、FLVPlaybackCaptioning コンポーネントのキューポイントの標準に関する情報を提供し、これらのキューポイントをキャプションニング用のテキストと関連付ける方法について説明します。ビデオの読み込みウィザードを使用してイベントキューポイントを埋め込む方法について詳しくは、『Flash ユーザーガイド』の第 16 章「ビデオの使用」を参照してください。

FLVPlaybackCaptioning キューポイントの標準について

FLV ファイルのメタデータ内では、キューポイントは name、time、type および parameters の各プロパティを持つオブジェクトとして表されます。FLVPlaybackCaptioning ActionScript キューポイントには次の属性があります。

name name プロパティは、キューポイントに割り当てられた名前を含むストリングです。name プロパティの先頭は接頭辞 `fl.video.caption.2.0.` である必要があり、この接頭辞の後にストリングが続きます。ストリングは、それぞれの名前を一意にするために毎回増分される一連の正の整数です。接頭辞には、FLVPlayback バージョン番号にも一致するバージョン番号が含まれます。Adobe Flash CS4 以降では、バージョン番号を 2.0 に設定する必要があります。

time time プロパティはキャプションを表示する時刻です。

type type プロパティは、値が "event" のストリングです。

parameters parameters プロパティは、次の名前と値のペアをサポートする配列です。

- **text:String** キャプションの HTML 形式のテキスト。このテキストは `TextField.htmlText` プロパティに直接渡されます。FLVPlaybackCaptioning コンポーネントは、オプションの `text:n` プロパティをサポートしており、このプロパティは、複数の言語トラックの使用をサポートしています。詳しくは、173 ページの「[埋め込まれたキューポイントを使用した複数の言語トラックのサポート](#)」を参照してください。
- **endTime:Number** キャプションを消去する時刻。このプロパティを指定しないと、FLVPlaybackCaptioning コンポーネントは、これを数値ではないもの (NaN) とみなし、FLV ファイルが終了するまで (FLVPlayback インスタンスが `VideoEvent.COMPLETE` イベントを送出するまで) キャプションが表示されます。endTime:Number プロパティは秒単位で指定します。
- **backgroundColor:uint** このパラメーターは `TextField.backgroundColor` を設定します。このプロパティは省略可能です。
- **backgroundColorAlpha:Boolean** `backgroundColor` のアルファが 0% の場合、このパラメーターは `TextField.background = !backgroundColor` を設定します。このプロパティは省略可能です。
- **wrapOption:Boolean** このパラメーターは `TextField.wordWrap` を設定します。このプロパティは省略可能です。

埋め込まれたイベントキューポイントのキャプションについて

FLV ファイルのキャプションが含まれる Timed Text XML ファイルがない場合は、キャプションが含まれる XML ファイルと、埋め込まれたイベントキューポイントを関連付けることによってキャプションを作成します。XML のサンプルでは、次の手順を実行して、埋め込まれたイベントキューポイントをビデオで作成すると仮定します。

- FLVPlaybackCaptioning の標準に従ってイベントキューポイントを追加し、ビデオをエンコードします。
- Flash で、FLVPlayback コンポーネントと FLVPlaybackCaptioning コンポーネントをステージにドラッグします。
- FLVPlayback および FLVPlaybackCaptioning コンポーネントのソースプロパティ (FLV ファイルの場所と XML ファイルの場所) を設定します。
- パブリッシュします。

次のサンプルでは、XML をエンコーダーに読み込みます。

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<FLVCoreCuePoints>

  <CuePoint>
    <Time>9136</Time>
    <Type>event</Type>
    <Name>fl.video.caption.2.0.index1</Name>
    <Parameters>
      <Parameter>
        <Name>text</Name>
        <Value><![CDATA[Captioning text for the first cue point]]></Value>
      </Parameter>
    </Parameters>
  </CuePoint>

  <CuePoint>
    <Time>19327</Time>
    <Type>event</Type>
    <Name>fl.video.caption.2.0.index2</Name>
    <Parameters>
      <Parameter>
        <Name>text</Name>
        <Value><![CDATA[Captioning text for the second cue point]]></Value>
      </Parameter>
    </Parameters>
  </CuePoint>

  <CuePoint>
    <Time>24247</Time>
    <Type>event</Type>
    <Name>fl.video.caption.2.0.index3</Name>
    <Parameters>
      <Parameter>
        <Name>text</Name>
        <Value><![CDATA[Captioning text for the third cue point]]></Value>
      </Parameter>
    </Parameters>
  </CuePoint>

  <CuePoint>
    <Time>36546</Time>
    <Type>event</Type>
    <Name>fl.video.caption.2.0.index4</Name>
    <Parameters>
      <Parameter>
        <Name>text</Name>
        <Value><![CDATA[Captioning text for the fourth cue point]]></Value>
      </Parameter>
    </Parameters>
  </CuePoint>

