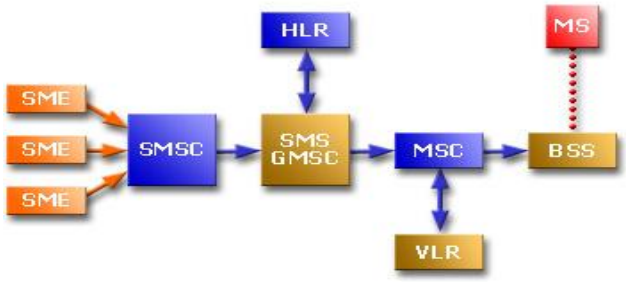
Localized SMS Application

PAN Localization

SMS

- Short Messaging Service (SMS) provides a mechanism to send and receive text messages using mobile phones.



PAN Localization

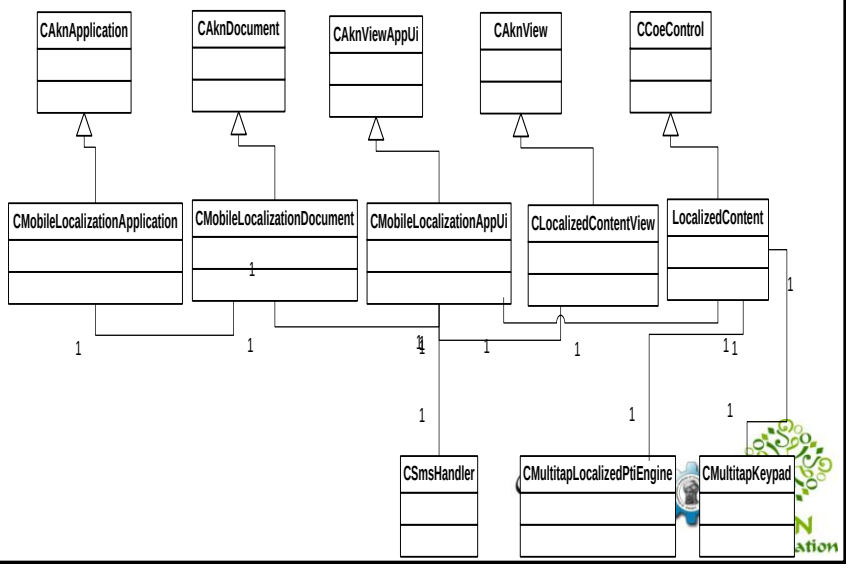
SMSLocalized Application

- The SMSLocalized application is a Symbian application designed for the languages supported through Pango.
- Implemented for Urdu
- Can be customized for other languages

Application Features

- Allows typing of text message in complex scripts using open type fonts such as Nafees Nastaleeq in Urdu and Arabic.
- Provides a customized on-screen keyboard layout for Urdu language.
- Send and Receive messages in Urdu language.
- Automatic application startup on receiving an Urdu SMS

Application Design



Definition of Custom Keyboard

- Custom key map defined so that appropriate characters of Urdu are rendered against each key press
- Mobile phones support multi-tapped input
- Definition of keyboard requires
  - Define and Load Keymap
  - Install and Load Fonts
  - Layout and Load the Keyboard

Define Customized Keymap

- PtiEngine APIs provide low level text input functionality
- A keymap is defined in a text file.
- The keymap defined in the step above, is loaded and passed to PtiEngine API

```
#define STR_EPtiKeyMap2
    <0x0628><0x067E><0x062A><0x0679><0x062B><0x06C3><0x06F2>
```

Localization of Mobile Phones (Pg 69-70)

Install & Load Fonts

- *TFontSpec, CFont*
- The font file should be in appropriate folders

- *Localization of Mobile Phones* (Pg 61-62)

Numeric Keyboard

- Definition of keyboard layout in a text file.
- Measurement of space available on the screen for laying out the keypad
- Layout the keypad in the local language

Localization of Mobile Phones (Pg 62-63)

