

UTILIZING EXTERNAL APPLICATION VIEWS

Version 1.0

20 Nov 02

Table of Contents

1.	SCOPE	3
2.	INTRODUCTION TO SERIES 60 VIEW STRUCTURE	3
2.1	VIEW ARCHITECTURE OVERVIEW	3
2.2	SWITCHING VIEWS	3
2.3	SENDING MESSAGES.....	3
2.4	DOACTIVATEL AND THE PREVIOUS VIEW ID	4
2.5	DODEACTIVATEL	4
2.6	USING OTHER APPLICATIONS' VIEWS	4
3.	MESSAGE UIDS AND MESSAGES	5
3.1	MAIN APPLICATIONS.....	5
3.1.1	Phonebook Application.....	5
3.1.1.1	Two main views.....	5
3.1.1.2	Focused views.....	5
3.1.2	Log Application.....	6
3.1.3	Messaging application.....	6
3.1.3.1	Message Uids for showing different folders.....	7
3.1.3.2	How to open a new message window.....	7
3.1.4	Calendar Application.....	8
3.1.5	Services Application (WAP).....	8
3.1.5.1	How to start browser with a specific Web page.....	8
3.1.6	Profile Application	9
3.1.7	To-Do Application	9
3.2	IMAGING	10
3.2.1	Camera Application	10
3.2.2	Photo Album Application	10
3.3	CONNECTIVITY	10
3.3.1	Bluetooth Application.....	10
3.4	EXTRAS	11
3.4.1	Recorder Application.....	11
4.	EXAMPLE APPLICATION.....	11

Change History

20 Nov 02	Version 1.0	Document published in Forum Nokia.
-----------	-------------	------------------------------------

Disclaimer

Nokia Corporation disclaims all liability including liability for infringement of any proprietary rights relating to the implementation of information presented in this document. Nokia Corporation does not warrant or represent that such use will not infringe such rights. Nokia Corporation retains the right to make changes to this specification at any time without notice.

License

A license is hereby granted to download and print a copy of this specification for personal use only. No other license to any other intellectual property rights is granted herein.

1. SCOPE

The following document describes how to call forward external application views. Initially we will discuss how to create a view application, as your application's view may be activated by another application. This will also serve as a reference point when you are considering whether to create a view or use a view of an existing application.

After the explanation of view architecture, several useful Uids are presented in Chapter 3.

2. INTRODUCTION TO SERIES 60 VIEW STRUCTURE

View architecture is widely used in application development in Series 60 Platform. A GUI application can create several views and display different application data or UI controls in each view. Each view has its own control stack. Each view's container and controls are created when it is called forward, and destroyed when another view of the same application is called forward. When activating a view, a message ID and a message can be passed. This provides a great deal of functionality that you can build and use in existing applications.

2.1 View Architecture Overview

In the `AppUi` class of a traditional Eikon GUI application, you create the container class, which creates all the controls. In a view architecture GUI application, you create a view class, derived from `CAknView` in the `AppUi`, which is derived from `CAknViewAppUi`. `CAknViewAppUi` adds view-handling methods to `CAknAppUi`. The activation and deactivation of a view are handled in the view class. This is done in the functions `DoActivateL()` and `DoDeActivateL()`. In these functions the container class is created and destroyed. The container class in turn creates and destroys the controls in the container.

2.2 Switching Views

To switch to a view within your application, you can use the `AppUi` class method `ActivateLocalViewL()`. The parameter for `ActivateLocalViewL()` is a `TUid`. The following is an example of view switching:

```
const TUid KDemolViewId = { 1 }; // UID of the first view
ActivateLocalViewL(KDemolViewId); // activate view 1
```

A new view is activated first and the previous view is deactivated after that. This allows quick view switching to take place. When deactivating, all controls, including menus and dialogs are also closed down. System dialogs automatically handle this. If you need to save data in your dialog when the dialog is open but the view is being shut down, you need to save the data when handling `EaknSoftkeyCancel`. In these cases `Cancel/back` route in the dialog is followed.

2.3 Sending Messages

`ActivateLocalViewL()` has been overloaded to include `MessageUid` and a message. `MessageUid` is a `TUid` and it is usually used for a specific dialog page in a view or to execute a certain functionality. Messages are in `TDesC8` descriptors and can be used to pass data between the views. Here is an example:

```
const TUid KViewUid= {1};  
const TUid KCustomMessageUid= {2};  
TBuf8<255> customMessage;  
customMessage.Copy(_L8("Some data here"));  
ActivateLocalViewL(KViewUid, KCustomMessageUid, customMessage);
```

2.4 DoActivateL and the Previous View ID

The purpose of `DoActivateL()` in the view class is to create the container and also to handle the messages that are passed. In addition to those described in the previous chapter, `DoActivateL()` gets the `TVwsViewId` for the previous view. This can be used to switch back to the original view with or without a message, which may be useful within your own application views. If your application is called by another application, and when the user exits your application, the previous view is automatically restored. Knowing the previous view can be useful if you want to specify which application to respond to.

