

FLASH® LITE™ 1.x ACTIONSCRIPT™の学習

© 2007 Adobe Systems Incorporated. All rights reserved.

Flash® Lite™ 1.x ActionScript™ の学習

本マニュアルがエンドユーザー使用許諾契約を含むソフトウェアと共に提供される場合、本マニュアルおよびその中に記載されているソフトウェアは、エンドユーザー使用許諾契約にもとづいて提供されるものであり、当該エンドユーザー使用許諾契約の契約条件に従ってのみ使用または複製することが可能となるものです。当該エンドユーザー使用許諾契約により許可されている場合を除き、本マニュアルのいかなる部分といえども、Adobe Systems Incorporated (アドビ システムズ社) の書面による事前の許可なしに、電子的、機械的、録音、その他いかなる形式・手段であれ、複製、検索システムへの保存、または伝送を行うことはできません。本マニュアルの内容は、エンドユーザー使用許諾契約を含むソフトウェアと共に提供されていない場合であっても、著作権法により保護されていることにご留意ください。

本マニュアルに記載される内容は、あくまでも参照用としてのみ使用されること、また、なんら予告なしに変更されることを条件として、提供されるものであり、従って、当該情報が、アドビ システムズ社による確約として解釈されることはありません。アドビ システムズ社は、本マニュアルにおけるいかなる誤りまたは不正確な記述に対しても、いかなる義務や責任を負うものではありません。

新しいア트워크を創作するためにテンプレートとして取り込もうとする既存のア트워크または画像は、著作権法により保護されている可能性のあるものであることをご留意ください。保護されているア트워크または画像を新しいア트워크に許可なく取り込んだ場合、著作権者の権利を侵害することがあります。従って、著作権者から必要なすべての許可を必ず取得してください。

例として使用されている会社名は、実在の会社・組織を示すものではありません。

Adobe、Adobe ロゴ、Flash Lite、および Flash は、アドビ システムズ社の米国ならびに他の国における商標または登録商標です。

サードパーティ情報

本マニュアルには、アドビ システムズ社が管理していない、サードパーティの Web サイトへのリンクが掲載されていますが、アドビ システムズ社はいかなるリンク先サイトの内容についても責任を持ちません。本マニュアルに記載されているサードパーティの Web サイトには、自己責任においてアクセスしてください。アドビ システムズ社はこれらのリンクを便宜上の目的においてのみ掲載しています。リンクを掲載することにより、アドビ システムズ社がこれらのサードパーティのサイトの内容について何らかの責任を持つことを示すものではありません。



Sorenson™ Spark™ ビデオ圧縮および圧縮解除テクノロジーは、Sorenson Media, Inc. のライセンス供与によって提供されます。

Fraunhofer-IIS/Thomson Multimedia : MPEG レイヤー 3 音声圧縮テクノロジーは、Fraunhofer IIS および Thomson Multimedia (<http://www.iis.fhg.de/amm/>) によりライセンス供与されています。

Independent JPEG Group: 本ソフトウェアの一部は、Independent JPEG Group による著作物に基づきます。

Nellymoser, Inc. : 音声圧縮および圧縮解除テクノロジーは、Nellymoser, Inc. (<http://www.nelly-moser.com>) のライセンス供与によって提供されます。

Opera® browser Copyright © 1995-2002 Opera Software ASA and its suppliers. All rights reserved.

Macromedia Flash 8 ビデオには On2 TrueMotion ビデオテクノロジーが使用されています。© 1992-2005 On2 Technologies, Inc. All Rights Reserved. <http://www.on2.com>.

Visual SourceSafe は米国およびその他の国における Microsoft Corporation の登録商標または商標です。

更新情報およびその他のサードパーティのコード情報は、http://www.adobe.com/go/thirdparty_jp/ で入手できます。

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA.

Notice to U.S. Government End Users. The Software and Documentation are "Commercial Items," as that term is defined at 48 C.F.R. §2.101, consisting of "Commercial Computer Software" and "Commercial Computer Software Documentation," as such terms are used in 48 C.F.R. §12.212 or 48 C.F.R. §227.7202, as applicable. Consistent with 48 C.F.R. §12.212 or 48 C.F.R. §§227.7202-1 through 227.7202-4, as applicable, the Commercial Computer Software and Commercial Computer Software Documentation are being licensed to U.S. Government end users (a) only as Commercial Items and (b) with only those rights as are granted to all other end users pursuant to the terms and conditions herein. Unpublished-rights reserved under the copyright laws of the United States. Adobe Systems Incorporated, 345 Park Avenue, San Jose, CA 95110-2704, USA. For U.S. Government End Users, Adobe agrees to comply with all applicable equal opportunity laws including, if appropriate, the provisions of Executive Order 11246, as amended, Section 402 of the Vietnam Era Veterans Readjustment Assistance Act of 1974 (38 USC 4212), and Section 503 of the Rehabilitation Act of 1973, as amended, and the regulations at 41 CFR Parts 60-1 through 60-60, 60-250, and 60-741. The affirmative action clause and regulations contained in the preceding sentence shall be incorporated by reference.

