



Nokia Data Gathering

Mobile Client
Developer Guide
(2.11)

Contents

Contents.....	2
Introduction.....	4
1. Packages description.....	5
1.1 br.org.indt.ndg.lwuit.control.....	5
1.2 br.org.indt.ndg.lwuit.extended.....	5
1.3 br.org.indt.ndg.lwuit.model.....	5
1.4 br.org.indt.ndg.lwuit.ui.....	5
1.5 br.org.indt.ndg.lwuit.ui.camera.....	5
1.6 br.org.indt.ndg.lwuit.mobile.....	6
1.7 br.org.indt.ndg.lwuit.mobile.download.....	6
1.8 br.org.indt.ndg.lwuit.mobile.error.....	6
1.9 br.org.indt.ndg.lwuit.mobile.logging.....	6
1.10 br.org.indt.ndg.lwuit.mobile.multimedia.....	6
1.11 br.org.indt.ndg.lwuit.mobile.settings.....	6
1.12 br.org.indt.ndg.lwuit.mobile.structures.....	6
1.13 br.org.indt.ndg.lwuit.mobile.submit.....	7
1.14 br.org.indt.ndg.lwuit.mobile.xmlhandler.....	7
1.15 com.jcraft.jzlib.....	7
1.16 com.nokia.mid.appl.cmd.....	7
1.17 com.twmacinta.util.....	7
1.18 resources.images and resources.text.....	7

2.	Starting sequence	8
3.	Example of UI flow. Views and classes	8
4.	UI map. Events and screens.....	13
5.	Building survey UI and data flow.....	14
6.	Send and receive data	15
7.	Parsing XML files	15
8.	Localization (GPS) module.....	15
9.	Camera.....	15

Introduction

Nokia Data Gathering is a solution that helps organizations *to collect field data using mobile phones* instead of paper forms, PDAs or laptops. Since mobile phones can send data from many remote locations, collected data can be transmitted in near real-time for analysis. This makes collection of data much *faster, more accurate and more cost effective* to gather especially in remote locations when dealing with critical issues, such as public health, agricultural stock levels, emergency services and alike.

The solution consists of *two modules, server and mobile phone*, to enable smooth information transfer from the survey administrators to the field workforce and vice versa. The process includes the creation of questionnaires, their delivery to mobile phones and the subsequent integration and analysis of results.

This mobile client developer guide gives information on the packages, UI flows, UI maps and some of the features in the client such as GPS module and camera. More information about Nokia Data Gathering and further resources can be found from <https://projects.forum.nokia.com/ndg/wiki>.



Figure 1 Nokia Data Gathering mobile client

1. Packages description

1.1 br.org.indt.ndg.lwuit.control

Object in this package encapsulates actions present in mobile client, are named with suffix “Command” and derived from Event class. Those objects link user actions, like menu selection or pressing a button, with some operation on data. More information in chapter 3

Additional classes:

ResultControl – to get results and sent results names

PersistenceManager – saves results as xml

SurveysControl – control class for operations on surveys

1.2 br.org.indt.ndg.lwuit.extended

Each question must be presented for the user in some form. For descriptive question there must be space to type quite long answer. From the other hand exclusive choice question does not need such free space. More appropriate is a short list with available options. What’s more we should ensure that one and only one option is selected. If we wish to select more than one option we should use multiple choice question, which has different functionality. All of those questions must have some graphical representation, and what is more important, we should ensure that logical integrity of the answer. All described functionality is ensured by this package objects.

1.3 br.org.indt.ndg.lwuit.model

We already have components responsible for graphical representation of question. Now we need something to store all questions and answers. This package contains mechanism to store and validate answers. As user might make some mistake, we need to check if current answer fulfill constrains.

For example, for numeric question we would like to have answer in some gaps like [1-10]. If user decides to answer with number greater than 10, we should inform that such an answer is out of expected gaps.

1.4 br.org.indt.ndg.lwuit.ui

This package contains all form used to display data for user. Here we have screens responsible for displaying list of survey, list of results, preview of result, and many, many other. This component is also responsible for interaction with user. So we also have menus, and handler responsible for key events. Each operation on UI is translates to one of action form “br.org.indt.ndg.lwuit.control” package.

1.5 br.org.indt.ndg.lwuit.ui.camera

This is a sub package related with photo questions. It handles preview and taking photo operation.

1.6 br.org.indt.ndg.lwuit.mobile

This is a main package of mobile client. This package is responsible for preparing all necessary resources. It also handle save and load operation for survey and result.

