

September 2009

Getting started with Project Capuchin

for Flash Lite developers



Preface

About this tutorial

The purpose of this document is to show Flash Lite developers how to create a simple Project Capuchin application, the Contacts application, using the Contacts MXP service. The major steps to be taken by Flash Lite developers while going through this tutorial are:

- Downloading the necessary MXP files needed for this tutorial.
- Installing the components.
- Creating a new Project Capuchin application. This will include using ActionScript to implement the Contacts application, importing the necessary classes, invoking function handlers, creating a Flash GUI, and so on.
- Compiling, testing and publishing the application.

Sony Ericsson Developer World

At www.sonyericsson.com/developer, developers find the latest technical documentation and development tools such as phone White papers, Developers guidelines for different technologies, Getting started tutorials, SDKs (Software Development Kits) and tool plugins. The Web site also features news articles, go-to-market advice, moderated discussion forums offering free technical support and a Wiki community sharing expertise and code examples.

For more information about these professional services, go to the Sony Ericsson Developer World Web site.

This Tutorial is published by:

Sony Ericsson Mobile Communications AB,
SE-221 88 Lund, Sweden

www.sonyericsson.com/

© Sony Ericsson Mobile Communications AB,
2009. All rights reserved. You are hereby granted
a license to download and/or print a copy of this
document.
Any rights not expressly granted herein are
reserved.

Second revised edition (September 2009)
Publication number: 1225-2326.2 (rev. B)

This document is published by Sony Ericsson Mobile Communications AB, without any warranty*. Improvements and changes to this text necessitated by typographical errors, inaccuracies of current information or improvements to programs and/or equipment, may be made by Sony Ericsson Mobile Communications AB at any time and without notice. Such changes will, however, be incorporated into new editions of this document. Printed versions are to be regarded as temporary reference copies only.

*All implied warranties, including without limitation the implied warranties of merchantability or fitness for a particular purpose, are excluded. In no event shall Sony Ericsson or its licensors be liable for incidental or consequential damages of any nature, including but not limited to lost profits or commercial loss, arising out of the use of the information in this document.

Typographical conventions

In this document code examples are written in Courier font:

```
contact.setFirstName(entries[0].getFirstName());
```

Names of labels, buttons and menus are written in *italics*, for example, *Select File – Install extension...*

Names of edited values, file names and input text are within quotes, for example, “MXP files.zip”

Trademarks and acknowledgements

Adobe, Adobe Flash Lite and Adobe Flash are either trademarks or registered trademarks of Adobe Systems Incorporated in United States and/or other countries.

Java and all Java based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc, in the U.S. and other countries.

Other product and company names mentioned herein may be the trademarks of their respective owners.

Document history

Change history		
2009-03-04	Doc. no. 1225-2326.1	First version published on Developer World
2009-08-20	Doc. no. 1225-2326.2	Second version. Example based on the Contacts application
2009-09-03	Doc. no. 1225-2326.2 (rev. B)	Second revised version. Minor changes

Contents

Tutorial	5
Prerequisites	5
Flash Lite developer knowledge areas	5
Applications and tools	5
Downloading the necessary MXP files	6
Installing the components	6
Creating the project	8
Setting up a new project	8
Creating your application in Flash IDE	9
Publishing your project	19
Further activities	20
Related links	20

Tutorial

Prerequisites

The applications and skills listed below are required to complete the example in this tutorial.

Flash Lite developer knowledge areas

Flash Lite developers should have knowledge in the following areas:

- Working with Adobe Flash CS3/CS4. (In the example of this tutorial, Adobe Flash CS3 is used)
- ActionScript 2.0 programming (more information can be found at <http://www.adobe.com/devnet/actionscript/>).
- Creating Flash animations (more information can be found at <http://www.adobe.com/devnet/flash/>).

Applications and tools

Below applications and tools are required for the Flash developer to be able to complete the work.

Adobe Flash CS3/CS4

A multimedia authoring application used to create web applications, games, movies, and content for embedded devices. It features support for vector and raster graphics and ActionScript.

<http://www.adobe.com/products/flash/>

Adobe Extension Manager

Adobe tool devoted to install new extensions for Adobe Flash CS3/CS4.

http://www.adobe.com/exchange/em_download/

Java runtime environment (JRE)

The latest version of JRE and installation instructions are available at

<http://www.java.com/en/download/manual.jsp>

Downloading the necessary MXP files

- Download the Capuchin Kit MXP file from Developer World at:
http://developer.sonyericsson.com/site/global/docstools/projectcapuchin/p_projectcapuchin.jsp.
Go to the Flash Lite tab and look for the following title: "Project Capuchin Kit". Right below the title you find links to the actual Capuchin MXP files for both PC and Mac.

By downloading this file you will get the core Project Capuchin MXP service. It provides a seamless bridge between your Flash Lite™ application and the platform. It is required by all the other service MXPs. In addition, CapuchinKit.MXP extends your Adobe® Flash CS3 and CS4 to support publishing your Flash content as a Project Capuchin application.

- Download the Contacts service MXP file from Developer World at:
http://developer.sonyericsson.com/site/global/docstools/projectcapuchin/p_projectcapuchin.jsp.
Go to the "Services MXPs" tab, scroll down and look for the Contacts MXP file.