目次

第1章：Flash Lite 1.x ActionScript について	7
Flash Lite 1.x ActionScript の概要.....	7
Flash Lite 1.0 と Flash Lite 1.1 ActionScript の違い.....	8
Flash Lite 1.x ActionScript がサポートしていない Flash 4 ActionScript.....	8
Flash Lite 1.x ActionScript で使用できない機能.....	9
第2章：Flash 4 ActionScript 入門	11
ムービークリッププロパティの取得と設定.....	11
その他のタイムラインの制御.....	12
変数の使用.....	13
配列のシミュレート.....	13
テキストおよびストリングの操作.....	14
call() 関数を使用しての関数作成.....	15
eval() 関数の使用.....	19
第3章：一般的なスクリプティング作業	21
デバイスとプラットフォームの機能の確認.....	21
Web ページを開く.....	22
通話の開始.....	22
テキストメッセージまたはマルチメディアメッセージの開始.....	23
電子メールメッセージの開始.....	23
外部 SWF ファイルのロード.....	24
外部データのロード.....	24
索引	27

Flash Lite 1.x ActionScript について

ActionScript を使用することで、Adobe® Flash® Lite™ アプリケーションにプログラミングロジックおよびインタラクティブ性を追加することができます。アドビシステムズ社の Macromedia® Flash® Lite™ 1.0 と 1.1 の ActionScript は、総称して Flash Lite 1.x ActionScript と呼ばれるアドビシステムズ社のハイブリッド版 Macromedia® Flash® 4 ActionScript です。通話やテキストメッセージの開始、デバイスからの時刻や日付の取得などの Flash Lite Player に固有のコマンドおよびプロパティが追加されています。

この章では、次のトピックについて説明します。

Flash Lite 1.x ActionScript の概要	7
Flash Lite 1.0 と Flash Lite 1.1 ActionScript の違い	8
Flash Lite 1.x ActionScript がサポートしていない Flash 4 ActionScript	8
Flash Lite 1.x ActionScript で使用できない機能	9

Flash Lite 1.x ActionScript の概要

Flash Lite 1.x ActionScript は次の部分で構成されています。

Flash Player 4 ActionScript これには、演算子 (例: 比較演算子、代入演算子)、ムービークリッププロパティ (例: `_height`、`_x`、`_y`)、タイムライン制御関数 (例: `gotoAndPlay()`、`stop()`)、`loadVariables()` および `loadMovie()` 関数 (Flash Lite 1.1 のみ) などのネットワーク関数が含まれます。サポートしていない Flash 4 ActionScript の一覧については、[8 ページの「Flash Lite 1.x ActionScript がサポートしていない Flash 4 ActionScript」](#)を参照してください。

電話統合コマンドおよびプロパティ Flash Lite には、たとえばデバイスに日付や時刻に関する情報を照会したり、通話やショートメッセージサービス (SMS) テキストメッセージを開始したり、あるいはデバイスにインストールされた外部アプリケーションを開始したりするコマンドが用意されています。

プラットフォーム機能変数 (Flash Lite 1.1 のみ) これらのプロパティはデバイスの機能や Flash Lite ランタイム環境に関する情報を提供します。たとえば、`_capLoadData` 変数は、ネットワーク経由でデータをロードできるアプリケーションかどうかを示します。

fscommand2() 関数 fscommand() 関数同様、fscommand2() 関数を使用して、ホスト環境またはシステム (この場合は携帯電話やモバイルデバイス) と通信することができます。fscommand2() 関数は、任意の数の引数を渡したり、(fscommand() のように次のフレームまで待つことなく) 即時に戻り値を取得したりできる能力など、fscommand() 関数の機能を拡張したものになっています。

Flash Lite 1.0 と Flash Lite 1.1 ActionScript の違い

次の Flash Lite 1.1 ActionScript の機能は Flash Lite 1.0 ではサポートされていません。

- ネットワークアクセスまたはネットワークステータス情報。たとえば、Flash Lite 1.0 では loadVariables() 関数や loadMovie() 関数を使用して外部データや SWF ファイルをロードしたり、各種 fscommand2() コマンドを使用してデバイスの接続シグナル強度やネットワーク要求のステータスを設定したりすることはできません。
- デバイスの時刻および日付情報の取得。
- プラットフォーム機能変数。これらの変数で、Flash Lite プラットフォームとデバイスの機能に関する情報が提供されます。
- fscommand2() 関数と、SetSoftKeys および FullScreen などの関連コマンド。
- scroll および maxscroll テキストフィールドプロパティ。

Flash Lite 1.x ActionScript がサポートしていない Flash 4 ActionScript

次の Flash 4 ActionScript の機能は Flash Lite 1.x ActionScript ではまったくサポートされていないか、一部しかサポートされていません。

- startDrag() 関数および stopDrag() 関数。
- Flash Lite 1.x ActionScript では、Flash Player 4 でサポートされているボタンイベントの一部のみがサポートされています。ボタンイベントの処理の詳細については、『Flash Lite アプリケーションの開発』の第 1 章の「インタラクティブ機能とナビゲーション機能の作成」を参照してください。
- Flash Lite 1.x ActionScript では、Flash Player 4 でサポートされているキーイベントの一部のみがサポートされています。Flash Lite でサポートされているキーイベントについては、『Flash Lite アプリケーションの開発』の第 1 章の「インタラクティブ機能とナビゲーション機能の作成」を参照してください。
- _dropTarget プロパティ。
- _soundBufTime プロパティ。