2.5 DoDeActivateL

`DoDeActivateL` is called when another view has been activated and the previous active window needs to be shut down. This order makes view switching fast. In `DoDeActivateL` the view is removed from the stack and therefore the view's container and its controls are destroyed.

2.6 Using Other Applications' Views

To activate a view of an external application, use `ActivateViewL()` of the `AppUi` class. `ActivateViewL()` behaves in the same way as `ActivateLocalViewL()` does. The only difference is that `ActivateViewL()` takes `TVwsViewId` parameter type instead of `TUid`. `TVwsViewId` consists of the `Uid` of the application and the `Uid` of the view in that application. Here is an example:

```
const TUid KPhotoAlbumUid = { 0x101F4CD1 };  
CCoeAppUi::ActivateViewL(TVwsViewId(KPhotoAlbumUid, TUid::Uid(1)));
```

`ActivateViewL()` has been overloaded in the same way as `ActivateLocalViewL()` to pass messages. Here is an example:

```
const TUid KCustomMessageUid= {2};  
TBuf8<255> customMessage;  
customMessage.Copy(_L8("Some data here"));  
const TUid KPhotoAlbumUid = { 0x101F4CD1 };  
CCoeAppUi::ActivateViewL(TVwsViewId(KPhotoAlbumUid, TUid::Uid(1)),  
KCustomMessageUid, customMessage);
```

When the user exits the launched application, the framework will automatically return the original view, from where the `ActivateViewL()` was called. A number of useful Uids, View Uids, MessageUids, and messages are described in Chapter 3.

3. MESSAGE UIDS AND MESSAGES

In this section, we will go through the application Uids, View Uids, MessageUids, and messages that can be useful in your application development. Only some of the most useful system application view Uids are dealt with in this document. Common alternatives to `ActivateViewL` are also presented. These solutions have been tested on Nokia 7650, and may or may not apply when used in other Series 60 devices.

3.1 Main Applications

3.1.1 Phonebook Application

3.1.1.1 Two main views

Application Uid: 0x101f4cce

View Uids:

0x01 // contacts view

0x02 // groups view

3.1.1.2 Focused views

Phonebook can be focused to show a focused view of the contacts.

Example to activate Phonebook's contact info view to show contact with ID 5:

```
#include < CPbkViewState.h>
CPbkViewState* pbkViewParam = CPbkViewState::NewLC();
// Focus on contact with id 5. You have to have contact 5 for
this to work.
pbkViewParam->SetFocusedContactId(5);
// Package and return object in a buffer.
HBufC8* paramBuf = pbkViewParam->PackLC();
```

NOKIA

Utilizing External Application Views

Version 1.0

```
// MakeUid for Phonebook's app UID
const TUid appUid = { 0x101f4cce };

// Activate the view
// Uid (4) is the Contact Info View's id
CCoeAppUi::ActivateViewL(TVwsViewId(appUid, TUid::Uid(4)),
CPbkViewState::Uid(), *paramBuf);

// Cleanup
CleanupStack::PopAndDestroy(2); // paramBuf, pbkViewParam
```

For more information, see SDK help: "CPbkViewState Class Reference" and "CPbkViewState.h".

You may need to comment out the following line from CPbkViewState.h for compilation:

```
// #include <PbkUID.h> // Phonebook UIDs
```

Remember to add PbkView.lib to your MMP file.

3.1.2 Log Application

Application Uid: 0x101f4cd5

The following view Uids can be used:

- 1 // log's event list view in logs state (Call register/Call Timers/Call Costs/GPRS counter)
- 6 // event list view in missed calls state
- 8 // event list view in dialed calls state

3.1.3 Messaging application

Application Uid: 0x100058C5

View Uids:

- 0x01 // for main view in messaging application
- 0x02 // show different folders
- 0x03 // delivery report

3.1.3.1 Message Uids for showing different folders

We recommend that you use these values for different message folders in messaging application view 0x02. These are defined in MSVSTD.HRH.

```
KMsvGlobalInBoxIndexEntryIdValue    // Inbox
KMsvGlobalOutBoxIndexEntryIdValue   // Outbox
KMsvDraftEntryIdValue               // Drafts
KMsvSentEntryIdValue                // Sent
```

Messages are not used in this interaction.

