EDOC089

EUROGAM PROJECT

NSF DATA ACQUISITION SYSTEM

Message Handling System

Edition 1.1
June 1994

NPSG Software Systems Group
UK Engineering and Physical Sciences Research Council
Daresbury Laboratory

Introduction

This document describes the Message Handling System which is a part of the Eurogam Data Acquisition System.

It is intended that these protocols be available over a number of communications media and hence the relevant *Communications Protocol* specification document should be consulted for detail in this area.

This protocol is an application carried by the RPC protocol which is described in the Eurogam document *Remote Procedure Call.*

The general structure of the protocol is described in the Eurogam document *RPC Application Programs.*

Document History

Version 1.0 Sept 1991 - Initial version.

Version 1.1 June 1994 - Allow source and system names up to 16 bytes long.

Use

The message and error logging system comprises of three components each of which equates to a simple RPC program protocol. This design allows the reporting and distribution of messages to be separated and the RPC methodology to be strictly followed.

Part One handles the definition of messages. A client wishing to report an exception condition or log activity sends the message text via RPC to a Central Message Server. Clients typically will be software running in the VME processors such as the Event Builder or applications controlling the Data Acquisition hardware.

Part Two handles requests to the Central Message System from clients who wish to be informed of the generation of messages. A client can register an interest in receiving a copy of a particular subset or number of subsets of the messages sent to the Central Message System. Clients typically will be software running in the user workstations.

Part Three handles the distribution of messages to interested parties. In this case the Central Message System acts as the client and sends copies of messages to server software running in the workstations nominated to receive them.

All messages are allocated a class and severity level by the generating client and a workstation can nominate to be informed for example of all messages of a given class (say all messages from the Event Builder) or all messages of higher than a given severity level (say all serious errors or worse) or any combination of these conditions.

The Central Message Server in addition will timestamp and record on disc all received messages for subsequent offline analysis as desired.

Severity Level

The following are suggested values

0	
1	test information
2	debug information
3	
4	warning
5	
6	error
7	alarm
8	failure
9	catastrophic

Class

To be supplied.

Part One

Message Logger Program

Allows clients to report exceptions, errors or monitoring information to a central server.

Program Specification

RPC Information

This service uses only the Null authentication

This service is supported on the UDP/IP transport only

This service is RPC Program 28000240

This is Version 1 of the protocol

This service uses UDP port number 10240

XDR Information

Basic Data Types

```
typedef   enum {
        OK = 0


} stat;

typedef   string   message_source<16>;
typedef   string   message_body<100>;

typedef   unsigned   message_id;
typedef   unsigned   message_class;
typedef   unsigned   message_level;
```

Procedure 0. Null procedure

This procedure does no work. There are no input parameters and no returned results. It is made available in all RPC services to allow server response testing and timing.

Procedure 1. Define Message

REQUEST

Sends an message or error report to the Message Server.

message id
class
level
source
body

Message id is a client generated field which will be used to distinguish between messages and possible retransmissions. This field must be different in successive procedure requests. If a client only generates one message at a time then *message id* need only alternate between 0 and 1.

Class identifies the general nature of the message and will have a value in the range 0 to 99.

Level identifies the degree of importance of the message and will have a value between 0 (a strictly informative message such as a process 'hello' message) and 9 (a fatal message such as an unrecoverable system error).

Source is a text string which identifies the generating task.

Body is a text string containing the message.

RESPONSE

procedure status
message id

Procedure status is a field which indicates the result of the request.

Message id is returned as supplied in the request.

Procedure status= 0 OK

XDR Specification

```
procedure   DefineMessage(id,class,level,source,body)   returns (status,id)
            message_id   id ;
            message_class   class ;
            message_level   level ;
            message_source   source ;
            message_body   body ;
            stat   status ;
```

Procedure Interface

definemessage(id,class,level,source,body)

id, *class*, and *level* are integer parameters. *source* and *body* are character string parameters.