- `_url` プロパティ。
- `String()` 変換関数。

Flash Lite 1.x ActionScript で使用できない機能

Flash Lite Player は Flash Player の古いバージョンを基にしているため、最近のリリースの Flash Player や使い慣れている他のプログラミング言語で利用可能なプログラミング機能すべてをサポートしているわけではありません。この項では、Flash Lite 1.x ActionScript で使用できないプログラミング機能を取り上げ、代替手段や回避策を説明します。

ユーザー定義関数 Flash Lite 1.x では、カスタム関数の定義および呼び出し機能はサポートしていません。ただし、`call()` 関数を使用して、タイムラインの任意のフレームにあるコードを実行できます。詳細については、[15 ページの「call\(\) 関数を使用しての関数作成」](#)を参照してください。

ネイティブ配列、オブジェクト、その他の複雑なデータタイプ Flash Lite 1.x では、ネイティブ配列データ構造やその他の複雑なデータタイプをサポートしていません。ただし、カンマ区切りの連結されたストリングを動的に評価する `eval()` 関数の使用を含む疑似配列によりネイティブ配列と同じ機能を果たすことができます。詳細については、[13 ページの「配列のシミュレート」](#)を参照してください。

外部イメージあるいはサウンドファイルのランタイムロード Flash Player デスクトップバージョンと異なり、Flash Lite 1.x ActionScript では外部 JPEG ファイルや MP3 ファイルをロードすることができません。Flash Lite 1.1 では、`loadMovie()` 関数を使用して、外部 SWF ファイルをロードできます。詳細については、[24 ページの「外部 SWF ファイルのロード」](#)を参照してください。

アドビシステムズ社の Macromedia Flash Lite 1.x ActionScript は、アドビシステムズ社の Macromedia Flash Player 4 で利用可能になったバージョンの ActionScript をベースにしています。このため、Flash Player (デスクトップシステム用) のその後のバージョンで利用可能になったプログラミング機能には、Flash Lite 1.x アプリケーションでは使用できないものがあります。

Flash 4 ActionScript のシンタックスや機能に詳しくない方や、以前の Flash 開発作業でのいくつかの詳細を忘れてしまった方のために、この章では、Flash Lite アプリケーションで Flash 4 ActionScript を使用する際の入門情報を提供します。

この章では、次のトピックについて説明します。

ムービークリッププロパティの取得と設定	11
その他のタイムラインの制御	12
変数の使用	13
配列のシミュレート	13
テキストおよびストリングの操作	14
call() 関数を使用しての関数作成	15
eval() 関数の使用	19

ムービークリッププロパティの取得と設定

ムービークリッププロパティの取得と設定 (設定可能な場合) を行うには、ドットシンタックス、または `setProperty()` 関数か `getProperty()` 関数を使用できます。また、`tellTarget()` 関数も使用することが可能です。

ドットシンタックスを使用するには、ムービークリップインスタンス名、ドット (`.`)、プロパティ名の順に指定します。たとえば、次のコードでは、`cartoonArea` という名前のムービークリップの `x` 画面座標 (`_x` ムービークリッププロパティで表される) を取得し、`x_pos` という名前の変数にその結果を割り当てます。

```
x_pos = cartoonArea._x;
```

次の例は、前の例と同等ですが、`getProperty()` 関数を使用してムービークリップの `x` 位置を取得します。

```
x_pos = getProperty(cartoonArea, _x);
```

`setProperty()` 関数では、次の例で示しているように、ムービークリップインスタンスのプロパティを設定できます。

```
setProperty(cartoonArea, _x, 100);
```

次の例は、前の例と同じですが、ドットシンタックスを使用しています。

```
cartoonArea._x = 100;
```

また、`tellTarget()` ステートメント内からムービークリッププロパティを取得または設定できます。次のコードは、上記の例で示した `setProperty()` と等しいものです。

```
tellTarget("/cartoonArea") {  
    _x = 100;  
}
```

`tellTarget()` 関数の詳細については、[12 ページの「その他のタイムラインの制御」](#)を参照してください。

その他のタイムラインの制御

タイムラインへのパスを指定するには、ドット (`.`) と組み合わせたスラッシュシンタックス (`/`) を使用してパス参照を構築します。また、Flash 5 の表記である `_levelN`、`_root`、`_parent` を使い、特定のムービーレベル、アプリケーションのルートタイムライン、親タイムラインをそれぞれ参照することもできます。

たとえば、SWF ファイルのメインタイムラインに `box` というムービークリップインスタンスがある場合を想定してみます。この `box` インスタンスには、`cards` という別のムービークリップインスタンスが含まれています。次の例は、どのタイムラインからでも、ムービークリップ `cards` をターゲットにしています。

```
tellTarget("/box/cards")  
tellTarget("_level0/box/cards")
```

次の例ではムービークリップ `cards` から、メインタイムラインをターゲットにしています。

```
tellTarget(".././cards")  
tellTarget("_root")
```

次の例では親ムービークリップ `cards` をターゲットにしています。

```
tellTarget("../cards")  
tellTarget("_parent/cards")
```

変数の使用

タイムラインの変数を指定するには、ドット (.) およびコロン (:) と組み合わせたスラッシュシンタックス (/) を使用します。次のドット表記を使用することもできます。

