GammaLib - Action #781

Feature # 591 (In Progress): Investigate how GammaLib can be made VO compliant.

Design SAMP usage in GammaLib

02/20/2013 02:35 PM - Knödlseder Jürgen

Status:	Closed	**Start date:**	02/20/2013
Priority:	Normal	**Due date:**	
Assigned To:	Knödlseder Jürgen	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

VO compliant applications communicate through the SAMP protocol. Communication runs over a Hub.

GammaLib should be able to register to the Hub and to exchange data using the SAMP protocol.

Here a couple of links that may be relevant:

- SAMP software
- sampy

History

#1 - 03/11/2013 10:41 PM - Knödlseder Jürgen

Here an example samp lockfile:

```
# SAMP lockfile written 2011-12-22T05:30:01
# Required keys:
samp.secret=734144fdaab8400a1ec2
samp.hub.xmlrpc.url=http://andromeda.star.bris.ac.uk:8001/xmlrpc
samp.profile.version=1.3
# Info stored by hub for some private reason:
com.yoyodyne.hubid=c80995f1
```

And here some peusdo-code of how an application might locate and register with a hub, and send a message requiring no response to other registered clients:

```
# Locate and read the lockfile.
string hubvar-value = readEnvironmentVariable("SAMP_HUB");
string lock-location = getLockfileLocation(hubvar-value);
map lock-info = readLockfile(lock-location);

# Extract information from lockfile to locate and register with hub.
string hub-url = lock-info.getValue("samp.hub.xmlprc.url");
string samp-secret = lock-info.getValue("samp.secret");

# Establish XML-RPC connection with hub # (uses some generic XML-RPC library)
xmlrpcServer hub = xmlrpcConnect(hub-url);

# Register with hub.
map reg-info = hub.xmlrpcCall("samp.hub.register", samp-secret);
string private-key = reg-info.getValue("samp.private-key");

# Store metadata in hub for use by other applications.
map metadata = ("samp.name" -> "dummy", "samp.description.text" -> "Test Application", "dummy.version" -> "0.1-3");
hub.xmlrpcCall("samp.hub.declareMetadata", private-key, metadata);

# Send a message requesting file load to all other
# registered clients, not wanting any response.
map loadParams = ("filename" -> "/tmp/foo.bar");
map loadMsg = ("samp.mtype" -> "file.load", "samp.params" -> loadParams);
hub.xmlrpcCall("samp.hub.notifyAll", private-key, loadMsg);
```

```
# Unregister
hub.xmlrpcCall("samp.hub.unregister", private-key);
```

#2 - 03/11/2013 10:59 PM - Knödlseder Jürgen

Here a draft of the GVOClient class that establishes the connection to a Hub for a client:

```
class GVOClient : public GBase {
    // Constructors and destructors
    GVOClient(void);
    GVOClient(const GVOClient& client);
    virtual ~GVOClient(void);

    // Operators
    GVOClient& operator=(const GVOClient& client);

    // Methods
    void        clear(void);
    GVOClient*  clone(void) const;
    void        register(void) const;
    void        unregister(void) const;
    std::string print(void) const;

protected:
    // Protected methods
    void init_members(void);
    void copy_members(const GVOClient& client);
    void free_members(void);
    bool find_hub(void);

    // Protected data area
    std::string m_secret;      //!< Secret Hub key
    std::string m_hub_url;     //!< The XML-RPC endpoint for communication with the hub
    std::string m_version;     //!< The version of the SAMP Standard Profile implemented by the hub
    std::string m_client_key;  //!< Client key
};
```

The GVOClient constructor automatically searches for a Hub and connects with the Hub. If no Hub is found, a GammaLib own hub should be started. The GammaLib own Hub will be implemented by the GVOHub class.

The find_hub() method performs the Hub discovery using the following steps:

1. Determine where to find the lockfile
2. Read the lockfile to obtain the hub connection parameters

The default location of the lockfile is the file named ".samp" in the user's home directory. However the content of the environment variable named SAMP HUB can be used to override this default. Thus, find_hub() will first check whether SAMP_HUB exists. The format is SAMP_HUB=std-lockurl:file:///tmp/samp1. To locate the lockfile therefore, a Standard Profile-compliant client MUST determine whether an environment variable named SAMP HUB exists; if so, the client MUST examine the variable's value; if the value begins with the prefix "std-lockurl:" the client MUST interpret the remainder of the value as a URL whose content is the text of the lockfile to be used for hub discovery. If no SAMP HUB environment variable exists, the client MUST use the file ".samp" in the user's home directory as the lockfile to be used for hub discovery. If the variable exists, but its value begins with a different prefix, the client MAY interpret that in some non-Standard way for hub discovery.