</FLVCoreCuePoints>
```

FLVPlaybackCaptioning コンポーネントは、埋め込まれたキューポイントを使用して複数の言語トラックもサポートしています。詳しくは、173 ページの「[埋め込まれたキューポイントを使用した複数の言語トラックのサポート](#)」を参照してください。

埋め込まれたキューポイントを使用した複数の言語トラックのサポート

Timed Text XML ファイルが FLVPlaybackCaptioning のキューポイントの標準に従っている限り、FLVPlaybackCaptioning の track プロパティは、埋め込まれたキューポイントを使用して複数の言語トラックをサポートします。詳しくは、171 ページの「[FLVPlaybackCaptioning キューポイントの標準について](#)」を参照してください。ただし、FLVPlaybackCaptioning コンポーネントは個別の XML ファイルでは複数の言語トラックをサポートしません。track プロパティを使用するには、このプロパティを 0 以外の値に設定します。例えば、track プロパティを 1 (track == 1) に設定すると、FLVPlaybackCaptioning コンポーネントはキューポイントパラメーターを検索します。一致が見つからない場合は、キューポイントパラメーターのテキストプロパティが使用されます。詳しくは、『[ActionScript 3.0 Reference for the Adobe Flash Platform](#)』の track プロパティを参照してください。

キャプションを使用した複数の FLV ファイルの再生

FLVPlayback コンポーネントの 1 つのインスタンス内で複数のビデオプレーヤーを開き、複数のビデオを再生して、再生中にそれらを切り替えることもできます。キャプションを FLVPlayback コンポーネント内部の各ビデオプレーヤーと関連付けることもできます。複数のビデオプレーヤーを開く方法について詳しくは、143 ページの「[複数のビデオプレーヤーの使用](#)」を参照してください。複数のビデオプレーヤーでキャプションを使用するには、VideoPlayer ごとに FLVPlaybackCaptioning コンポーネントのインスタンスを 1 つ作成し、FLVPlaybackCaptioning videoPlayerIndex を対応するインデックスに設定します。VideoPlayer が 1 つだけの場合、VideoPlayer インデックスのデフォルト値は 0 です。

一意のキャプションを一意のビデオに割り当てるコードの例を次に示します。この例を実行するには、例の架空の URL を有効な URL に置き換える必要があります。

```
captioner0.videoPlayerIndex = 0;
captioner0.source = "http://www.[yourDomain].com/mytimedtext0.xml";
flvPlayback.play("http://www.[yourDomain].com/myvideo0.flv");
captioner1.videoPlayerIndex = 1;
captioner1.source = "http://www.[yourDomain].com/mytimedtext1.xml";
flvPlayback.activeVideoIndex = 1;
flvPlayback.play ("http://www.[yourDomain].com/myvideo1.flv");
```

FLVPlaybackCaptioning コンポーネントのカスタマイズ

FLVPlaybackCaptioning コンポーネントをすばやく使い始めるには、キャプションを FLVPlayback コンポーネントに直接配置する FLVPlaybackCaptioning のデフォルトを使用できます。キャプションをビデオから離すように FLVPlaybackCaptioning コンポーネントをカスタマイズすることもできます。

次のコードでは、キャプションボタンを使用して、FLVPlayback オブジェクトを動的に作成する方法を説明します。

- 1 FLVPlayback コンポーネントをステージの 0,0 に配置し、インスタンス名を **player** とします。
- 2 FLVPlaybackCaptioning コンポーネントをステージの 0,0 に配置し、インスタンス名を **captioning** とします。
- 3 CaptionButton コンポーネントをステージ上に配置します。
- 4 次のコード例では、testVideoPath:String 変数を FLV ファイルの絶対パスまたは相対パスに設定します。

注意：このコード例では、testVideoPath 変数を、Flash のサンプルビデオ caption_video.flv に設定します。この変数を、キャプションボタンコンポーネントの追加先となるビデオキャプションコンポーネントのパスに変更してください。

- 5 次のコード例では、testCaptioningPath:String 変数を適切な Timed Text XML ファイルの絶対パスまたは相対パスに設定します。

注意：このコード例では、testCaptioningPath 変数を、Timed Text XML ファイル caption_video.xml に設定します。この変数を、ビデオのキャプションが含まれる Timed Text XML ファイルのパスに変更してください。

6 次のコードを "FLVPlaybackCaptioningExample.as" として、FLA ファイルと同じディレクトリに保存します。

7 FLA ファイルで、DocumentClass を FLVPlaybackCaptioningExample に設定します。

```
package
{
    import flash.display.Sprite;
    import flash.text.TextField;
    import fl.video.FLVPlayback;
    import fl.video.FLVPlaybackCaptioning;

    public class FLVPlaybackCaptioningExample extends Sprite {

        private var testVideoPath:String = "http://www.helpexamples.com/flash/video/caption_video.flv";
        private var testCaptioningPath:String =
"http://www.helpexamples.com/flash/video/caption_video.xml";

        public function FLVPlaybackCaptioningExample() {
            player.source = testVideoPath;
            player.skin = "SkinOverAllNoCaption.swf";
            player.skinBackgroundColor = 0x666666;
            player.skinBackgroundAlpha = 0.5;

            captioning.flvPlayback = player;
            captioning.source = testCaptioningPath;
            captioning.autoLayout = false;
            captioning.addEventListener("captionChange", onCaptionChange);
        }
        private function onCaptionChange(e:*):void {
            var tf:* = e.target.captionTarget;
            var player:FLVPlayback = e.target.flvPlayback;

            // move the caption below the video
            tf.y = 210;
        }
    }
}
```

すべての FLVPlaybackCaptioning パラメーターについては、『**ActionScript 3.0 Reference for the Adobe Flash Platform**』を参照してください。