libpackedobjectsd tutorial

libpackedobjectsd - a light weight XML messaging framework

Table of Contents

1 Introduction

1.1 What is libpackedobjectsd?

libpackedobjectsd is a light-weight XML messaging library. It is created using multiple publisher and subscriber model and built using libpackedobjects and ZeroMQ in C language. It is simple to implement and is suited to embedded systems and mobile devices. The library provides simple API to send and receive XML data to and from multiple nodes. The library connects all the nodes sending/receiving XML data using the same XML schema to one group. The data is also validated by the schema during the send and receive process.

libpackedobjectsd is based on libpackedobjects, libxml2 and ZeroMQ and therefore should run on any system that libxml2 and ZeroMQ runs on.

1.2 Key features

- Light-weight and simple
- Validates XML data on send and receive
- Simple API with two main function calls
- Highly portable - designed for embedded and mobile devices
- Simple subset of XML Schema required to create protocols

1.3 Limitations

todo

2 Installation

2.1 Installing packedobjectsd

To install from the latest source:

```
git clone git://gitorious.org/libpackedobjectsd/libpackedobjectsd.git
cd libpackedobjectsd
autoreconf -i
./configure
make
make check
sudo make install
```

2.2 Further reading

3 Getting started

3.1 Quick start

After compiling and running 'make check' you should find a binary called 'packedobjectsdtest' in your src directory. This is command-line tool built with packedobjectsd which you can use to test out sending and receiving:

```
$ ./packedobjectsdtest --help
usage: packedobjectsdtest --schema <file> --xml <file>
```

To send and receive run:

```
$ ./packedobjectsdtest --schema foo.xsd --xml foo.xml
```

3.2 API basics

There are only 4 main function calls which are made available by adding `#include <packedobjectsd/packedobjectsd.h>` to your code.

```
packedobjectsdObject *init_packedobjectsd(const char *schema_file);

int packedobjectsd_send(packedobjectsdObject *pod_obj, xmlDocPtr doc);

xmlDocPtr packedobjectsd_receive(packedobjectsdObject *pod_obj);

void free_packedobjectsd(packedobjectsdObject *pod_obj);
```

You first must initialise the library using your XML Schema. Typical use would be one called to init_packedobjectsd at startup and then multiple calls to send/receive data. The interface to the packedobjectsd_send function requires a libxml2 doc type and returns number of bytes sent. The packedobjectsd_receive function returns a libxml2 doc type. A very simple program demonstrating the API is as follows:

```
#include <stdio.h>
#include <unistd.h>      /* for sleep() */
#include <packedobjectsd/packedobjectsd.h>

#define XML_DATA "helloworld.xml"
#define XML_SCHEMA "helloworld.xsd"


int main ()
{
  packedobjectsdObject *pod_obj = NULL;
  xmlDocPtr doc_sent = NULL;
  xmlDocPtr doc_received = NULL;

  ////////////////////// Initialising //////////////////

  /* Initialise packedobjectsd */

  if((pod_obj = init_packedobjectsd(XML_SCHEMA)) == NULL) {
    printf("failed to initialise libpackedobjectsd");
```

```
    exit(1);
  }

  sleep(1); /* Allow broker to start if it's not already running */

  ////////////////////// Sending XML data //////////////////////

  /* create an XML DOM */
  if((doc_sent = xml_new_doc(XML_DATA)) == NULL) {
    printf("did not find .xml file");
    exit(1);
  }

  /* send the XML DOM */
  if(packedobjectsd_send(pod_obj, doc_sent) == -1){
    printf("failed to send with error %s", pod_strerror(pod_obj->error_code));
    exit(1);
  }

  printf("size of the original xml: %d bytes\n", xml_doc_size(doc_sent));
  printf("size after the encoding: %d bytes\n", pod_obj->bytes_sent);
  /* free the XML DOM */
  xmlFreeDoc(doc_sent);

  ////////////////////// Receiving XML data //////////////////////

  if((doc_received = packedobjectsd_receive(pod_obj)) == NULL) {
    printf("failed to receive with error %s", pod_strerror(pod_obj->error_code));
    exit(1);
  }

  printf("size before the decoding: %d bytes\n", pod_obj->bytes_received);
  printf("size of the decoced xml: %d bytes \n", xml_doc_size(doc_received));

  /* output the DOM for checking */
  xml_dump_doc(doc_received);
  /* free the XML DOM */
  xmlFreeDoc(doc_received);

  ////////////////////// Freeing //////////////////////

  /* free memory created by packedobjectsd */
  free_packedobjectsd(pod_obj);

  return 0;
}
```

helloworld.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:include schemaLocation="http://zedstar.org/xml/schema/packedobjectsDataTypes.xsd"/>
              <xs:element name="foo" type="string"/>
</xs:schema>
```

helloworld.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<foo>Hello World!</foo>
```

If during runtime your schema changed you must call the init function again with the new file. The library is designed to communicate to the server during the init function which gives it back the network address and port numbers to send or receive the data. Therefore, do not call init_packedobjectsd more than once if you do not plan on supporting dynamically changing protocols at runtime.

To build an application with the software you must link with the library. Using autoconf you can add `PKG_CHECK_MODULES([LIBPACKEDOBJECTSD], [libpackedobjectsd])` to your configure.ac file and then use the variables `$(LIBPACKEDOBJECTSD_CFLAGS)` and `$(LIBPACKEDOBJECTSD_LIBS)` in your Makefile.am file.

A

F

I

K

L

Q

W