1.7 br.org.indt.ndg.lwuit.mobile.download

If user decides to check if there are new survey prepared to be downloaded, this package contains whole logic which is needed to do such operation. First there is a query to check if there are new surveys available to download. If yes, user can download it immediately or do it later. List of all survey available is presented. User is also informed if some of these surveys were already download to his mobile.

1.8 br.org.indt.ndg.lwuit.mobile.error

Package, mainly responsible for translating numeric error codes to some human readable communicates, for example a problem with remote server, problem with network coverage etc.

1.9 br.org.indt.ndg.lwuit.mobile.logging

To improve debugging and bug tracking developer can write some extra information to file. This component is only needed during development. This shouldn't be part of the final release.

1.10 br.org.indt.ndg.lwuit.mobile.multimedia

Here we have utilities responsible for encoding photos and preparing thumbnails.

1.11 br.org.indt.ndg.lwuit.mobile.settings

Some settings in mobile client are configurable. User can enable or disable some features, like GPS module. Such settings are stored in settings.xml. This package is responsible for preparing such file during first application start. Later on is responsible for reading and updating it according user to settings. What is more, component detect memory card and if card is available all data is store on phone's MMC instead of phone internal memory. Location and language handlers are ale located in this package.

1.12 br.org.indt.ndg.lwuit.mobile.structures

Here we have some intermediate data model. Those structures are used while parsing xml file with surveys and results. Later on those structures are encapsulated in data model which is more flexible in use and exposes some additional functionality.

1.13 br.org.indt.ndg.lwuit.mobile.submit

As br.org.indt.ndg.lwuit.mobile.download is responsible for downloading survey, this component send starts connection with remote server and sends result back. Also some diagnostic functionality is available to check client configuration, for example to check if network is configured.

1.14 br.org.indt.ndg.lwuit.mobile.xmlhandler

Package delivers easy to use tools to read survey and result for xml file stored in local file system.

1.15 com.jcraft.jzlib

Stores algorithm use to compress data before sending to remote server.

1.16 com.nokia.mid.appl.cmd

Tries to load message in phone default language. If translation is not available it loads English version. This is a part of Nokia localization tool for J2ME.

1.17 com.twmacinta.util

Store algorithm used to check survey integrity. This allows to detect that survey was tempered. And is no guarantee that something wasn't change in questions.

1.18 resources.images and resources.text

Stores images and messages used in application. It allows to prepare another translation for application as everything is stored in one place.

NOTE: To add additional string to translation you must open proper xls file from \localization directory. Once you add a string to xls it must be opened with i18n.jar (part of Nokia localization tool for J2ME) and saved in \src\main\java\resources\text format of default language data base file with extension pointing to proper language. Now you have to change locale.java and resources.java.

2. Starting sequence

Important events during start-up sequence:

- Applet initializes (AppMIDlet.init())
public void init(boolean showSplashScreen)
- Resources are initialized
resources = new Resources();
- File system is initialized
fileSystem = new FileSystem(Resources.ROOT_DIR);
fileStores = new FileStores();
- LWUIT is initialized (note that splash screen is already displayed)
initLWUIT();
- If the application was not registered it tries to register and register screen is displayed.
registerApp()
- Optionally Location Provider is registered and Survey List displayed

3. Example of UI flow. Views and classes

Entities that connect user actions with operations are named with suffix “Command” and derived from Event class. Those objects link user actions, like menu selection or pressing a button, with some operation on data

For example when user starts new survey by pressing “New result” element on ResultList, it will execute proper action. In this case “**NewResultCommand**”.

```
protected void doAction(Object parameter) {
#1   WaitingScreen.show(Resources.PROCESSING);
#2   NewResultRunnable orr = new NewResultRunnable();
#3   Thread t = new Thread(orr); //create new thread to compensate for waitingform
```

```
#4  t.setPriority(Thread.MIN_PRIORITY);
#5  t.start();
}
```

#1. On the screen progress bar will be displayed.

#2-#5 In the background survey is prepared;

In details:

```
#1  AppMIDlet.getInstance().getFileStores().resetQuestions();
#2  AppMIDlet.getInstance().setTimeTracker((new Date()).getTime());
#3  AppMIDlet.getInstance().getFileSystem().setLocalFile(false);
#4  AppMIDlet.getInstance().getFileStores().resetResultStructure();
#5  SurveysControl.getInstance().reset();
#6  SurveysControl.getInstance().resetQuestion();
#7  AppMIDlet.getInstance().setDisplayable(AppMIDlet.getInstance().getInterviewForm());
```