Send Receive Message

- Communication using Sockets
- Messaging Using MTM API

Communication using Sockets

- Similar to the Berkeley sockets mechanism
- The host opens a listening socket and binds it to a local port.
- The host listens for incoming connections on the listening socket. It calls RSocket::Listen(), passing in a queue size, which specifies the number of connections that it will accept on that port.
- The host opens a blank socket which is not yet connected to a remote socket an passes it to the listening socket via RSocket::Accept(). The host waits for a client t connect.
- The client opens a connecting socket with the host phone's address, protocol and port number. The format of these values depends on the type of connection.
- The client calls RSocket::Connect() and waits for the host to accept the connection.
- When the host detects an incoming connection, its accept operation completes and it establishes a connection between the connecting client and the blank socke
- On the client side, the connect operation completes. The connect socket can now exchange data with the accept socket on the host.


Messaging Using MTM API

- **Protocol Plug-Insfor the Message Server**
- Handle lower level communication (TCP/IP, …)
- Offer two APIs: **Server and Client-API**
- Pre-installed for: SMS (EMS), MMS, POP3, IMAP4, SMTP


MTM Supported functionality

- Create
- Store and retrieve an SMS message entry in the Message Store
- Send an SMS
- Reply to a received SMS
- Forward existing SMS


MTM Key Concepts

- SMS Client MTM (CSMSMtmClient)
  - Provides operation
    - Accessing a telephone handset's service centre information
    - Scheduled sending of SMS messages
- SMS Service Settings(CSmsSettings)
  - Settings for SMS connections e.g. Service centre addresses, are stored in the message store (CMsvStore) for the SMS service entry.
- Progress Information(TSmsProgress)
  - e.g. type of operation, and number of messages processed


Send SMS MTM(1)

1. Establish session to the MessageServer
2. Create Client MTM Registry
3. Use this for creating handle to required MTM
4. Create a local index entry with status "in preparation" (TMsvEntry)
5. Construct entry in MessageServerthrough MTM (CMsvEntry; assigns ID, Store, Folder,…)
6. Add data(text, recipient)
7. Savefinal message


Send SMS MTM(2)

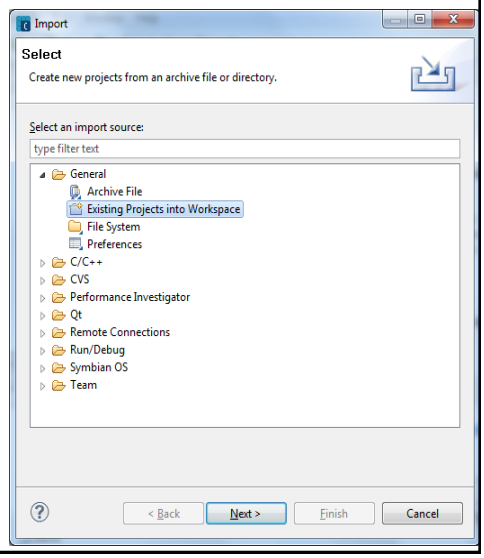
8. Validate (optional) –Check if message conforms to message type
9. Set SMS Service Center number through MTM settings
10. Switchstatus:"in preparation" "waiting to be sent"
11. Submit asynch. request to send the message
12. Check return value (RunL(), iStatus)


Interfacing with Pango

- Capture the Unicode characters strings entered by the user. These strings are then passed to pango for rendering in desired script.
- A class CPangoInterface has been added in the application to provide interfacing with Pango.
- CPangoInterface set values of
  - cairo format, display mode, font family name, font size, font weight and language.
- To draw text a function on top of Pango has been defined which takes as input the unicodeString and returns the rendered text in bitmap format.