次のコードは、メインタイムラインの car 変数を参照します。

```
/:car  
_root.car
```

次の例は、メインタイムラインにある入れ子になったムービークリップインスタンスの car 変数を表しています。

```
/mc1/mc2/:car  
_root.mc1.mc2.car
```

次の例は、現在のタイムラインにあるムービークリップインスタンスの car 変数を表しています。

```
mc2/:car  
mc2.car
```

配列のシミュレート

配列は、変数や値など、情報の順序付けられたリストを作成または操作する際に便利です。ただし、Flash Lite 1.1 では、ネイティブの配列データ構造をサポートしていません。Flash Lite (および Flash 4) プログラミングでの一般的なテクニックは、ストリング処理で配列をシミュレートすることです。シミュレートされた配列は、疑似配列とも呼びます。疑似配列処理の鍵は eval() ActionScript 関数であり、この関数を使用することで、変数、プロパティ、ムービークリップに名前によってアクセスできます。詳細については、[19 ページの「eval\(\) 関数の使用」](#)を参照してください。

通常、疑似配列は、同一の基本名の後に数値の接尾辞が続く 2 つ以上の変数で構成されます。接尾辞は、各配列要素のインデックスです。

たとえば、次の ActionScript 変数を作成するとします。

```
color_1 = "orange";  
color_2 = "green";  
color_3 = "blue";  
color_4 = "red";
```

このとき、次のコードを使用して疑似配列で要素をループさせることができます。

```
for (i = 1; i <=4; i++) {  
    trace (eval ("color_" + i));  
}
```

既存変数の参照に加え、変数代入の左側に eval() 関数を使用して、実行時に変数を作成できます。たとえば、ゲームをプレイしたときのハイスコアのリストを保持したいとします。ユーザーが 1 回のゲームを終えたときに、毎回リストにユーザーのスコアを追加します。

```
eval("highScore" add scoreIndex) = currentScore;
scoreIndex++;
```

このコードは、実行するたびに、ハイスコアのリストに新しい項目を追加してから `scoreIndex` 変数を加算します。この変数はリスト内の各項目のインデックスを決定します。たとえば、次の変数で終了するとします。

```
highScore1 = 2000
highScore2 = 1500
highScore3 = 3000
```

テキストおよびストリングの操作

Flash Lite では、テキストの操作を行うためのいくつかの基本的な `ActionScript` コマンドとプロパティを提供しています。テキストフィールドの値の取得と設定、ストリング、URL エンコード、または URL デコードテキストストリングの連結、およびスクロールするテキストフィールドの作成ができます。

このセクションでは、次のトピックについて説明します。

- [14 ページの「ストリングの連結」](#)
- [14 ページの「スクロールテキスト」](#)

ストリングの連結

Flash Lite でストリングを連結するには、次の例に示されているように `add` 演算子を使用します。

```
city = "Boston";
team = "Red Sox";
fullName = city add " " add team;
// 結果 :
// fullName = "Boston Red Sox"
```

スクロールテキスト

ダイナミックおよびテキスト入力フィールドの `scroll` プロパティを使用すると、フィールドの現在のスクロール位置を取得または設定できます。また、`maxscroll` 位置を使用して、テキストフィールドの現在のスクロール位置を最大スクロール位置に対して相対的に決定することができます。詳細については、『Flash Lite アプリケーションの開発』の「スクロールテキストの作成」を参照してください。

call() 関数を使用しての関数作成

Flash Player 5 およびそれ以降で行っているカスタム関数の定義や呼び出しは、Flash Lite で行うことはできません。ただし、call() ActionScript 関数を使用して、タイムラインの任意のフレームにあるコードを実行できます。この方法により、頻繁に使用されるコードを 1 か所でカプセル化することができ、メンテナンスが簡単にできるようになります。

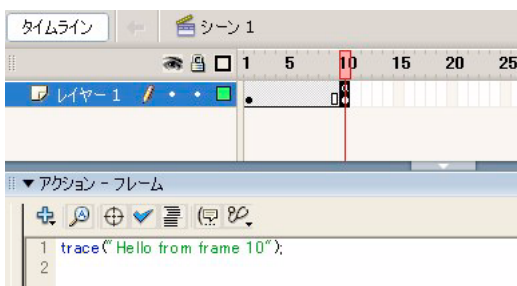
call() 関数では、フレーム番号またはフレームラベルをパラメータとして使用します。たとえば、次の ActionScript は、moveUp というラベルのフレームにあるコードを呼び出します。

```
call("moveUp");
```

call() 関数は同期化して動作し、call() 関数呼び出しに続く ActionScript は、指定されたフレームの ActionScript がすべて実行されるまで、実行されません。

別のフレームの ActionScript を呼び出すには：

1. 新しい Flash ドキュメントで、フレーム 10 にキーフレームを挿入します。



2. 新しいキーフレームが選択されている状態で、[アクション] パネルを開いて ([ウィンドウ] - [アクション])、次のコードを入力します。

```
trace("Hello from frame 10");
```