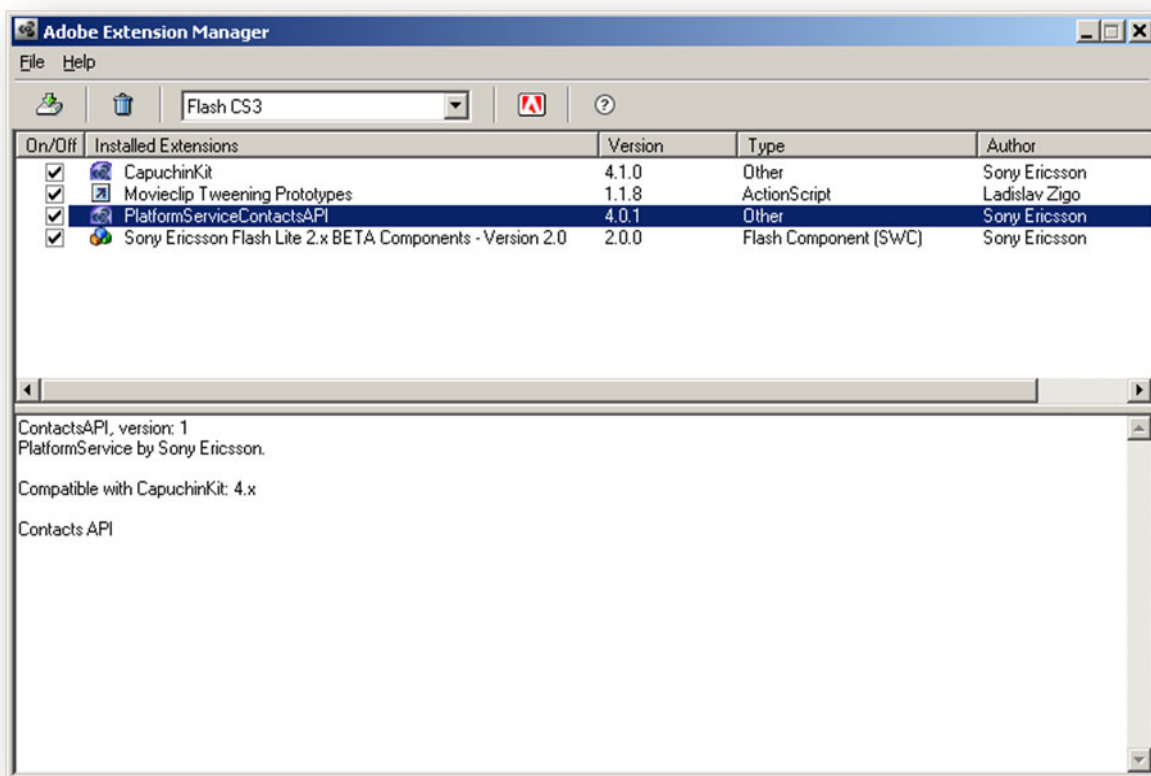
By downloading this file you will get the Contacts service which allows the access and control of the Phone's address book. With this service, the user can add, remove, update and list all the contacts on the phone.

Installing the components

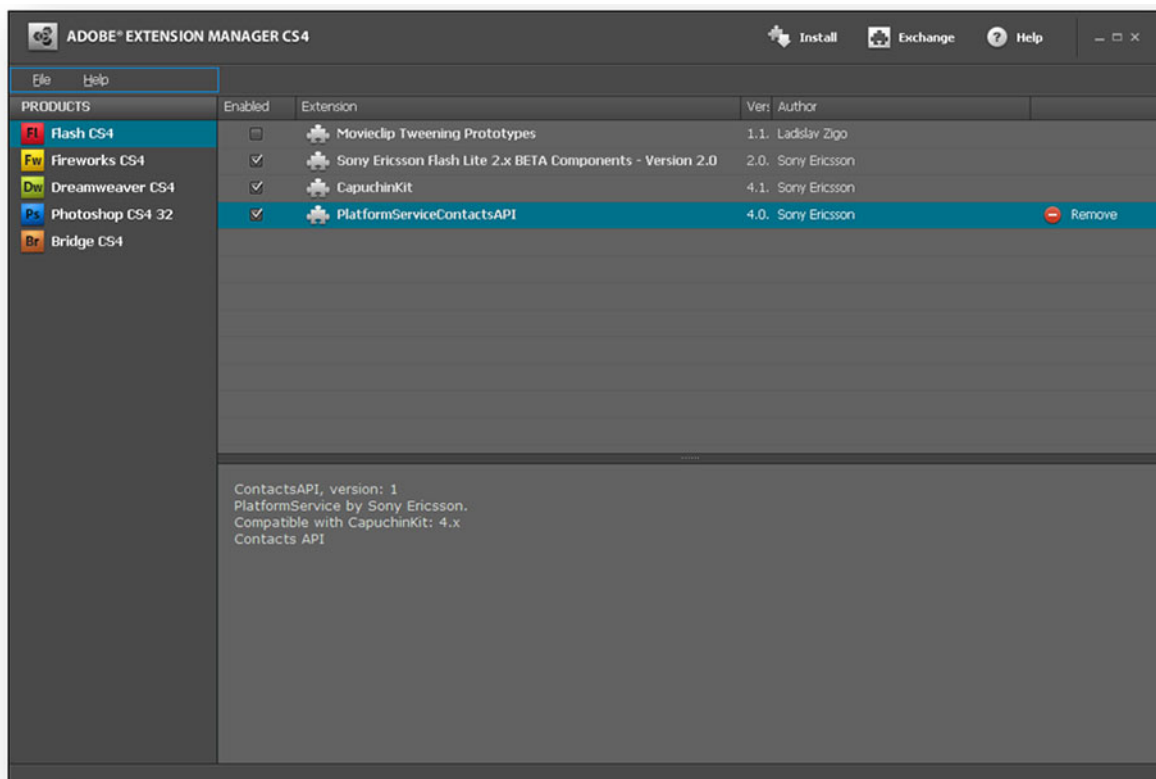
- Install the "CapuchinKit.MXP" and "Contacts.MXP" by double clicking on each MXP file.

Note: If you have both Adobe Flash CS3 and CS4 installed on your computer, and you are installing the components in Flash CS4, it is recommended to launch the Adobe Extension Manager CS4 from the Flash main menu. From the main menu, select *Help – Manage Extensions*. When the Extension Manager CS4 is open, click on the *Install* icon in the top right of the window to add your components to the Flash IDE.

- After a successful installation, the MXP packages are listed in the Adobe Extension Manager, MXP packages installation tab, as in the images below.



