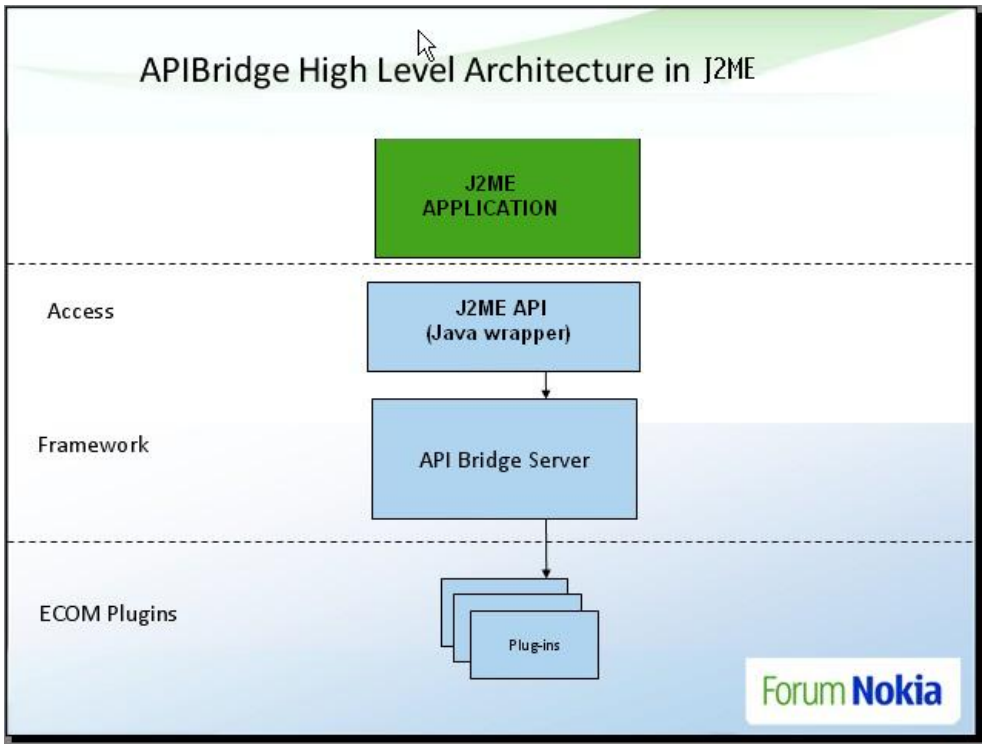
# J2ME API Bridge Interface for accessing native Symbian apps/services

This article explains how to use the services provided by the APIBridge component from J2ME applications on Nokia devices. The document describes the API used to access the services provided by the APIBridge and how to set up your J2ME applications to make use of the APIBridge.

18 Apr 2010

## APIBridge High-Level Architecture

The APIBridge is a Symbian server that exposes an HTTP interface for communication between the server and its clients. The APIBridge can be used by making an http call to the local host port that the APIBridge listens to. The following architecture diagram explains the different parts of the system:




Here is a description of each of the components:

| Component | Description |
|---|---|
| Access Layer J2ME API | The J2ME API provides java applications with a function-based interface to the services of the APIBridge. It performs the following activities:<br>▪ Authentication with the Bridge<br>▪ HTTP requests to the APIBridge from the function calls, abstracting this complexity from the application developer; |
| Framework Layer APIBridge Server | The APIBridge Server is responsible for authenticating clients, receiving requests, routing requests to the appropriate APIBridge plug-in for execution, and returning the results. |
| ECOM Plug-ins Layer Plug-ins | Plug-ins are responsible for analyzing the parameters in the request, calling the appropriate Symbian APIs to execute it, and creating the HTTP response. |

## API Bridge Java Interface

Spite that some of this functionality is already present in the current JavaME implementation. The API bridge opens the gate to new functionality created by the own community since it allows to create native components and expose them via http request.