3. フレーム 1 のキーフレームを選択し、[アクション] パネル内で次のコードを入力します。

```
stop();  
call(10);
```

このコードにより、フレーム 1 で再生ヘッドが停止し、フレーム 10 のコードが呼び出されます。

4. Adobe® Device Central™ でアプリケーションをテストします。

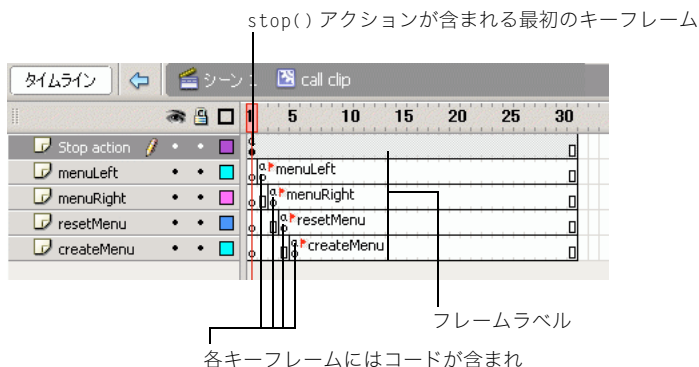
エミュレータ上に "Hello from frame 10" と表示されます。

ムービークリップのタイムラインなど、別のタイムラインにあるコードを呼び出すこともできます。コードを実行するには、ムービークリップインスタンス名とコロン、およびフレーム番号またはラベルを指定します。たとえば、次の ActionScript では、callClip という名前のムービークリップインスタンスで moveUp というラベルのフレームにあるコードを呼び出します。

```
call("callClip:moveUp");
```

この方法は、呼び出しクリップあるいは関数クリップと呼ばれる、定期的に変更されるコードをカプセル化することを唯一の目的とするムービークリップを作成するのに頻繁に使用されます。呼び出しクリップには、作成する各関数用のキーフレームが含まれています。通常は、キーフレームの目的に従ってそれぞれにラベルを付けていきます。また、新規のキーフレームそれぞれに対して新しいレイヤーを作成し、そのレイヤーに対してキーフレームに割り当てたフレームラベルと同じ名前を割り当てることをお勧めします。

次の図は例として取り上げた呼び出しクリップのタイムラインを表しています。呼び出しクリップの最初のキーフレームには常に `stop()` アクションが含まれており、これによって再生ヘッドがそのタイムライン内でフレームをループし続けることのないようになっています。後続のキーフレームには、各 "関数" のコードが含まれます。各関数キーフレームには、その機能を表すラベルが付きます。呼び出しクリップの編集と表示を容易にするために、各関数キーフレームは、通常、独立したレイヤーに挿入されます。

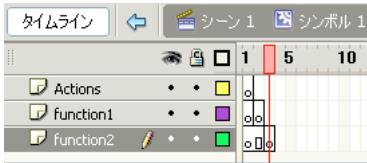


次の手順では、呼び出しクリップの作成方法と使用方法について説明します。

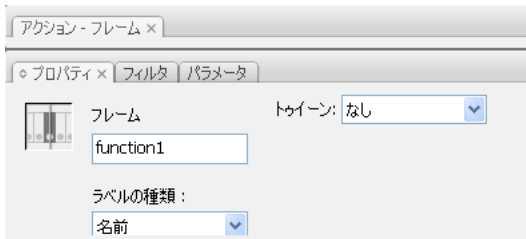
呼び出しクリップを作成し、編集するには：

1. Adobe® Flash® CS3 で、Flash Lite 1.1 Symbian Series 60 ドキュメントテンプレートから新しいドキュメントを作成します。
2. [挿入]-[新規シンボル] を選択します。
3. [新規シンボルの作成] ダイアログボックスの [名前] テキストボックスに「呼び出しクリップ」と入力して [OK] をクリックします。
ムービークリップが編集モードで開きます。
4. [タイムライン] ウィンドウの [レイヤーの追加] ボタンを 2 回クリックして 2 つの新しいレイヤーを挿入します。
最上位レイヤーには **Actions**、2 番目のレイヤーには **function1**、3 番目のレイヤーには **function2** とそれぞれ名前を付けます。

5. 次の図のように、function1 レイヤーのフレーム 2 と function2 レイヤーのフレーム 3 にそれぞれキーフレームを挿入します。

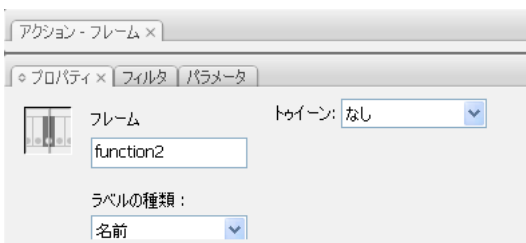


6. Actions レイヤーのキーフレームを選択してから [アクション] パネルを開きます。
7. stop() アクションを [アクション] パネルに追加します。
8. function1 レイヤーのフレーム 2 にあるキーフレームを選択して、次の操作を実行します。
- プロパティインスペクタで、[フレームラベル] テキストボックスに「function1」と入力します。



- [アクション] パネル ([ウィンドウ]-[アクション]) に次のコードを入力します。

```
trace("function1 was called.");
```
9. function2 レイヤーのフレーム 3 でキーフレームを選択し、次の操作を実行します。
- プロパティインスペクタで、[フレームラベル] テキストボックスに「function2」と入力します。