The "home directory" referred to above is a somewhat system-dependent concept: we define it as the value of the HOME environment variable on Unix- like systems

The find_hub() method will returns true if a hub was found, false otherwise.

#3 - 03/12/2013 10:00 AM - Knödlseder Jürgen

- Target version set to 00-08-00

#4 - 03/12/2013 10:21 AM - Knödlseder Jürgen

- Status changed from New to In Progress

- % Done changed from 0 to 10

Start implementation of GVOClient.

#5 - 03/12/2013 10:27 AM - Knödlseder Jürgen

Concerning socket programming, see http://www.linuxhowtos.org/C_C++/socket.htm.

The steps involved in establishing a socket on the client side are as follows:

1. Create a socket with the socket() system call
2. Connect the socket to the address of the server using the connect() system call
3. Send and receive data. There are a number of ways to do this, but the simplest is to use the read() and write() system calls.

Here an example code:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

void error(const char *msg)
{
    perror(msg);
    exit(0);
}

int main(int argc, char *argv[])
{
    int sockfd, portno, n;
    struct sockaddr_in serv_addr;
    struct hostent *server;

    /* Input handling */
    char buffer[256];
    if (argc < 3) {
       fprintf(stderr,"usage %s hostname port\n", argv[0]);
       exit(0);
    }
    portno = atoi(argv[2]);

    /* Create socket */
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
        error("ERROR opening socket");

    /* Set server address */
    server = gethostbyname(argv[1]);
    if (server == NULL) {
        fprintf(stderr,"ERROR, no such host\n");
        exit(0);
    }
    bzero((char *) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    bcopy((char *)server->h_addr, (char *)&serv_addr.sin_addr.s_addr, server->h_length);
    serv_addr.sin_port = htons(portno);

    /* Connect to server */
    if (connect(sockfd,(struct sockaddr *) &serv_addr,sizeof(serv_addr)) < 0)
        error("ERROR connecting");
```

```
    /* Send message */
    printf("Please enter the message: ");
    bzero(buffer,256);
    fgets(buffer,255,stdin);
    n = write(sockfd,buffer,strlen(buffer));
    if (n < 0)
         error("ERROR writing to socket");

    /* Retrieve message */
    bzero(buffer,256);
    n = read(sockfd,buffer,255);
    if (n < 0)
         error("ERROR reading from socket");
    printf("%s\n",buffer);

    /* Close socket */
    close(sockfd);

    /* Ciao ... */
    return 0;
}
```

#6 - 03/12/2013 11:15 AM - Knödlseder Jürgen

- Assigned To set to Knödlseder Jürgen

#7 - 03/13/2013 02:13 AM - Knödlseder Jürgen

- % Done changed from 10 to 30

Things are progressing. Hub registration and Meta data publishing is working.

There is only one thing I don't understand so far: For each write/read action by the client the Hub needs to be connected again. Why is this so?

#8 - 03/14/2013 05:26 PM - Knödlseder Jürgen

Jürgen Knödlseder wrote:

> There is only one thing I don't understand so far: For each write/read action by the client the Hub needs to be connected again. Why is this so?

This somehow seems the normal behavior. Once all data are read through the socket the connection is closed. Thus each Hub access requires that connect_hub is called first.

#9 - 03/14/2013 05:26 PM - Knödlseder Jürgen

- % Done changed from 30 to 80

The only element that is missing is disconnection of the client from the Hub. Will now attack that one ...

#10 - 03/14/2013 06:11 PM - Knödlseder Jürgen

- Status changed from In Progress to Feedback

- % Done changed from 80 to 100

- Remaining (hours) set to 0.0

Done.

Controlled using integration branch, merged into devel.

#11 - 12/03/2013 10:00 AM - Knödlseder Jürgen

- Status changed from Feedback to Closed

- Estimated time set to 0.00

#12 - 12/10/2013 12:17 AM - Knödlseder Jürgen

- Target version deleted (00-08-00)