These are the current plugins available for this release:

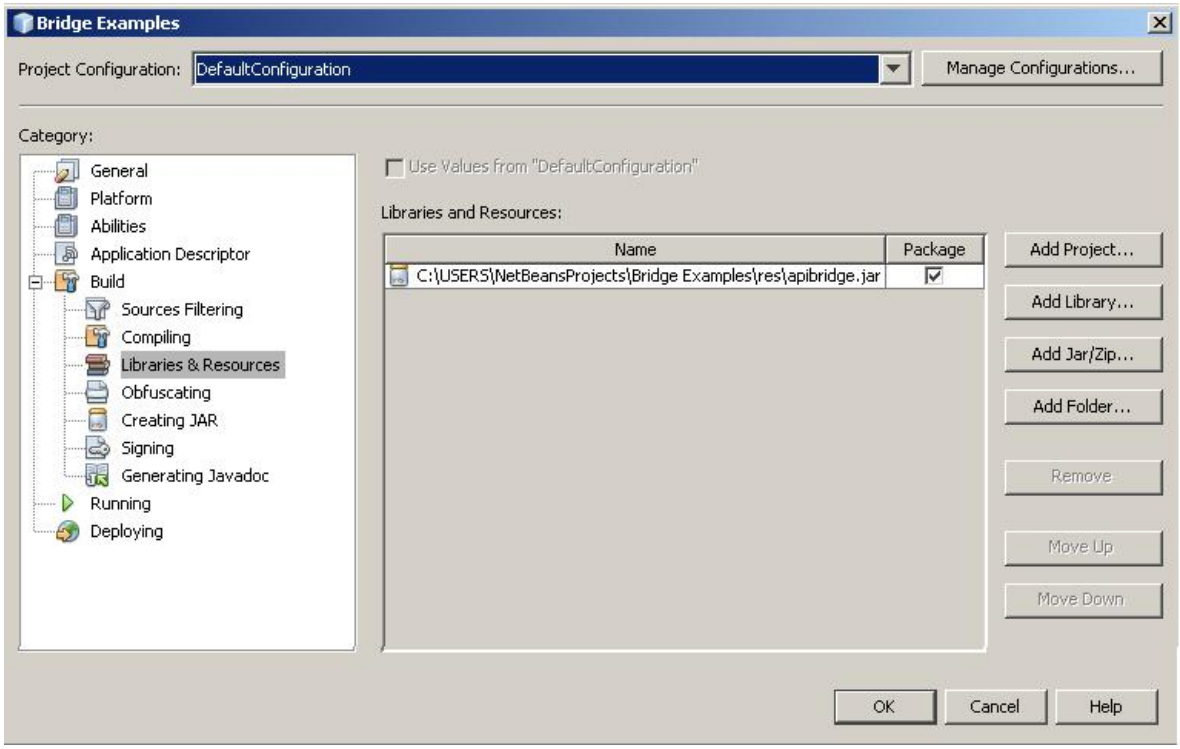
- Logging Service - Allows access to phone log (received calls, outgoing calls, and missed calls)
- Location Service - Access to location data on the phone
- Media Management Service - Retrieves a list of all the media files in the phone
- New File Service - Provides a control to the native camera and its entire functionality.

## Downloading the required JavaME API Bridge library

In the following link you will find the J2ME bundle to the API Bridge. File:J2MEBridge 1 0.zip This library encloses the needed classes and resources that allow your application to communicate with the bridge.

### Adding the J2MEAPIBridge library to the project

- Create a J2ME project using Netbeans (Not covered here)
- Go to project properties: File -> "my project" properties. Or right click over the project name in the project panel and select "Properties"
- In the emerging window, select "Libraries and Resources "
- In the buttons in the right side, select "Add Jar/Zip"
- Select the file "apibridge.jar" you have just downloaded (previous section)
- click OK




## Using the API Bridge

Using the API Bridge, involves 3 steps.

1. Obtain a reference to the APIBridge class and initialize it

```
APIBridge apiBridge = APIBridge.getInstance();
apiBridge.Initialize(myMidlet);
```

   where **apiBridge** is the instance of the APIBridge class and **myMidlet** is the midlet instance that will request the call to the service.