- [アクション] パネル ([ウィンドウ]-[アクション]) に次のコードを入力します。

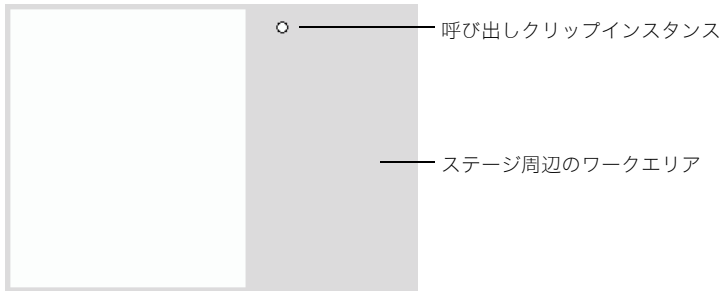
```
trace("function2 was called.");
```
10. Ctrl+E (Windows) または Command+E (Macintosh) を押してメインタイムラインに戻ります。

11. [表示]-[ワークエリア]を選択して、ステージの周りのワークエリアを含めるようドキュメントの表示を設定します。

呼び出しクリップは、ユーザーに表示する必要がないため、ワークエリアに配置できます。

12. [ライブラリ]パネルを開き([ウィンドウ]-[ライブラリ])、呼び出しクリップシンボルをステージ周辺のワークエリアにドラッグします。

呼び出しクリップにはビジュアルエレメントは含まれておらず、ステージ上では、ムービークリップの基準点を表す小さな円として表示されます。



P レ ア	ステージで呼び出しクリップをより確認しやすくするには、呼び出しクリップのタイムラインで最初のキーフレームにテキストや他の表示可能な要素を追加します。
-------------	--

13. プロパティインスペクタで、[インスタンス名]テキストボックスに「callClip」と入力します。
14. タイムラインで、ActionScript というレイヤーのフレーム 1 を選択します。
15. [アクション]パネルに次のコードを入力します。

```
call("callClip:function1");  
call("callClip:function2");
```

16. エミュレータでアプリケーションをテストします([制御]-[ムービープレビュー])。

次のテキストが表示されます。

```
function1 was called.  
function2 was called.
```

eval() 関数の使用

eval() 関数を使用すると、ランタイムに変数とムービークリップインスタンスを動的に参照できます。eval() 関数では、ストリング式をパラメータとして扱い、式によって表される変数の値またはムービークリップの参照を返します。

たとえば、次のコードは ActionScript 変数 name の値を評価して、nameValue にその結果を割り当てます。

```
name = "Jack";
nameValue = eval("name");
// 結果: nameValue = "Jack"
```

Flash Lite はネイティブ配列データ構造をサポートしていないため、eval() 関数は通常、ストリングベースの配列を作成するために、for() ループおよび add (ストリング連結) 演算子と一緒に使用されます。詳細については、[13 ページの「配列のシミュレート」](#)を参照してください。

また、eval() を使用して名前によってムービークリップインスタンスを参照することもできます。たとえば、clip1、clip2、および clip3 という名前の 3 つのムービークリップがあるとします。次の for() ループによって、各クリップの x 位置は 10 ピクセルずつ加算されます。

```
for(index = 1; index <= 3; index++) {
    eval("clip" + index)._x += 10
}
```


この章では、ユーザーのデバイス进行操作する際の一般的な Flash Lite スクリプティング作業について説明します。これには、デバイスの機能情報の取得、通話とテキストメッセージの開始、ネットワークステータスの確認などが含まれます。

この章では、次のトピックについて説明します。

デバイスとプラットフォームの機能の確認	21
Web ページを開く	22
通話の開始	22
テキストメッセージまたはマルチメディアメッセージの開始	23
電子メールメッセージの開始	23
外部 SWF ファイルのロード	24
外部データのロード	24

デバイスとプラットフォームの機能の確認

アドビ システムズ社の Macromedia Flash Lite 1.1 にはいくつかの ActionScript 変数が含まれおり、特定のデバイスで実行されている Flash Lite アプリケーションで利用可能な機能に関する情報を提供できます。たとえば、_capLoadData 変数はデバイスが外部データのロードをサポートするかどうかを示し、_capSMS 変数はデバイスが送信 SMS (ショートメッセージサービス) メッセージをサポートするかどうかを示します。機能変数の一覧については、Flash Lite 1.x ActionScript リファレンスガイドの「機能」を参照してください。

通常、機能変数は、あるデバイスが特定の機能をサポートしているかどうかを、その機能を使用する前に確認するために使用します。たとえば、loadVariables() 関数を使用して Web サーバーからデータをダウンロードするアプリケーションを開発する予定であるとします。データをロードする前に、次のようにまず _capLoadData 変数の値を確認すれば、そのデバイスがその機能をサポートしているかどうかを判断できます。

```
if(_capLoadData == 1) {
    loadVariables("http://foo.com/data.txt");
} else {
    status_message = "Sorry, unable to load external data."
}
```

Flash Lite では、機能変数はメイン SWF ファイルのルートタイムラインで定義されます。そのため、たとえばムービークリップのタイムラインなど、他のタイムラインからこれらの変数にアクセスする場合は、変数までのパスを修飾する必要があります。たとえば、次の例では、スラッシュ (/) を使用して `_capSMS` 変数に対して完全修飾されたパスを提供しています。

```
canSendSMS = /:_capSMS
```