1 – preparing new survey

2 – timer is started

3-6 – preparing storage and data model to store result

7 – new screen with questions list is presented

Now user can collect answers for selected survey. After filling questionnaire result can be saved on phone memory. It is done by connecting “Save” button and “Save Result” menu option with command “**SaveResultCommand**”. This command fires PersistenceManager to save result. Two example command maps are described below.

```
protected void doAction(Object parameter) {
    Vector questions = (Vector)parameter;
    PersistenceManager.getInstance().save(questions);
}
```

Some examples of views with selected commands and actions performed or views displayed by those commands:

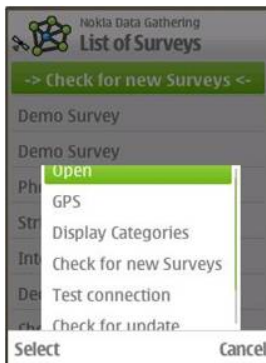


Figure 2 Screen SurveyList

Screen: SurveyList

Open: OpenSurveyCommand -> ResultList (also triggered by selecting item from list)

GPS: GPSCommand -> GPSForm

Display Categories: DisplayCategoryViewCommand -> DisplayCategoryForm

Check for new surveys: CheckNewSurveysCommand -> DownloadNewSurveys .check()

Test Connection: TestConnectionCommand -> TestConnection.doTest()

Check for update: UpdateCommand -> UpdateClientApp



Figure 3 Screen ResultList

Screen: ResultList

New Result: NewResultCommand -> [CategoryList or InterviewForm2]

Open: OpenResultCommand -> [CategoryList or InterviewForm2], with answers already loaded

View: ViewResultCommand -> ResultView

Send: SendResultNowCommand -> SubmitResultRunnable, SubmitServer

Delete: DeleteResultNowCommand -> SurveysControl.deleteResults()

Mark All: MarkAllResultsCommand -> renderer.markAll();

View Sent Results: ViewSentResultsCommand -> SentResultList



Figure 4 Screen GPSForm



Figure 5 Screen SimpleLocation



Figure 6 Screen DisplayCategoryForm

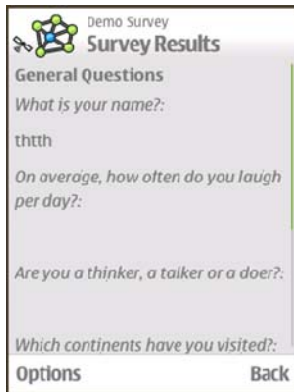


Figure 7 Screen ResultView



Figure 8 Screen SentResultList



Figure 9 Screen TestConnectionNewUI

4. UI map. Events and screens

Diagram as presented in the Figure 10 is the complete map of UI flow between Nokia Data Gathering classes. Arrows mean flow direction, description text mean commands or button actions, boxes are UI classes.

To make diagram more readable back and cancel commands are marked only as two way arrows with only one description. In cases where there is arrow with two descriptions, text near arrow point designates direction of a flow. “+” sign at the beginning of description means that two actions trigger the same UI flow.