Components installed by Adobe Extension Manager CS3



Components installed by Adobe Extension Manager CS4

Creating the project

In this tutorial focus is on a particular Platform Service that is called "Contacts", so all references to the Platform Service are through the "Contacts" name. Accordingly a simple Contact application is created where feedback is displayed on the screen when data are retrieved from the Contacts list. Graphical and design areas of the application development process are not covered here. Only the mechanisms for developing Project Capuchin applications, such as importing the classes, reading data and updating the view are discussed. However, the same concept can be used to develop more complex logics and appealing user interfaces by combining different MXP packages, ActionScript programming and Sony Ericsson UI components.

Setting up a new project

In this chapter a Capuchin application is created using the "Contacts" Platform Service functionality. There are two steps in the process for creating your application using the installed MXP packages:

- Creating your application in your Flash IDE and then running or debugging it on Adobe Device Central. Fake data and dummy values are read from an XML file when the application is running in the emulator. It is possible to alter these data if needed.
- Capuchin application to run on phones, that is, a Java MIDlet application packaged in a jar file. JAR (Java Archive) is generally used to distribute Java classes and associated metadata. When the application is running on a phone, it automatically retrieves data from Java instead of the XML file.

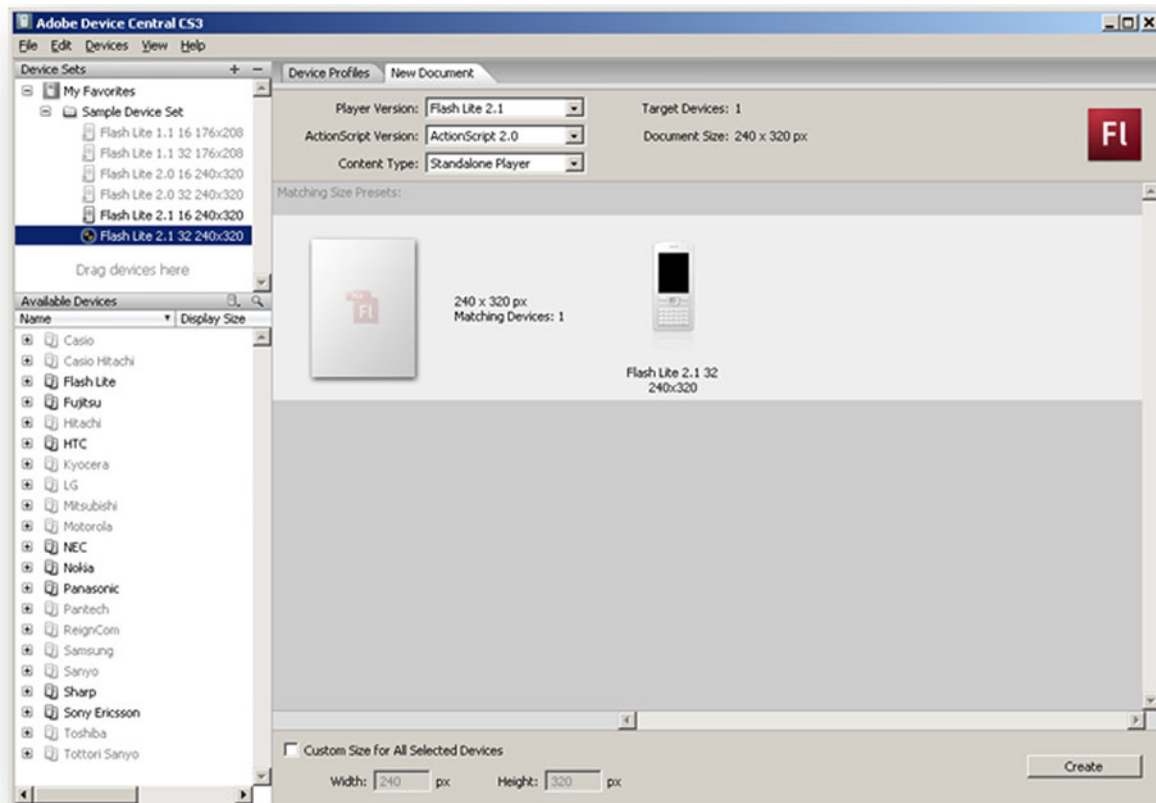
Creating your application in Flash IDE

Step 1 – Create a new project

Open Adobe Device Central CS3/CS4. Select *File – New document* from the menu and select *Flash File (Mobile)*. The “New document” pane opens.

Step 2 – Specify the document properties

Select *Flash Lite 2.1* from the *Player Version* drop-down list, and *ActionScript 2.0* from the *ActionScript Version* drop-down list. *Content Type* should be set to *Standalone Player*. In the Device Sets panel to the left, select *Flash Lite 2.1 32 240x320*. The image below shows Adobe Device Central CS3 with the FLA document settings.



Click the *Create* button to open Adobe Flash IDE with a new Flash Lite document (.FLA file) created.

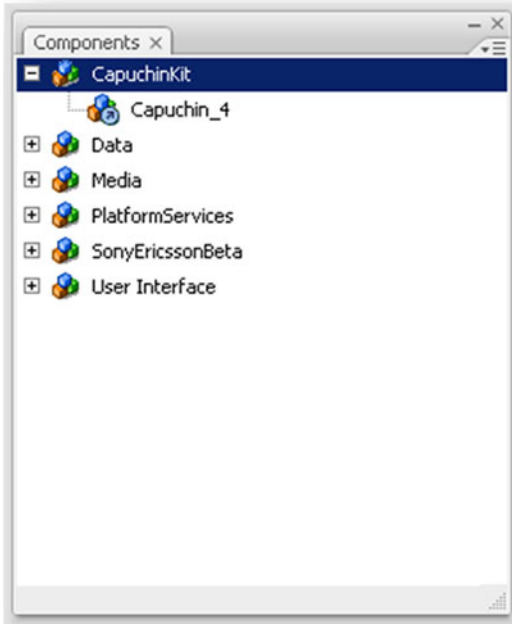
Step 3 – Save the project

Save the new FLA document as "Contacts fla" in the directory where you want to store your Project Capuchin applications, for example, "C:/CapuchinApps/".