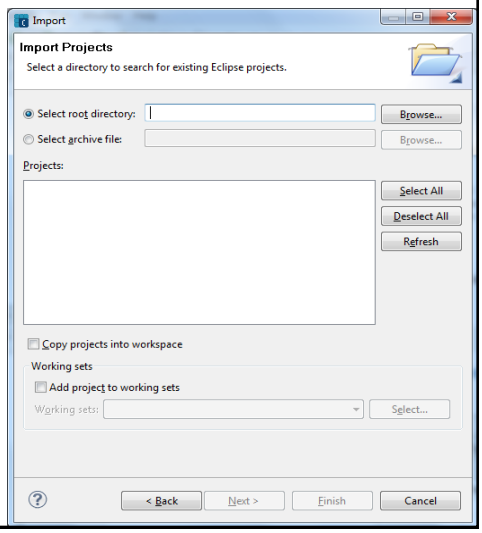
Import Existing Pango & SMS Localized Application

- Copy Pango & SMS Localized Application to your Workspace
- Go to top menu File->Import
- Select Existing Projects Option
- Click Next



Import Existing Pango & SMS Localized Application

- Browse Your Pango Project located in your workspace
- Click Finish
- Imported Pango project will be visible in solution explorer
- Repeat these steps for SMS Localized application



Steps to Create SMS Localized Application

- Create new workspace
- Add and Build PangoCairo Project
  - Import PangoCairo project by SolutionExplorer->Import
  - Select Project Type General ->Existing Projects into Workspace

Steps to Create SMS Localized Application

- Create New S60 project of type "GUI Application with UI Designer". Project creation wizard and project properties are already discussed in previous chapter.
- Open mmp file of SMSLocalized project and open Libraries tab
  - Go to Libraries listbox and click on Add button
  - Add library dialogue appears
  - Select and add cairo.lib, pangocairo.lib, glib-missing.lib, fontconfig.lib, freetype.lib, libexpat.lib, libglib.lib, libpng.lib one by one

Steps to Create SMS Localized Application

- Open mmp file of SMSLocalized project and open Options tab
  - Modify compiler settings click on System includes.
  - Click on add button , Edit include path windows will appear
  - Browse Cairo folder from your S60/epoch32/include.
  - Similraly add expat, fontconfig, freetype, freetype/config, pango, pixman,png, stdapis, stdapis/glib-2.0

Steps to Create SMS Localized Application

- Open .pkg file and add up the following lines to include pangocairo on device.

- "$(EPOCROOT)Epoc32\release\gcce\udeb\glib-missing.dll" - "!:\sys\bin\glib-missing.dll"
- "$(EPOCROOT)Epoc32\release\gcce\udeb\libexpat.dll" - "!:\sys\bin\libexpat.dll"
- "$(EPOCROOT)Epoc32\release\gcce\udeb\freetype.dll" - "!:\sys\bin\freetype.dll"
- "$(EPOCROOT)Epoc32\release\gcce\udeb\fontconfig.dll" - "!:\sys\bin\fontconfig.dll"
- "$(EPOCROOT)Epoc32\release\gcce\udeb\libpng.dll" - "!:\sys\bin\libpng.dll"
- "$(EPOCROOT)Epoc32\release\gcce\udeb\pixman-1.dll" - "!:\sys\bin\pixman-1.dll"
- "$(EPOCROOT)Epoc32\release\gcce\udeb\cairo.dll" - "!:\sys\bin\cairo.dll"
- "$(EPOCROOT)Epoc32\release\gcce\udeb\pangocairo.dll" - "!:\sys\bin\pangocairo.dll"
-
-
- "$(EPOCROOT)Epoc32\data\c\data\romedalen.png" - "!:\data\romedalen.png"
- "$(EPOCROOT)Epoc32\data\c\data\fontconfig\fonts.dtd" - "!:\data\fontconfig\fonts.dtd"
- "$(EPOCROOT)Epoc32\data\c\data\fontconfig\fonts.conf" - "!:\data\fontconfig\fonts.conf"
- "$(EPOCROOT)Epoc32\release\winscw\udeb\z\resource\fonts\Nafees Nastaleeq.ttf" - "!:\resource\fonts\Nafees Nastaleeq.ttf"

Steps to Create SMS Localized Application

- After successful integration of PangoCairo in SMSLocalized project our next step is to design keyboard and display keyboard.
- Open SMSLocalized file SMSLocalizedContainer.uidesign add text area for message and phone number.
- Add menuitem for send message and inbox in SMSLocalizedContainer.uidesign
- Add Class for interfacing with PangoCairo. This class will be responsible for setting up value for font family, and font size. This class will also include the functionality to take Unicode string and return image.

SMSLocalized Application Flow