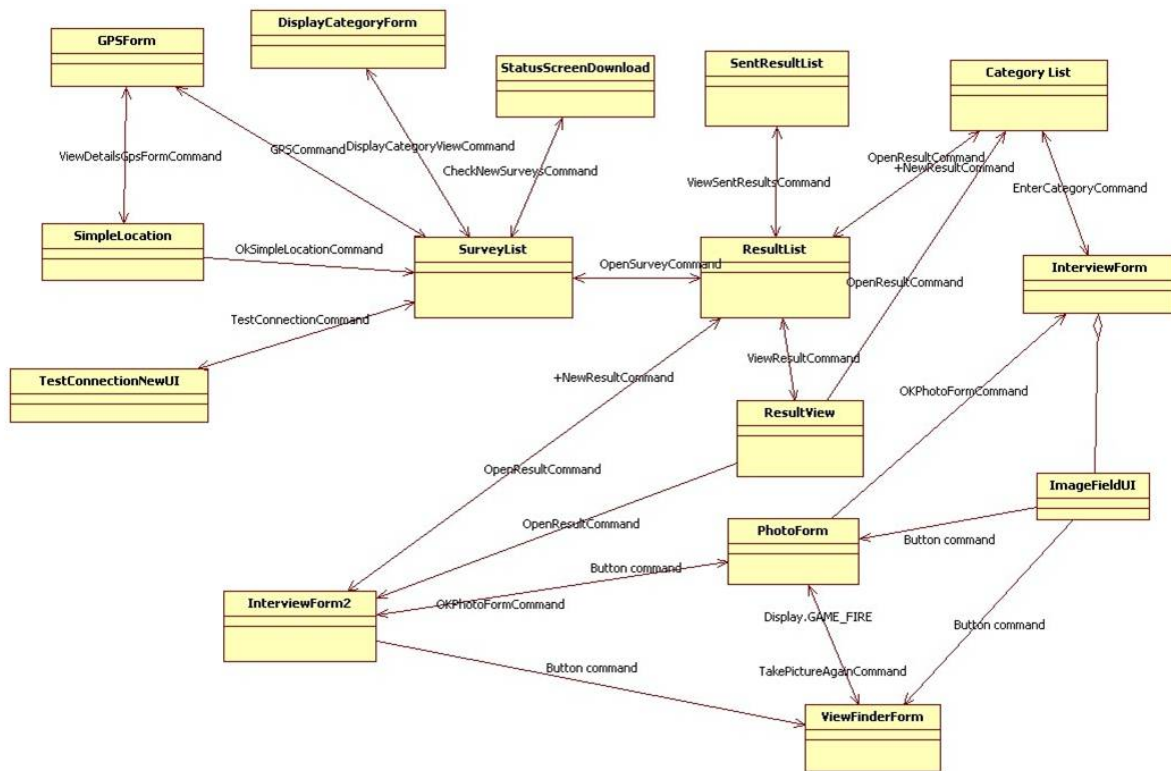


Figure 10 UI flow between Nokia Data Gathering classes

5. Building survey UI and data flow



Figure 11 Screen InterviewForm

InterviewForm shows questions in category mode while **InterviewForm2** shows questions without categories. Those classes are very similar. Both get list of questions from **SurveysControl** which is a central class for managing surveys. Then **InterviewForm** built one UI element from every question. Questions are divided into categories but all implement **NDGQuestion** interface and contain Answer field, which holds user answer. Once user inputs necessary data and decides to save the result, **SaveResultCommand** is used to pass filled questions to **PersistenceManager** which saves them into xml file.

All results and surveys are hold in xml files that can be found in *root\ndg* directory and are loaded by **FileStores** class by proper command (ex. **OpenSurveyCommand**).

Example flow of data for input of descriptive question:

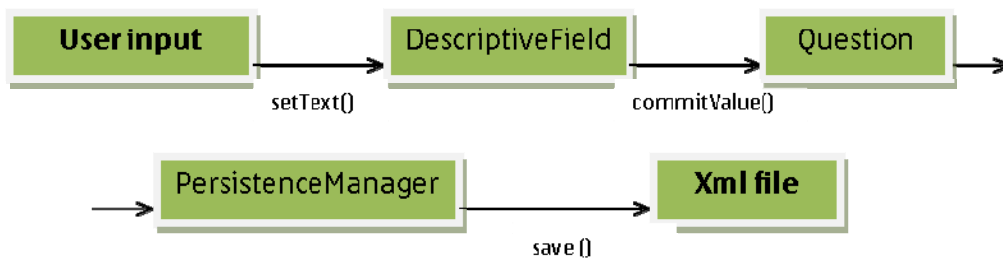


Figure 12 Data flow example I

Example flow of data for sending of descriptive question:



Figure 13 Data flow example II

6. Send and receive data

Server setting can be found in `|root|ndg|settings.xml`. This file also contains application all persistent application settings.

Results are send after proper command (for example `SendResultNowCommand`) in the form previously saved of xml files. This is managed by `SubmitServer`. Xmls are compressed as an option.

Downloading surveys is done mainly by `DownloadNewSurveys` class.

7. Parsing XML files

KParser: parses survey file with details

Parser: SAX parser that uses additional handlers to parse. With `SurveyHandler` used to parse survey title and survey ID, with `SettingsHandler` parses settings.xml file, with `FileSystemResultHandler` parser result files titles and finally with `ResultHandler` parses result file to result structure.

8. Localization (GPS) module

To manage location JSR-179 API is used and it is mainly managed by `LocationHandler` class. API does not have many possibilities to use so once application tries to obtain a location, it has no control whether location comes from internal GPS, Bluetooth connected GPS or from network location services. To change localization source (particularly for Bluetooth GPS connection) `UpdateProvider` runnable is used and it allows restart of location provider if new location data source is connected.

9. Camera

Camera actions are managed by few classes: `PhotoForm`: displays taken photo, `ViewFinderForm` displays viewfinder used to take photo, `Camera` that is connection layer to hardware API and `NDGCameraManager` which central point of control.

```
try
{
    //for S40 devices
    player = Manager.createPlayer("capture://image");
}
catch(Exception ex)
{
    //for S60 devices
    player = Manager.createPlayer("capture://video");
}
```