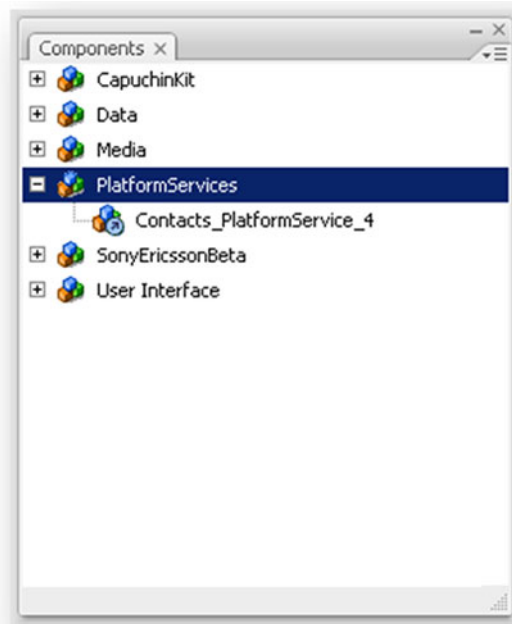
Step 4 – Add Capuchin components to your Flash library

The two installed MXP packages, "CapuchinKit.mxp" and "Contacts.mxp", must be available when creating a Capuchin application. For that reason they have to be added to your Flash library.

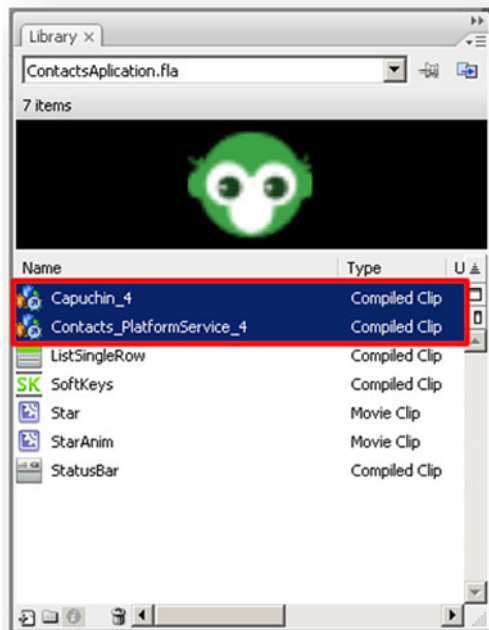
While the "Contacts fla" document is open in Adobe Flash CS3/CS4, select *Window – Components* from the menu or press *CTRL+F7* to open the Adobe Flash CS3/CS4 Components window, see image below, expand CapuchinKit, and double-click on *Capuchin_4*.



In the same way, add *Contacts_PlatformService_4* component for PlatformServices in the Components window, see image below. Save your changes.



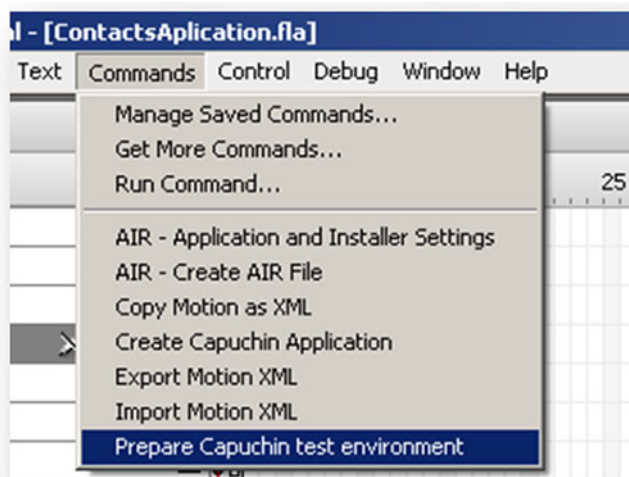
Select *Window – Library* from the menu to open the Adobe Flash CS3 Library window and verify that Capuchin_3_emu and Contacts_PlatformService_4 have been added to the Library, see image below