Web ページを開く

`getURL()` コマンドを使用すると、デバイスの Web ブラウザで Web ページを開くことができます。これは、デスクトップの Flash アプリケーションから Web ページを開くのと同一方法です。たとえば、次の手順でアドビ システムズ社の Web ページを開きます。

```
getURL("http://www.adobe.com");
```

Flash Lite は、1つのフレームまたは1つのイベントハンドラで1つの `getURL()` アクションのみを処理します。一部のハンドセットでは `getURL()` アクションはキー押下イベントに限定されており、キー押下イベントハンドラ内でトリガされた `getURL()` 呼び出しだけが処理されます。そのような場合でも、キー押下イベントハンドラごとに処理される `getURL()` アクションは1つだけです。ステージ上のボタンインスタンスに割り当てられる次のコードにより、ユーザーがデバイスの選択ボタンを押すと、Web ページが開きます。

```
on (keyPress "<Enter>"){
    getURL("http://www.adobe.com");
}
```

通話の開始

Flash Lite アプリケーションから通話を開始するには、`getURL()` 関数を使用します。通常、この関数は Web ページを開くのに使用しますが、ここでは `tel:` を (`http` の代わりに) プロトコルとして指定し、次に電話をかける相手の番号を入力します。この関数を呼び出すと確認用のダイアログボックスが表示され、入力された番号に電話をかけてよいか許可を求めてきます。

たとえば、次のコードは電話番号 `555-1212` への通話を開始します。

```
getURL("tel:555-1212");
```

Flash Lite では、フレームまたはイベントハンドラごとに1つの `getURL()` アクションが処理されます。一部のハンドセットでは `getURL()` アクションはキーを押すイベントに限定されており、キー押下イベントハンドラ内でトリガされた `getURL()` 呼び出しだけが処理されます。そのような場合でも、キー押下イベントハンドラごとに処理される `getURL()` アクションは1つだけです。次の例では、ユーザーがデバイスの選択ボタンを押すと通話が開始されます。

```
on (keyPress "<Enter>"){
    getURL("tel:555-1212");
}
```

テキストメッセージまたはマルチメディアメッセージの開始

Flash Lite を使用すると、ショートメッセージサービス (SMS) またはマルチメディアメッセージサービス (MMS) のメッセージを開始することができます。Flash Lite アプリケーションで SMS または MMS メッセージを送信するには、getURL() コマンドを使用し、標準の http プロトコルではなく、sms: または mms: プロトコルに続いて、メッセージ送信先の電話番号を指定します。

```
getURL("sms:555-1212");
```

次の例のように、URL クエリースtring内にメッセージ本文を指定することもできます。

```
getURL("sms:555-1212?body=More info please");
```

MMS メッセージを送信するには、次のように sms: プロトコルではなく mms: プロトコルを使用します。

```
getURL("mms:555-1212");
```



Flash Lite で MMS メッセージの添付ファイルを指定することはできません。

Flash Lite は、1つのフレームまたは1つのイベントハンドラで1つの getURL() アクションのみを処理します。一部のハンドセットでは getURL() アクションはキー押下イベントに限定されており、キー押下イベントハンドラ内でトリガされた getURL() 呼び出しだけが処理されます。そのような場合でも、キー押下イベントハンドラごとに処理される getURL() アクションは1つだけです。次の例では、ユーザーがデバイスの選択ボタンを押すと SMS メッセージが送信されます。

```
on (keyPress "<Enter>"){
    getURL("sms:555-1212");
}
```

電子メールメッセージの開始

Flash Lite を使用して、電子メールメッセージを開始することができます。電子メールメッセージを送信するには、getURL() コマンドを使用し、mailto: プロトコルの後に受信側の電子メールアドレスを指定します。次の例のように、URL クエリースtring内にメッセージの題名と本文を指定することもできます。

```
getURL("mailto:mobile-developer@adobe.com?subject=Flash Lite");
```

クエリースtringでメッセージ本文だけを指定するには、以下のコードを使用します。

```
getURL("mailto:mobile-developer@adobe.com?body=More+info+please");
```

外部 SWF ファイルのロード

loadMovie() 関数を使用することで、ネットワークやローカルファイルから SWF ファイルをロードできます。この機能は Flash Lite 1.1 以降から使用できます。次の補足説明は、外部 SWF ファイルをロードする場合に適用されます。

- Flash Lite では、Flash Lite 1.0 または Flash Lite 1.1 SWF ファイル、あるいは Flash 4 以前の形式の SWF ファイルをロードできます。その他の形式 (たとえば、Flash Player 6 SWF) の SWF ファイルをロードしようとすると、ランタイムエラーが発生します。
- Flash Lite では、JPEG や GIF などの外部イメージファイルを直接ロードすることはできません。これらのタイプのメディアをロードする場合は、イメージデータを SWF ファイル形式に変換する必要があります。この変換は、Flash オーサリングツールを使ってイメージファイルを新しいドキュメントに読み込み、次に Flash Lite または Flash 4 SWF ファイルに書き出すことで、“手動”で実行することが可能です。また、自動変換用のサードパーティユーティリティもあります。

SWF ファイルのロードの詳細については、『Flash Lite 1.x ActionScript リファレンスガイド』の loadMovie() を参照してください。

外部データのロード

外部データを Flash Lite アプリケーションにロードするには、loadVariables() 関数を使用します。データは、ネットワーク (HTTP アドレス) からでも、ローカルファイルシステムからでもロードできます。この機能は Flash Lite 1.1 以降から使用できます。