2. Obtain the instance of the service: you need to provide a **service string** and according to that you will receive a service class

```
LogginService loggin =
(LogginService)apiBridge.createService('service.logging');
```

The strings and services returned are described in the following table

| Service String | class Returned |
|---|---|
| "service.logging" | LoggingService |
| "service.location" | LocationService |
| "service.mediamanagement" | MediaManagementService |
| "service.newfileservice" | NewFileService |

3. Using the service

Once you have the reference to the service provider as described above, you can actually use the exposed method to retrieve the information you need. Every service provider has a method returning a **BridgeResult**. This object encloses an object that could be a hashtable or a Vector with the results. More detailed information about each service is shown in the next section.

In the following section you will see how to do it.

## Location Service

The method **GetLocation** allows to get the coordinate pair and the altitude retrieved from the embedded GPS.

```
  public BridgeResult GetLocation() throws BridgeException
```

It returns a **BridgeResult** object enclosing a hashTable with the position values.

The following example shows how to retrieve the GPS information

```
    APIBridge bridge=APIBridge.getInstance();
    bridge.Initialize(this);
    final LocationService service = (LocationService)
bridge.createService("service.Location");
    Thread thread = new Thread() {
        public void run() {
            try {
                BridgeResult res = service.GetLocation();
                Hashtable returnValue = (Hashtable) res.getReturnValue();
                String longitude = returnValue.get("Longitude");
                String latitude = returnValue.get("Latitude");
                String altitude = returnValue.get("Altitude");
            } catch (BridgeException ex) {
                tbox.setString("Bridge error occurred - unable to retrieve data. " +
ex.getMessage());
            } catch (Exception ex) {
                tbox.setString("General error occurred - unable to retrieve data. " +
ex.getMessage());
            }
        }
    };
    thread.start ();
```

## Logging Service

The method `GetList` allows to get a complete and detailed list of the phone events stored in memory (outgoing calls, incoming

calls, missed calls).

```
public BridgeResult GetList() throws BridgeException
```

It returns a **BridgeResult** object with a Vector of Hashtable elements. Each Hashtable element represents an event in the phone log (outgoing calls, incoming calls, and missed calls) Each Hashtable can have the following keys

```
"Description"
"EventType"
"id"
"Subject"
"PhoneNumber"
"EventTime"
"DeliveryStatus"
"Direction"
"RemoteParty"
```

The following example shows how to retrieve the last 10 events in the phone log and display each field of it.

```
        APIBridge bridge=APIBridge.getInstance();
        bridge.Initialize(this);
        final LoggingService service = (LoggingService)
bridge.createService("service.logging");
        Thread thread = new Thread()
        {
            public void run()
            {
                try
                {
                    BridgeResult res = service.GetList();
                    Vector returnValues = (Vector)res.getReturnValue();
                    StringBuffer out = new StringBuffer("Result: \n");

                    for(int i = 0; i < Math.min(10, returnValues.size()); i++)
//Iterate through the latest 10 events
                    {
                        Hashtable item = (Hashtable)(returnValues.elementAt(i));
//Every element in the vector contains a Hashtable

                        Enumeration enumer = item.keys();                          //Get
the list of the keys in the hashtable
                        while(enumer.hasMoreElements()){

                        String key = (String) enumer.nextElement();
                        out.append(  key +" "+ item.get(key)+ "\n");
//print the key and values of each element
                        }
                    }
                    System.out.println(out.toString());
                }
                catch(BridgeException ex)
                {
                   tbox.setString("Bridge error occurred - unable to retrieve data. " +
```

```
ex.getMessage());
                }
                catch(Exception ex)
                {
                    tbox.setString("General error occurred - unable to retrieve data. "
+ ex.getMessage());
                }
            }
        };

        thread.start();
```

This is the information "keys" you will find in the logging service

## Media Management Service

Allows getting a filtered and sorted list of the media in the phone.

```
public BridgeResult GetList(Hashtable filter, Hashtable sort) throws BridgeException
```