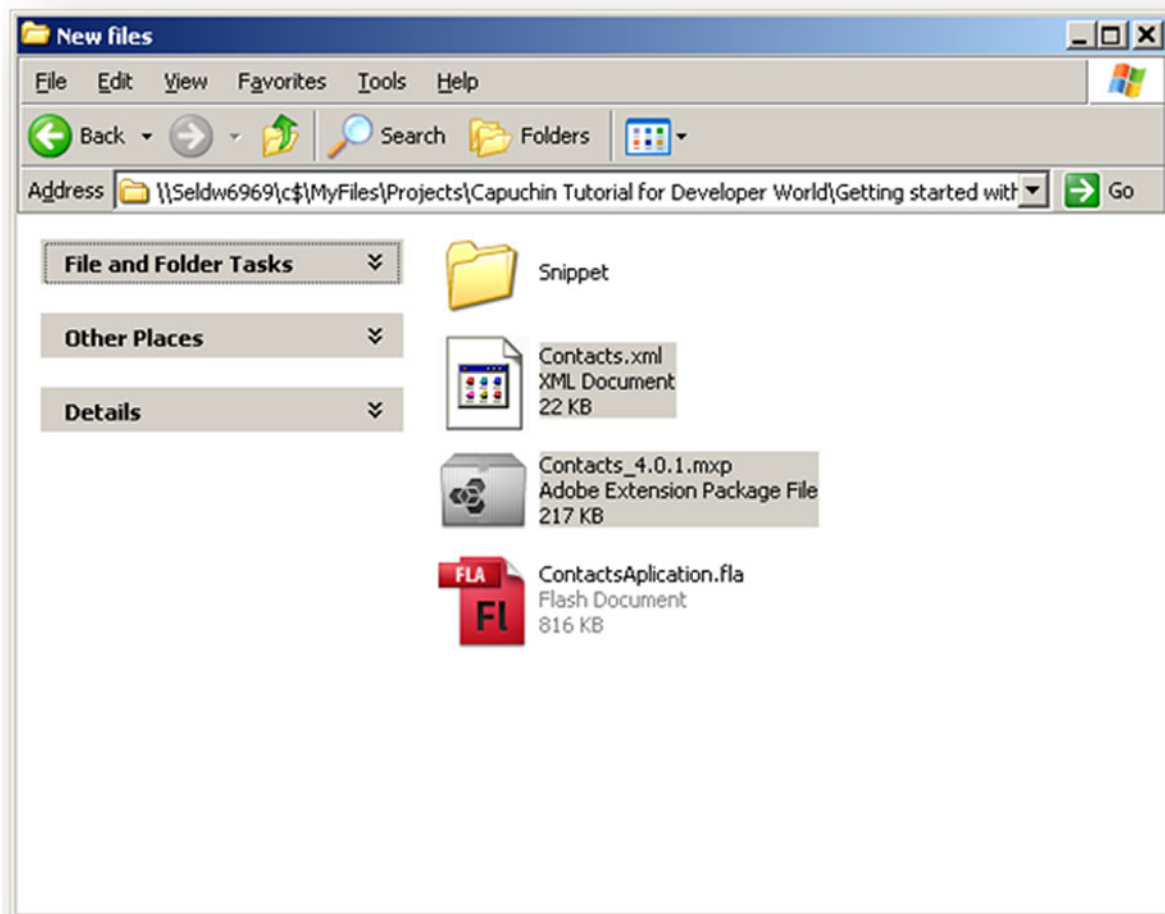
Step 5 – Working with the XML file

As previously mentioned, the emulator application requires an XML file that simulates response data from the platform side. The XML file contains fixed data that on a device would be sent from Java to Flash.

To generate the adequate service XML file, select *Commands – Prepare Capuchin test environment* from the main menu. See image below.



Note: Every Platform Service has its own XML file with a name matching that of the API. In this example the file is named "Contacts.xml", see image below. The emulator recognises which file to read by its name, so the XML file has to have a name matching that of a Platform Service and must not be renamed.



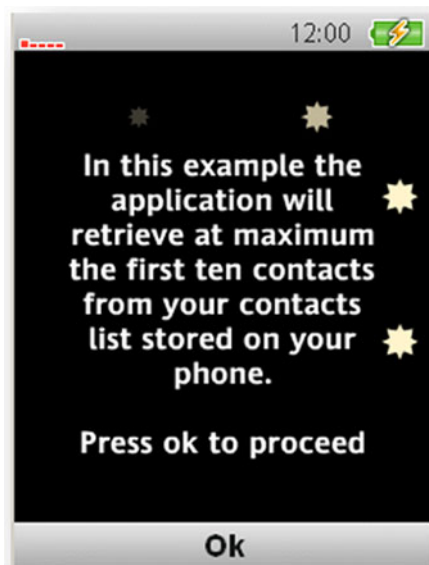
The test values stored in the XML file provide a mechanism for the Flash developer to test the application without having an actual phone. Each method call in ActionScript returns dummy values defined in the XML file.

Note: Input parameters of called methods are not taken into consideration. Returned values always remain the same, regardless of passed parameters.

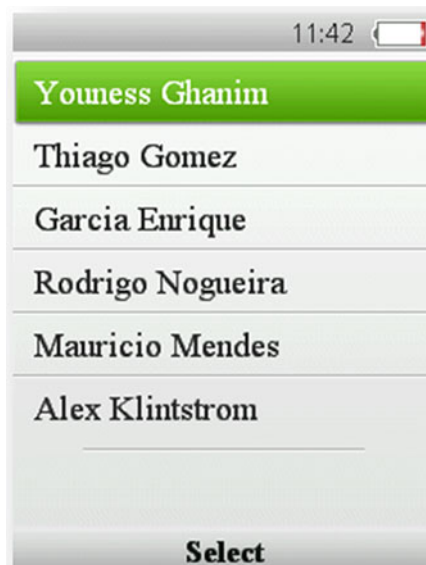
Step 6 – Add objects to the stage

On the Flash stage, there are already two components added during step 4. More symbols for presenting the contacts on the screen may also be created. For example, a view with some nice GUI using Sony Ericsson UI components and some animation can be added. For this tutorial the Soft Keys, Status Bar and List components were added. Sony Ericsson UI components can be downloaded from Developer World, Flash Lite section at http://developer.sonyericsson.com/site/global/docstools/flashlite/p_flashlite.jsp.

On the first view (frame one) we placed some instructions on what the application will do and how to start it. Some animated starts that will move around the text have been added. A specific area may also be designated for data to be shown and easily read by the user. On the second view there can be several contact names to retrieve from the phone. One way to present this data is by using the list component where each element of the list is fed from an array holding the retrieved values from the phone. The images below are snapshots of the first and second views in the Capuchin Contacts example running in Adobe Device Central.

