

# Amsterdam prijsobject/stadsmarkering system architecture

Simon de Bakker

December 6, 2005

## 1 General description

Amsterdam prijsobject / stadsmarkering is the location dependent sending of content to people passing by with bluetooth-enabled devices. Devices, based on Gumstix<sup>TM</sup> hardware, represent prizes awarded bij the AFK (Amsterdam Fonds voor de Kunsten). Developed in cooperation with Airplant.

This document describes the software and hardware components developed for the Amsterdam prijsobject project. For the software there are 3 main software components:

- OBEX Push implementation
- administration service
- automation layer

The OBEX Push implementation consists of a library, libbtopush, to make device discovery and pushing/getting objects to/from bluetooth enabled devices easier, and a program utilizing this library. Next, the administration service makes it possible to administer the device using bluetooth. Last is the automation layer which is a simple shell script that incorporates the program and the administration service.

## 2 OBEX Push implementation

To send content to bluetooth enabled devices we make use of the Object Exchange (OBEX) protocol by the Infrared Data Association (IrDA<sup>TM</sup>). A drawback of this method is that the device has to support given protocol. However, most modern bluetooth enabled devices do so. We only utilize the PUSH and GET methods (OPUSH) of the protocol.

Our OPUSH implementation (btopush) is build upon the OpenOBEX<sup>1</sup> and BlueZ<sup>2</sup> library. Second part of the OBEX Push implementation is the bluespam program utilizing the libbtopush library.

---

<sup>1</sup><http://openobex.sourceforge.net>

<sup>2</sup><http://bluez.sourceforge.net>

## 2.1 btopush library

This section briefly describes the btopush library functionality:

- discovery of OPUSH capable devices
- opening and closing a connection
- pushing/pulling data to/from OPUSH capable devices
- getting device information

For a detailed API description please refer to the btopush API specification.

### 2.1.1 Discovery of OPUSH capable devices

The process of OPUSH capable device discovery with libtopush is a simple process. Only needed is a call to

```
int btopush_inq_objpush(btopush_dev_t * dev)
```

with an array of *btopush\_dev\_t* large enough to contain *BTOPUSH\_MAX\_DEV* devices. The routine issues a general inquiry. Next a SDP service search is issued for every device. If a device has the OBJPUSH service class identifier set it is queried for the channel and added to the device array. The function returns after all found devices are probed in this manner.

### 2.1.2 Opening and closing a connection

Before any communication with a remote device can occur a connection has to be established. This is a two step process. First a RFCOMM link with the remote devices has to be opened. Next a connection request has to be sent to the remote device. If the remote device acknowledges the connection request the connection is established and further communication can take place. btopush provides this functionality with two methods.

Opening the RFCOMM link call:

```
int btopush_open(btopush_ctx_t * ctx)
```

and establishing the connection

```
int btopush_connect(btopush_ctx_t * ctx)
```

After all communication has taken place it is important that the connection is closed and the link destroyed. This is done by calling following methods respectively:

```
int btopush_disconnect(btopush_ctx_t * ctx)
int btopush_close(btopush_ctx_t * ctx)
```

### 2.1.3 Pushing/pulling of data to/from OPUSH capable devices

Pushing data is only a matter of opening a file and pushing it over the established connection. This is done by calling

```
int btopush_open_file(btopush_ctx_t * ctx, char * file)
```

and pushing the data over the open connection

```
btopush_push_stream(btopush_ctx_t * ctx)
```

If all has gone well close the file

```
int btopush_close_file(btopush_ctx_t * ctx)
```

Pulling a file from the remote device involves calling only one method:

```
int btopush_get(btopush_ctx_t * ctx, char * name)
```

After receiving has finished the data can be found in *ctx->dev.data*. Currently no method for retrieving this data is available.

### 2.1.4 Getting device capability

Getting device capability information is based on the IrMC standard. Although the btopush implementation is very minimal it is possible to receive the telecom/devinfo.txt, telecom/pb/info.log and the telecom/cal/info.log from devices that support these standard locations. The more general capability service based on MIME-type, as specified in the OBEX standard, is not implemented. To retrieve the capability information btopush provides the following three methods:

```
int btopush_get_devinfo(btopush_ctx_t * ctx)
int btopush_get_pbinfo(btopush_ctx_t * ctx)
int btopush_get_calinfo(btopush_ctx_t * ctx)
```

## 2.2 The bluespam program

The core of the system is the bluespam program. Bluespam is a simple program using the btopush library scanning for OPUSH capable devices. Once found the device is checked against a database to verify it has not yet received a message. If the device exists in the database it is ignored, else the program connects and sends the content. In case this procedure fails, the routine is repeated for three subsequent scans. If it is still not possible to connect or send the content the device is added to the database and further ignored. Devices for which sending succeeds the first time are added to the database immediately, no further attempts are made to connect to this device again. Bluespam is called periodically from the automation layer (section 4) on which it performs one discover and push cycle and exits.

Locations of the database and content are hardcoded in the program. They are defined as follows:

```
#define BSP_DB_ENV_HOME “/mnt/mmc/db”
#define BSP_DB_NAME “btaddr.db”
#define BSP_FILEDIR “/mnt/mmc/content”
```

If the content directory contains more than one file the files are send in version order<sup>3</sup>.

If the bluespam program is issued with a bluetooth address as argument (e.g. ./bluespam 00:0F:DE:0E:E2:A8) the program will only check for the existence of the address in the database and remove the address if it does. This feature can be usefull when testing.

### 3 Administration service

Together with the bluespam program comes a remote administration service based on the OBEX protocol. The administration service is implemented as a modified version of the opd<sup>4</sup> server. Commands are send to the server as files. If a command takes an argument, this is done within the body of the file. The server accepts 3 different command types:

- files with a .frm extension are treated as firmware updates
- files with a .cmd extension are treated as actions to be performed
- files with no extensions are treated as new content

<i>Command</i>	<i>Function</i>	<i>Argument</i>
add_admin.cmd	Add a trusted bluetooth device.	Bluetooth address
rem_admin.cmd	Remove a trusted bluetooth device.	Bluetooth address
remove.cmd	Remove a device from the 'ignore' list.	Bluetooth address
name.cmd	Rename the device.	Name string
stop.cmd	Stops scanning for devices.	<i>na</i>
start.cmd	Starts scanning for devices.	<i>na</i>

Because the scanning process takes up most of the bandwidth sending a command might fail. For this reason it is best to first send a stop.cmd so the scanning process is stopped, which makes the administration service more responsive. To reanable the scanning process send a start.cmd.

### 4 Automation layer

The automation layer is a shell script started at device startup taking care of a number of tasks:

- periodically execute the bluespam program
- start and monitor the administrator service
- take care of locking
- execution of the remove, name, stop and start commands received by the administrator service

---

<sup>3</sup>For detailed sorting rules see the manual page of strverscmp(3)

<sup>4</sup>opd is an OBEX Push daemon written by S. Dauskardt among others, distributed under the GPL license.

- content replacement

## 5 Requirements

The Amsterdam prijsobject software runs on any normal linux architecture. The standard Unix shell (sh) or the Bourne-Again SHell (bash) is needed to run the automation layer.

### 5.1 btopush library

- openobex  $\geq 1.0.1$
- bluez-libs  $\geq 2.15$

### 5.2 bluespam program / administrator service

- btopush 1.0
- berkeley db  $\geq 4.2.25$