Message Log File

The server will maintain a message log file using a new file each day and generate a record in the file for each received message. The entries will be entirely text, will be time stamped and in a fixed format (except for the variable length message body).

bytes	0-7	time stamp
byte	8	space
bytes	9-10	class
byte	11	space
byte	12	level
byte	13	space
bytes	14-29	source
byte	30	space
bytes	31 on	message body

The maximum record length is thus 131 bytes.

Part Two

Message Signon Program

Allows clients to register an interest in receiving copies of messages of a given class and level range.

The client may either register on behalf of its own workstation or some other.

Program Specification

RPC Information

This service uses only the Null authentication

This service is supported on the UDP/IP transport only

This service is RPC Program 28000240

This is Version 1 of the protocol

This service uses UDP port number 10240

XDR Information

Basic Data Types

```
typedef  enum  { FALSE = 0, TRUE =1 } boolean;
typedef  enum {
        OK = 0


} stat;
typedef  string  system_name<16>;
typedef  unsigned  message_class;
typedef  unsigned  message_level;
```

Procedure 0. Null procedure

This procedure does no work. There are no input parameters and no returned results. It is made available in all RPC services to allow server response testing and timing.

Procedure 2. Signon

REQUEST

Allows clients to register an interest in receiving copies of messages.

name length
name
valid=true
class
low level
high level
multiplex count
:
valid=false

Name is a string containing the network name of the workstation on whose behalf this request is made. A null length string implies that the request is for the client workstation.

Following are a number of Signon records each identifying a catagory of message to be received. For each selection the class and range of levels (from *low level* to *high level* inclusive) of the messages required are specified.

Multiplex count contains the maximum number of messages the workstation is prepared to accept via the Message Distribution Program before it must send a response.

Each workstation may make only one *Signon* request which may contain as may selection requests as desired. A *Signon* request overwrites any existing request for that workstation.

RESPONSE

Procedure status is a 4 byte field which indicates the result of the request.

procedure status

Procedure status= 0 OK

XDR Specification

```
procedure   Signon(name,item)   returns (status)
            system_name   name ;
               typedef   union switch (boolean valid)   {
                 TRUE:
                 struct {
                    message_class   class;
                    message_level   low_level;
                    message_level   high_level;
                    unsigned   count;
                    entry   nextentry;
                 }
                 FALSE:
                 struct {}
               } entry;
            stat   status ;
```

Procedure 3. Signoff

REQUEST

Cancels any outstanding Signon for the workstation identified by *name*.

name length
name

RESPONSE

Procedure status is a 4 byte field which indicates the result of the request.

procedure status

Procedure status= 0 OK

XDR Specification

```
procedure   Signoff(name)   returns (status)
            system_name   name ;
            stat   status ;
```

Part Three

Message Distribution Program

Allows messages received by the central message server to be distributed to parties which have previously expressed an interest by using the Signon procedure of the Message Signon Program.

Program Specification

RPC Information

This service uses only the Null authentication

This service is supported on the UDP/IP transport only

This service is RPC Program 28000242

This is Version 1 of the protocol

This service uses UDP port number 10242

XDR Information

Basic Data Types

```
typedef  enum  { FALSE = 0, TRUE =1 } boolean;

typedef  enum {
        OK = 0


} stat;

typedef  string  message_source<16>;
typedef  string  message_body<100>;
typedef  string  system_name<16>;

typedef  unsigned  message_id;
typedef  unsigned  message_class;
typedef  unsigned  message_level;
```

Procedure 0. Null procedure

This procedure does no work. There are no input parameters and no returned results. It is made available in all RPC services to allow server response testing and timing.

Procedure 1. Receive Message

REQUEST

This has the same format as the Define Message procedure in the Message Logger Program.

message id
class
level
source
body

RESPONSE

Procedure status is a 4 byte field which indicates the result of the request.

Procedure status= 0 OK

procedure status
message id

XDR Specification

```
procedure   ReceiveMessage(id,class,level,source,body)   returns (status,id)
            message_id   id ;
            message_class   class ;
            message_level   level ;
            message_source   source ;
            message_body   body ;
            stat   status ;
```