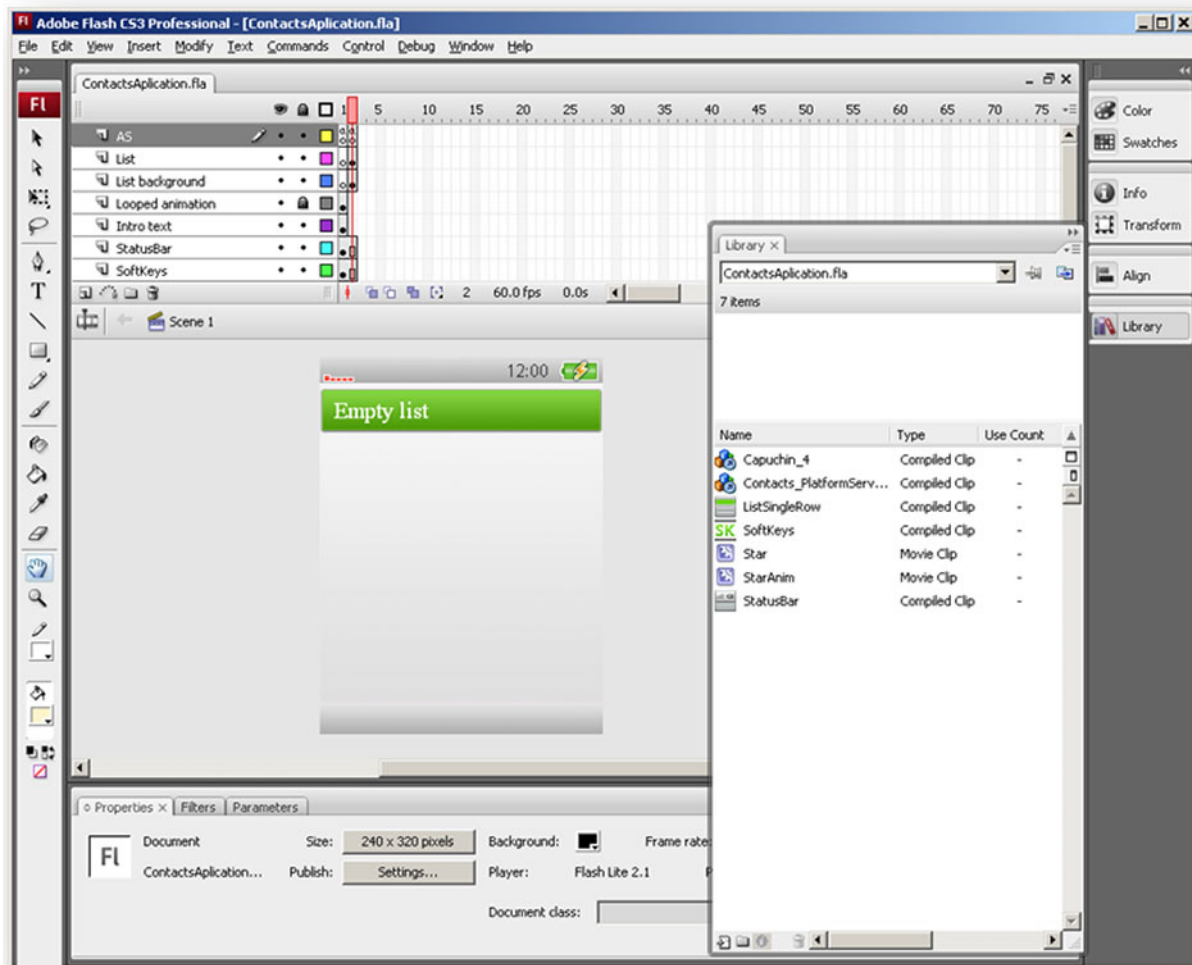


First view for the application running in Device Central



Second view for the application running in Device Central

The dynamic data for each list element is retrieved through `entries`. This is an array created in ActionScript and used as a parameter for the handler function `onGetEntries()` to report what contacts have been received from the phone.



The image above shows the seven layers holding the components and the movie clips needed for building the scene. The seventh layer (first in the list) is the ActionScript layer, holding all the code that handles the application logic and simulates reading data from Java.

Note: The Capuchin Kit and Contacts MXP components must not necessarily be placed on the stage if they are included in your Adobe Flash Library.

Step 7 – Write the code

Select the frame in the ActionScript layer (labelled “AS”), and add some code there. To open the ActionScript editor, select *Window – Actions* from the menu or select the frame and press **F9**.

In the first part of the code on frame two, the Contacts class and the associated data types are imported. For this tutorial the following classes are imported. Though the `ContactFilter` object is set to `null` later in the code, it was used just for educational purposes:

```
import com.sonyericsson.capuchin.contacts.Contacts;
import com.sonyericsson.capuchin.contacts.datatypes.ContactFilter;
```

Next follows variable declarations and soft keys initialization:

```
// Declaring variables
var filter:ContactFilter = null // Filter is set to null if entries should not
be filtered
var startIndex:Number = 0; // startIndex defines start index number of contacts
to be returned.
var maxItems: Number = 4; // maxItems defines maximum number of contacts to be
returned.
```

`filter` is an object of type `ContactFilter`. It is used to specify searched values of concrete fields and it can be set to `null` if entries should not be filtered, as is the case in this example. `startIndex` and `maxItems` define the starting index and maximum number of contacts to be returned.

Next the middle soft key label is set to “Select” and the right soft key label is set to “Quit”. The Select label was added in case you may want to expand on this example so that each time you press Select, a small pop up window appears and displays more data about the selected contact item in the list.

Then the `Contacts` method called `getEntries()` is invoked with five arguments, the owner of the handler and the actual name of this handler, then follows the filter, starting index and the maximum number of contacts to retrieve.

```
Contacts.getEntries(this, onGetEntries, filter, startIndex, maxItems);
```

The ActionScript code is as follows (can be copied and pasted from here):

```
// Importing the necessary classes
import com.sonyericsson.capuchin.contacts.Contacts;
import com.sonyericsson.capuchin.contacts.datatypes.ContactFilter;

// Declaring variables
var filter:ContactFilter = null // Filter is set to null if entries
should not be filtered
var startIndex:Number = 0;
// startIndex defines start index number of contacts to be returned.
var maxItems: Number = 4;
// maxItems defines maximum number of contacts to be returned.

// Initializing the Soft Keys component
SESoftKeys._MSK = "Select";
// Set middle Soft Key label to Select
SESoftKeys._RSK = "Quit";
// Set right Soft Key label to Quit
```

```

// Invoking the getEntries static method to get the contacts from the phone
Contacts.getEntries(this, onGetEntries, filter, startIndex, maxItems);

// Attach and show wait indicator (before data are loaded)
this.attachMovie("WaitIndicator","NewWaitIndicator",1,{_x:103,_y:142});
NewWaitIndicator._shape = "circle";

// The call back handler, invoked once data are retrieved
function onGetEntries(owner:Object, _status:Boolean, entries:Array)
{
    // Remove wait indicator
    NewWaitIndicator.removeMovieClip();

    if (_status && entries[0] != null) // This means if everything was ok on
the phone (Java side) and we have something to show, then execute the following
lines
    {
        // Create a temporary array, needed by the List component. Go to Sony
Ericsson Developer World, Flash Lite page for more information on the List
component
        var entriesList:Array = new Array();
        for (var i:Number = 0; i < entries.length; i++)
        {
            // Retrieves values from entries and feeds them to the temporary
array
            entriesList[i] = entries[i].firstName + " " +
entries[i].lastName;
        };
        // List component gets data from the temporary array
        ContactsList._text = entriesList;
    }
    else
    {
        // Display warning message when nothing is retrieved from the phone
        _root.attachMovie("Dialogue","NewDialogue",2);
        NewDialogue._titleText = "Data error!";
        NewDialogue._bodyText = "Empty contact list or issues with reading
data! from the phone device!";
        NewDialogue._icon = "error";
        NewDialogue._placement = "center";
        NewDialogue.show();
    }
}

// Quit the application when the right soft key is pressed (works newer phones
only)
SESoftKeys.onSoftKeyDown = function(softKey:String)
{
    if(softKey == "RSK") fscommand2("quit");
}

