Receiving Game Data

Support Guide

1. Introduction

This document describes the needed actions to enable receiving of game data in Series60. The main component for content handling in Series60 is Document Handler, which is used by all communication applications, including WAP Browser, Mail, IR and Bluetooth. The game data exchange architecture is a special case, but still makes use of Document Handler. For further information about he Document Handler refer to Series60 SDK documentation.

The data format recommended for Series60 games is shared with all Nokia phones, also non-Symbian OS based. The main goal for common format is to archive compatibility between devices. This data format must be used in Series60 to be able to take advantage of the game data exchange framework.

When Document Handler receives data from some of its sources and the content is recognized as game data, it copies the content to a game specific directory where the game application can find it. What happens next is in hands of the game developers.

2. Check List

2.1 Register the Global Game ID

In Series60, this Game ID is agreed to be the same as the Symbian OS application UID for the game. Game ID's from 0 to 1000 are reserved for internal use.

2.2 Declare the mime type in the AIF file

The MIME type is application/x-NokiaGameData-<GAME_ID>, where <GAME_ID> is the registered game id in hex numbers without "0x" part. For example: application/x-nokiaGameData-01234567. Here is an example of an AIF file.

```
//
// MyCoolGame.RSS
//

#include <aiftool.rh>
RESOURCE AIF_DATA
   {
     // Example game's application UID
     app_uid = 0x10009ACD;
     // Number of icons to be included in AIF file
     num_icons = 1;
     // Application capabilities.
     embeddability = KAppNotEmbeddable;
     hidden=KAppNotHidden;
     // MIME types application supports
     datatype_list=
         {
         DATATYPE
            {
            priority = EDataTypePriorityNormal;
            type = "application/x-nokiaGameData-012434567";
            },
            {
             // Other MIME types can be added here if needed.
            }
         };
      } // AIF_DATA
```

2.3 Declare the data directory

Declare your game's data directory with a SharedData keyword. Document Handler must know the correct place in which to save your data files. Here is how to do it:

1. Write a plain text file containing Keyword-value pair. The keyword must be SDDataDir and the value is a relative path starting form global game root. For example, SDDataDir=myCoolGame. In version 1.0 of Series 60 the root is always "c:\nokia\games".
2. Save the file in Unicode format.
3. Name the file as <myAppid>.ini, where <myAppid> part is your application ID in hex without the "0x" part. For example, 10039110.ini

Copy the file from the installation package to c:\system\SharedData directory. Here is an example of a line in the application installation package descriptor:
"\MyFirmApp\group\10039110.ini"-"!:\system\SharedData\10039110.ini "

Tip: If your game can handle multiple game IDs and needs separate directories for the data files, here is the way to do it:
Append '-' and a MIME type that you support after the normal SDDataDir key:

Example:

SDDataDir**-application/x-nokiaGameData-01234567**=myCoolGame\one

SDDataDir**-application/x-nokiaGameData-56789012**=myCoolGame\two

2.4 Declare default data file name in resources.

Sometimes the original file name is lost when the data file is handed over to Document Handler. Therefore the application developers can declare a default name for the framework. The correct place for the name is in the application's resources. If the name is not available, Document Handler uses a global name for documents. Example:

```
// RESOURCE DEFINITIONS
// These are standard Symbian OS resources
RESOURCE RSS_SIGNATURE { }
RESOURCE TBUF { buf = (localized)_default_name; }
RESOURCE EIK_APP_INFO
    {
    }
```

Warning: If you do declare your default data file name in resources you must inherit your document class from CAknDocument. The default implementation of CApaDocument class creates a scratch document with this name. The behavior will, in out of disk situations, prevent the device to start up.

3. Game Data File Structure

The common game data file structure is described in detail below.

File Header Format

	Starts at byte
Game ID	0
Data Type	4
Data Length (bytes)	5
Size of name string (X Unicode charectes)	9
Name String	10
Data ID	10+2x
Data version	10+2x+4
NGDX header	10+2x+5
[Data]	10+2x+9

Game ID: A unique number identifying each game (e.g. Snake2).

Data Type: Can be used to distinguish between, for example, a level, a character, a weapon… for the same game.

Data Length: Size of the following field as 32bit number.

Size of Name String: 1 byte so max 255 characters identifying size of following field in Unicode (double-byte) characters. Can be zero.

Name String: Unicode string, which the game can use to register as a user visible text for selection of this data item from the menus.

Data ID: A unique number identifying the data content. This is a 32bit number, which is unique for this type of data.

Data Version: A number to allow data compatibility checking by the game. Typical use will be to use different version number for data for different size displays.

NGDX Header: Identification for NGDX game data. 4 bytes. String "NGDX" in ASCII characters. Not present in original engine games, but mandatory for all Series60 games. The recognition of data file type can be based on this field.