3.1.3.2 How to open a new message window

It is not possible to open a window to write a message using the `ActivateViewL`. However, it is possible to accomplish this using the `CSendAppUi` class. This is a frequently asked question, which is why it is presented here.

```
CParaFormatLayer* paraFormatLayer = CParaFormatLayer::NewL();
CleanupStack::PushL(paraFormatLayer);

CCharFormatLayer* charFormatLayer = CCharFormatLayer::NewL();
CleanupStack::PushL(charFormatLayer);

CRichText* messageBodyContent = CRichText::NewL(paraFormatLayer,
charFormatLayer);
CleanupStack::PushL(messageBodyContent);
messageBodyContent->InsertL(0, KSAEXTag);

// See CSendAppUi Class Reference in SDK help for other
parameters.

iSendUi->CreateAndSendMessageL(KUIdMsgTypeSMS,
messageBodyContent);

CleanupStack::PopAndDestroy(3); // messageBodyContent,
charFormatLayer, paraFormatLayer
```

3.1.4 Calendar Application

Application Uid: 0x10005901

View Uids:

0x01 // Month View

0x02 // Week View

0x03 // Day View

These views do not take messages.

If the calendar application was not previously running, the view for the current day view is automatically opened. If you set month or week view, this screen will glimpse before the current day view.

If the calendar application is already running and the cursor is set on another day, this day will automatically show in day view. If the application is already running, it will also show the month and week views, and not only glimpse them.

3.1.5 Services Application (WAP)

Application Uid: 0x10008D39

3.1.5.1 How to start browser with a specific Web page

Because it is not possible to open a specific WAP page using `ActivateViewL()`, the following example shows how to start a browser with a specific WAP page:

```
HBufC* param = HBufC::NewLC( 20 );  
//param->Des().Copy( _L( "4 http://wap.yahoo.com/" ) );  
param->Des().Copy( _L( "4 wap.yahoo.com/" ) );  
// Wap Browser's constants UId  
  
const TInt KWmlBrowserUid = 0x10008D39;  
TUid id( TUid::Uid( KWmlBrowserUid ) );  
TApTaskList taskList( CEikonEnv::Static()->WsSession() );  
TApTask task = taskList.FindApp( id );  
if ( task.Exists() )
```

NOKIA

Utilizing External Application Views

Version 1.0

```
{
    HBufC8* param8 = HBufC8::NewLC( param->Length() );
    param8->Des().Append( *param );
    task.SendMessage( TUid::Uid( 0 ), *param8 ); // Uid is not
used
    CleanupStack::PopAndDestroy();
}
else
{
    RApaLsSession  appArcSession;
    User::LeaveIfError(appArcSession.Connect()); // connect
to AppArc server
    TThreadId id;
    appArcSession.StartDocument( *param, TUid::Uid(
KWmlBrowserUid ), id );
    appArcSession.Close();
}
CleanupStack::PopAndDestroy(); // param
```

3.1.6 Profile Application

Application Uid: 0x100058F8

View Uids:

0x01 // main view

0x02 // settings view

No messages or message Uids are used here.

3.1.7 To-Do Application

Application Uid: 0x10005900

This application does not take View Uids, Message Uids, or messages.

NOKIA

Utilizing External Application Views

Version 1.0

3.2 Imaging

3.2.1 Camera Application

Application Uid: 0x1000593F

View Uids:

0x01 // standby mode

0x02 // viewfinder mode

0x04 // Name base, and Quality settings

None of the views takes Message Uids or messages.

3.2.2 Photo Album Application

Application Uid: 0x101F4CD1

View Uids:

0x01 // Imagelist view – the default view

0x03 // Messaging Picture Grid view

0x04 // Messaging Picture view

3.3 Connectivity

3.3.1 Bluetooth Application

Application Uid: 0x10005951

View Uids:

0x01 // Main view

0x02 // Paired devices view

These views do not take messages.

3.4 Extras

3.4.1 Recorder Application

Application Uid: 0x100058CA

View Uids:

0x01 // filelist

0x02 // recorder

MessageUids for recorder view 0x02:

0x00 // play and record available on demand

0x01 // recording immediately

0x02 // recording on one button press

0x03 // player option only

If your message length is larger than 0, then the application converts the 8-bit descriptor to a 16-bit one without altering data.

4. EXAMPLE APPLICATION

An example application, "MyView," has been created to make the information presented in this document more practical. The application demonstrates local view switching and message passing. It also activates several different kinds of external application views from the menu items. You can download this application from Forum Nokia's web site www.forum.nokia.com -> Symbian -> Downloads -> **Series 60 Downloads**.