```


Step 8 – "Contacts" Platform Service help

Platform Service API help can be found by selecting *Help – Flash Help* from the menu and heading to the book named "Contacts Component". By right-clicking the API method in the ActionScript frame and selecting *View help* from the context menu, help for the specific method is opened. Apart from the method signature description, the method handler specification is also found there.

All Platform Service methods are asynchronous. When the caller invokes one of the asynchronous functions, no action is taken immediately. Instead, the action is taken at some future point in time. After the action is taken, a handler method is invoked.

Custom actions can be useful while writing code. For example, if you start typing "com." (for com.sonyericsson.capuchin...) and then press *Ctrl + Space*, a context menu will pop up with some auto complete suggestions. It is also possible to double-click on one of the methods in the Capuchin Platform Services directory in the "Actions - Frame" pane to place it in the Actions editor together with a popup containing syntax help.

Flash Lite 2.1 ActionScript

```

1 import com.sonyericsson.capuchin.contacts.Contacts;
2 import com.sonyericsson.capuchin.contacts.datatypes.ContactFilter;
3
4 var filter: ContactFilter = null; // Filter set to null if entries should not be filtered
5 var startIndex:Number = 0; // startIndex defines start index number of contacts to be returned
6 var maxItems: Number = 4; // maxItems defines maximum number of contacts to be returned
7 SESoftKeys._MSK = "Select"; // Set middle soft key label to Select
8
9 // Invoking the getEntries static method to get the contacts from the phone
10 Contacts.getEntries(this, onGetEntries, filter, startIndex, maxItems);
11
12 // The call back handler, invoked once data are retrieved
13 function onGetEntries(owner:Object, _status:Boolean, entries:Array)
14 {
15     if (_status) // This means if everything was ok on the phone (Java side), then execute
  
```

Press "F1" to see the help files

import com.sonyericsson.capuchin.contacts.*;
import com.sonyericsson.capuchin.contacts.datatypes.*;
var param:Array = new Array("lastName1", "lastName");
var filter: ContactFilter = new ContactFilter(param);
var startIndex:Number = 0;
var maxItems: Number = 10;
Contacts.getEntries(this, onGetEntries, filter, startIndex, maxItems);

Contact entries are then sent from java side to onGetEntries callback function in Flash. Contact entries will be passed as a parameter if everything goes fine in Java.

```

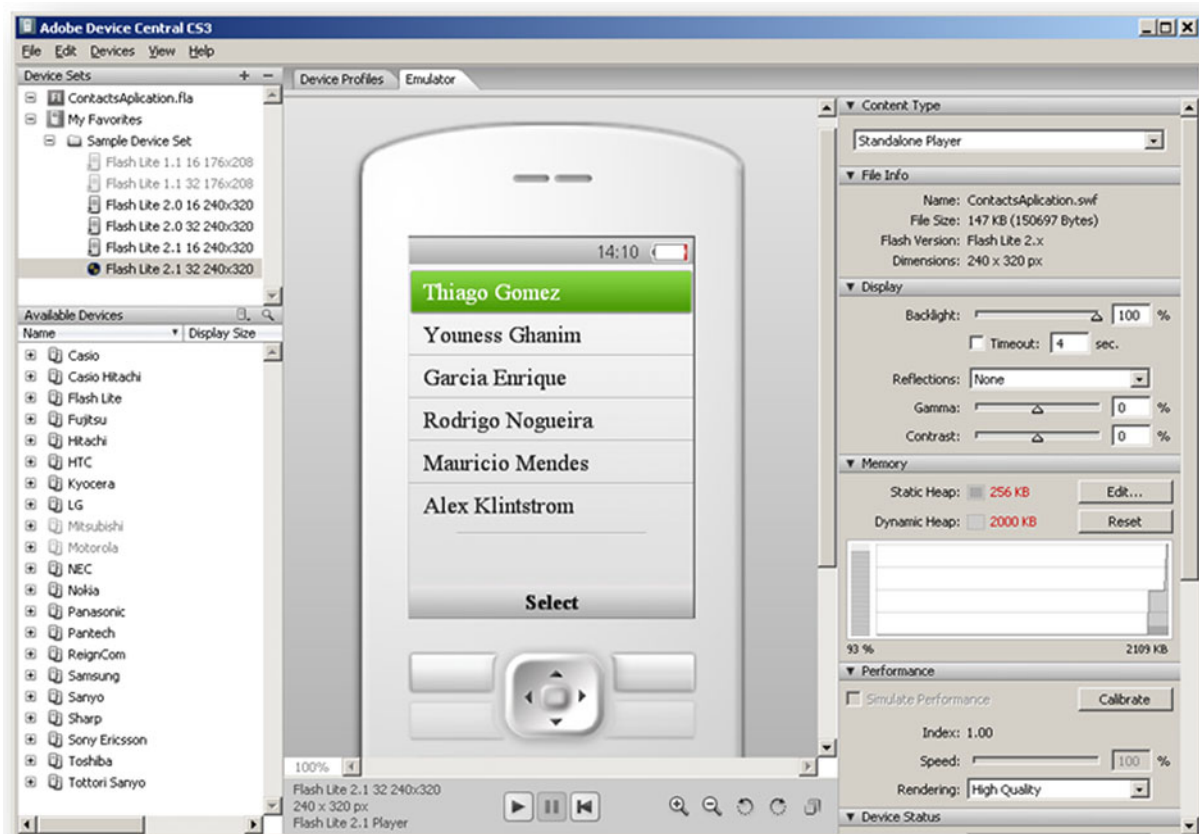
function onGetEntries (owner:Object, status:Boolean, entries:Array) {
    trace("\nonGetEntries:\n");
    if (status == false) {
        trace("Status = " + status + " failed to retrieve contacts");
    } else {
        trace(" owner: " + owner.toString() + ", "
            + " status: " + status + ", "
            + " entries: " + entries + ".");
    }
}
  