この項では、loadVariables() 関数を使用して外部ファイルからデータをロードし、ダイナミックテキストフィールドに表示する方法について説明します。最初に、アンパサンド (&) で区切った名前と値のペアを 5 組含んだテキストファイルをデータファイルとして作成します。次に、テキストファイルに含まれているデータをロードし、表示する Flash Lite アプリケーションを作成します。

この例では、データファイルと SWF が、ユーザーのコンピュータ (エミュレータによるテストの場合) かデバイスのメモ리카ード (実際のデバイスによるテストの場合) の同じフォルダに保存されていることを前提としています。デバイスでこのアプリケーションをテストするには、次のいずれかの操作を実行する必要があります。

- テキストファイルをデバイスに転送し、SWF ファイルを同じフォルダに保存します。
- テキストファイルを Web サーバーの URL (例: `www.your-server.com/data.txt`) に送信します。次に、サンプルアプリケーションの呼び出しを変更して、以下のように URL を指定します。

```
loadVariables("http://www.your-server.com/data.txt", "data_clip");
```

ネットワークを経由してデータをロードするアプリケーションの例については、『Flash サンプル』の「Flash Lite ニュースリーダー」を参照してください。

データファイルの作成

1. テキストエディタ (例: Notepad または SimpleText) を使用して、次のテキストを含むファイルを作成します。

```
item_1=Hello&item_2=Bonjour&item_3=Hola&item_4=Buon+giorno&item_5=G'day
```

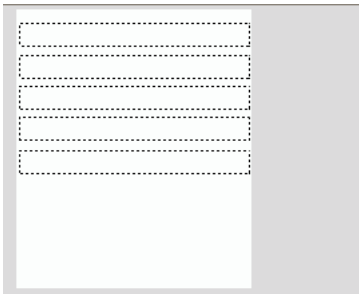
2. ファイルを data.txt として保存します。

データロード用 Flash Lite アプリケーションの作成

1. Flash Lite 1.1 Symbian Series 60 ドキュメントテンプレートから新しいドキュメントを作成します。

Flash Lite ドキュメントテンプレートの使用の詳細については、『Flash Lite 1.x ファーストステップガイド』の「Flash Lite ドキュメントテンプレートの使用」を参照してください。

2. ファイルを、既に作成してあるテキストファイル (data.txt) と同じフォルダに "dataloading fla" として保存します。
3. タイムラインで、Content レイヤーのフレーム 1 を選択します。
4. テキストツールを使用して、次の図に示されているように、ダイナミックテキストフィールドを 5 つ、ステージ上に作成します。



5. 最初の (一番上の) テキストフィールドを選択し、プロパティインスペクタの [変数] テキストボックスに item_1 と入力します。
この変数名は、前に作成した "data.txt" ファイルで定義した最初の変数の名前 (item_1=Hello) に対応しています。
6. 前の 2 つの手順で説明したのと同じ方法で、残る 4 つのテキストフィールドに、item_2、item_3、item_4、item_5 という変数名を割り当てます。
7. Shift キーを押しながら各テキストフィールドを選択して全選択とし、[修正]-[シンボルに変換] を選択します。
8. [シンボルに変換] ダイアログボックスで、シンボルタイプとして [ムービークリップ] を選択し、[OK] をクリックします。

9. プロパティインスペクタで作成したムービークリップを選択し、[インスタンス名] テキストボックスに **data_clip** と入力します。
10. タイムラインで Actions レイヤーのフレーム 1 を選択し、[アクション] パネルを開きます ([ウィンドウ]-[アクション])。
11. [アクション] パネルで、次のコードを入力します。

```
loadVariables("data.txt", "data_clip");
```
12. 変更を保存し ([ファイル]-[保存])、エミュレータでアプリケーションをテストします ([制御]-[ムービープレビュー])。
次の図のように、各テキストフィールドにはテキストファイルのデータが入力されます。



索引

C

call() 関数、使用 15

E

eval() 関数、使用 19

F

Flash Lite 1.x ActionScript

1.0 と 1.1 の違い 8

概要 7

サポートされていない Flash 4 ActionScript 8

使用できない機能 9

G

getURL() 関数

Web ページを開く 22

通話の開始 22

テキストメッセージの開始 23

電子メールメッセージの開始 23

マルチメディアメッセージの開始 23

L

loadMovie() 関数、使用 24

loadVariables() 関数、使用 24

W

Web ページ、開く 22

か

開始、通話 22

外部 SWF ファイル、ロード 24

外部データ、ロード 24

関数クリップ、作成 15

関数、call() によるエミュレート 15

く

配列、ストリングによるエミュレート 13

す

スクロールテキスト、作成 14

ストリング、連結 14

た

タイムライン、ActionScript による制御 12

つ

通話、開始 22

て

テキストメッセージ、開始 23

電子メールメッセージ、開始 23

ひ

開く、Web ページ 22

ふ

プラットフォーム機能変数、説明 21

へ

変数

参照 13

動的参照 19

ドットシンタックスとスラッシュシンタックス 13

ま

マルチメディアメッセージ、開始 23

む

ムービークリップ

取得と設定、プロパティ 11

動的参照 19

め

メッセージ、開始 23

れ

連結、ストリング 14