**Parameters**

| Parameter | Description |
|---|---|
| Hashtable filter | contains the elements used for filtering the result set. The different possible keys are:<br>■ "FileType" : type of files to retrieve, it's possible values are "Music" "Sound" "Image" "Video" "StreamingURL"<br>■ "Key"<br>■ "StartRange"<br>■ "EndRange" |
| Hashtable sort | contains the elements used for sorting the result set. The different possible keys are:<br>"key" "order" |

the returned **BridgeResult** contains a **Vector** of **Hashtable** elements where each element contains the information of a single file. Every Hashtable has the following keys:

```
"FileNameAndPath"
"FileDate"
"Drive"
"FileName"
"FileSize"
"FileExtension"
"MimeType"
```

The following example shows how to get the information of the first 10 image files of the phone

```
  APIBridge bridge = APIBridge.getInstance();
            bridge.Initialize(this);
            final MediaManagementService service = (MediaManagementService)
bridge.createService("service.mediamanagement");
            Thread thread = new Thread() {

                public void run() {
                    try {
```

```
                    Hashtable filter = new Hashtable();
                    filter.put("FileType", "Image");

                    Hashtable sort = null;

                    BridgeResult res = service.GetList(filter, sort);
                    Vector returnValues = (Vector) res.getReturnValue();
                    StringBuffer out = new StringBuffer("Result: \n");
                    for (int i = 0; i < Math.min(10, returnValues.size()); i++) {

                        Hashtable item = (Hashtable) (returnValues.elementAt(i));

                        Enumeration enumer = item.keys();
                        while (enumer.hasMoreElements()) {

                            String key = (String) enumer.nextElement();
                            out.append(key + " " + item.get(key) + "\n");
                        }
                    }
                    System.out.println(out.toString() + "\n");
                } catch (BridgeException ex) {
                    System.out.println("Bridge error occurred - unable to retrieve
data. " + ex.getMessage());
                } catch (Exception ex) {
                    System.out.println("General error occurred - unable to retrieve
data. " + ex.getMessage());
                }
            }
        };

        thread.start();
```

## New Multimedia File Service

Allows creating a new multimedia file using the phone multimedia capturing resources. The API will pass the control to the native camera interface, once picture or video is taken, the API returns the path of the picture or video stored on the phone. The following method is responsible for starting execution of the native multimedia interface:

```
public BridgeResult TakePhoto(Hashtable filter) throws BridgeException
```

**Parameters**

| Parameter | Description |
|---|---|
| Hashtable filter | contains the commands to control the type of media being created. The different possible keys are:<br>▪ "NewFileType" : type of files to create, its possible values are "Image" "Video" "Audio" |

The **BridgeResult** returned encloses a hashtable where the path of the new image is stored. The key to retrieve this value is "src". The following example shows how to use this function:

```
        APIBridge bridge = APIBridge.getInstance();
        bridge.Initialize(this);
        final NewFileService service = (NewFileService)
```

```
bridge.createService("service.newfileservice");
        Thread thread = new Thread() {

            public void run() {
                try {
                    Hashtable filter = new Hashtable();
                    filter.put("NewFileType", "Image");

                    BridgeResult res = service.TakePhoto(filter);
                    Hashtable returnValue = (Hashtable) res.getReturnValue();
                    System.out.println("Source: " + returnValue.get("src").toString());
                } catch (BridgeException ex) {
                    System.out.println("Bridge error occurred - unable to retrieve data.
" + ex.getMessage());
                } catch (Exception ex) {
                    System.out.println("General error occurred - unable to retrieve
data. " + ex.getMessage());
                }
            }
        };
        thread.start();
```

## Packaging the App

1. Download the latest version of the Api Bridge: [1]
2. To bundle everything together follow the instructions in this link: Installing Java apps and WRT widgets using sis files

## Downloading the examples

You will find complete examples in the following link File:J2MEBridgeExamples.zip

Please make sure to read the "Readmefirst.txt" file to correctly setup the jar library in your project.

## See also

- MIDletNativeServicesFramework