```

Parameters

Parameter	Type	Description
owner	Object	
handlerFunction	Function	Method handler.
filter	ContactFilter	Properties to be used when filtering contacts. If ContactFilter is set to null, all entries will be returned.
startIndex	Number	The inclusive starting point of the collection range. 0 means the first item. This should always be >=0.
maxItems	Number	The maximum number of items to return. If this is set to -1, all items will be collected

Step 9 – Launch the application

To run the application in the emulator (Adobe Device Central), select *Control – Test Movie* from the menu. It should look like the image below. Remember that the returned dummy values depend on what test values are set in the XML file.

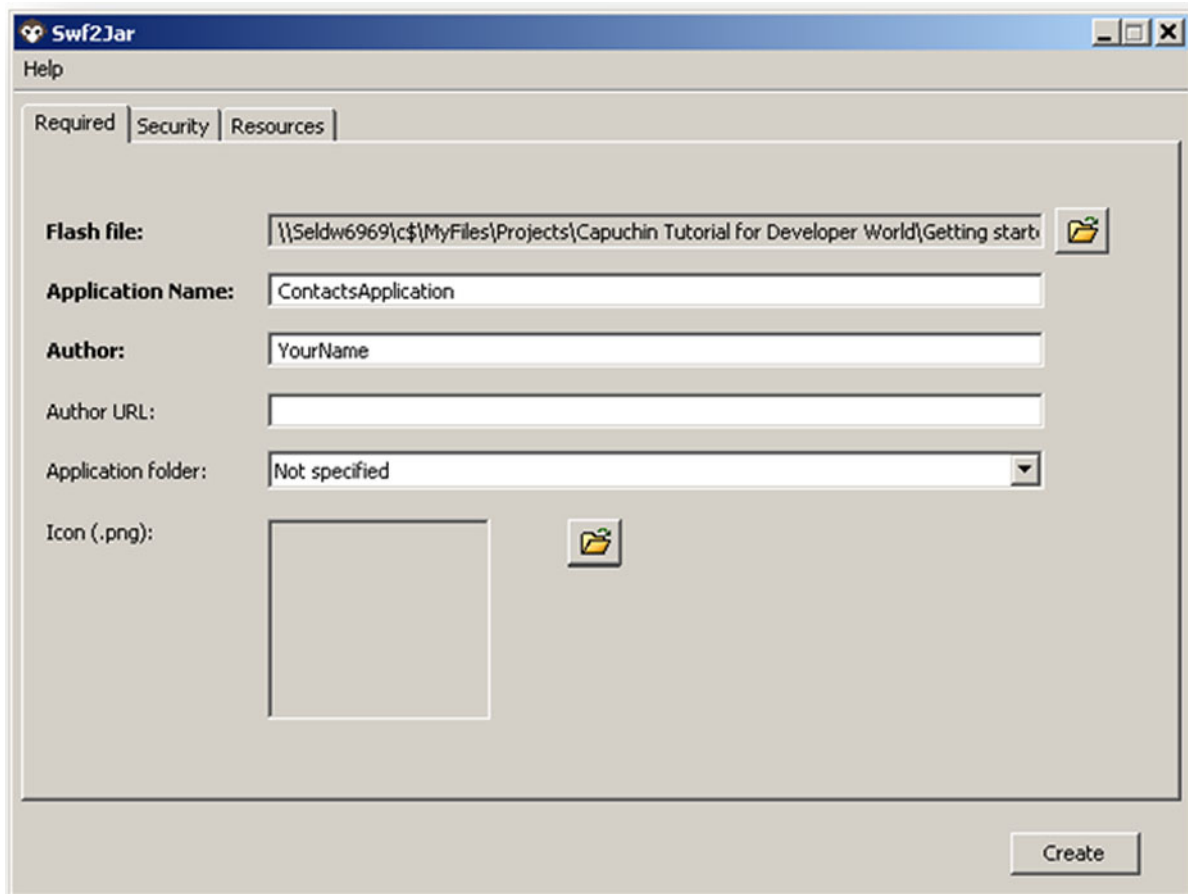


Publishing your project

Publish the application as a jar file. To do that, select *Commands – Create Capuchin Application* from the main menu. A SWF2JAR window opens. (The User guide for the SWF2JAR tool is enclosed in the zip file.

Note: If you have CapuchinKit installed in your Flash IDE then there is no need to download and install the standalone version of SWF2JAR.

As in the image below, there are input text fields under the *Required* tab that need to be filled in (see the SWF2JAR User guide enclosed in the zip file). The *Security* tab is optional and not necessary for the example of this tutorial to work properly. Once the required fields are filled in, click the "Create" button.



Further activities

The complete source code for the Capuchin Contacts example is packed in a .zip file together with this document.

The created jad and jar files install the application on a mobile phone with Project Capuchin support. This can be done using the "Device Explorer" application provided with the Sony Ericsson Java ME SDK. More information on how to install Capuchin applications on a Sony Ericsson phone can be found at http://developer.sonyericsson.com/site/global/techsupport/tipstrickscode/java/p_quick_midlet_install.jsp

Related links

[Sony Ericsson Developer World - Project Capuchin.](#)

[Phone gallery and technical specifications.](#)

[Sony Ericsson Flash UI components](#)