EDOC034

EUROGAM PROJECT

NSF DATA ACQUISITION SYSTEM

Eurogam Register Server Protocol

Edition 1.1
May 1991

NSF Software Systems Group
UK Science and Engineering Research Council
Daresbury Laboratory

Introduction

This document describes a general purpose protocol used within the Eurogam Data Acquisition System to enable clients to communicate with the various applications programs which control equipment and acquire data.

The general concepts of the Register Server are defined in the document *EDOC079* (The Eurogam Register Server - An Overview) and an example of its use is described in the document *EDOC024* which defines the requirements for the *Eurogam Frontend Crate Server.*

It is intended that this protocol 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

Edition 1.1 May 1991 - first entry in Eurogam document server

Operation

Program Specification

RPC Information

The ERS service uses only the Null authentication

ERS is supported on the UDP/IP transport only

ERS is RPC Program 28000220

This is Version 1 of the protocol

ERS uses UDP port number 10220

XDR Information

Basic Data Types

```
typedef  enum  { FALSE = 0, TRUE =1 } boolean;
typedef  enum {
        OK = 0,
        Resource_not_known = 1,
        Resource_already_in_use = 2,
        Capability_invalid = 3,
        Register_not_known = 4,
        Register_already_defined = 6,
        Prototype_not_known = 7,
        Application_not_responding = 8,
        Invalid_cookie = 9,
        Application_detected_failure = 10,
        Illegal_register_name = 11,
        Out_of_memory = 12,
        Count_too_small = 13,
        Bad_request = 14
} stat;

typedef  string  resource<255>;
typedef  string  register<255>;
typedef  string  class<255>;
typedef  string  pattern<255>;

typedef  opaque  capability[8];
typedef  opaque  cookie[4];
typedef  opaque  registervalue<1024>;
typedef  opaque  registerattribute<1024>;
```

```
typedef   union switch (boolean valid)   {
        TRUE:
        struct {
          register   reg;
          registervalue   rval;
          cookie   kk;
          regvallist   nextentry;
        }
        FALSE:
        struct {}
} regvallist;
typedef   union switch (boolean valid)   {
        TRUE:
        struct {
          register   reg;
          registerattribute   rattrib;
          cookie   kk;
          regattriblist   nextentry;
        }
        FALSE:
        struct {}
} regattriblist;
typedef   union switch (boolean valid)   {
        TRUE:
        struct {
          register   reg;
          cookie   kk;
          reglist   nextentry;
        }
        FALSE:
        struct {}
} reglist;
```

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.

XDR Specification

```
procedure   Null ()            returns ()
```

Procedure 1. Claim Resource

REQUEST

Allocates the specified resource for exclusive use of the client. The available resources will be dependant on the specific server and its supported applications.

resource name length
resource name

RESPONSE - success

The response returns a 8 byte (64 bit) *capability* which is an opaque field generated by the server and used by the client as an access token in subsequent procedure requests for the allocated resource.

success = 0
capability

RESPONSE - failure

Procedure status is a 4 byte field which is non zero and indicates the cause of the failure.

Procedure status=	1	resource not known
	2	resource already in use

procedure status

XDR Specification

```
procedure   ClaimResource (res)                    returns (reply)
            resource  res;
            union switch (stat status)  {
               OK:
               capability  cap;
               default:
               struct  {}
            }  reply;
```

Procedure 2. Free Resource

The specified resource is released by the client. The server will cancel the capability and no further access to this resource will be permitted until after a successful Claim Resource request.

capability
resource name length
resource name

RESPONSE

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

procedure status

Procedure status=	0	Ok
	1	resource not known
	3	capability invalid

XDR Specification

```
procedure   FreeResource (cap,res)    returns (status)
            capability  cap;
            resource   res;
            stat   status;
```

Procedure 3. Define Register

If the resource specified is known by the server and the access capability is valid defines a new register name which is to be associated with the object known by the specified class.

capability
resource name length
resource name
register name length
register name
class name length
class name

RESPONSE

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

procedure status

Procedure status=	0	Ok
	1	resource not known
	3	capability invalid
	6	register already defined
	7	unknown class
	11	illegal register name
	12	out of memory

XDR Specification

```
procedure   DefineRegister (cap, res, reg, p)  returns (status)
            capability  cap;
            resource  res;
            register  reg;
            class  c;
            stat  status;
```

Procedure 4. Undefine Register

If the register specified is known by the server for the resource specified and the access capability is valid the register is removed from those known by the server for the specified resource.

capability
resource name length
resource name
register name length
register name

RESPONSE

procedure status

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

Procedure status=	0	Ok
	1	resource not known
	3	capability invalid
	4	register not known

XDR Specification

```
procedure   UndefineRegister (cap, res, reg)   returns (status)
            capability  cap;
            resource  res;
            register  reg;
            stat  status;
```

Procedure 5. Read Register

capability
resource name length
resource name
register name length
register name

If the register specified is known by the server for the resource specified and the access capability is valid the read procedure defined by the object associated with the register is envoked and the data obtained is returned.

RESPONSE - success

success = 0
register value length
register value

Register length is a 4 byte field which defines the length of the following register data in bytes.

Register value contains the data returned by the read procedure. Data in this field conforms to the XDR standard for the particular data type.

RESPONSE - failure

Procedure status is a 4 byte field which is non zero and indicates the cause of the failure.

procedure status

Procedure status=	1	resource not known
	3	capability invalid
	4	register not known
	8	application not responding
	10	application detected failure

XDR Specification

```
procedure   ReadRegister (cap, res, reg)          returns (reply)
            capability  cap;
            resource  res;
            register  reg;
            union switch (stat status)  {
               OK:
               registervalue  v;
               default:
               struct  {}
            }  reply;
```

Procedure 6. Write Register

If the register specified is known by the server for the resource specified and the access capability is valid the write procedure defined by the object associated with the register is envoked with the data supplied.

capability
resource name length
resource name
register name length
register name
register value length
register value

RESPONSE

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

procedure status

Procedure status= 0 OK

0	OK
1	resource not known
3	capability invalid
4	register not known
8	application not responding
10	application detected failure

XDR Specification

```
procedure   WriteRegister (cap, res, reg, v)   returns (status)
            capability  cap;
            resource  res;
            register  reg;
            registervalue  v;
            stat  status;
```

Procedure 7. Initialise Register

If the register specified is known by the server for the resource specified and the access capability is valid the initialise procedure defined by the object associated with the register is envoked.

capability
resource name length
resource name
register name length
register name

RESPONSE

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

procedure status

Procedure status=	0	OK
	1	resource not known
	3	capability invalid
	4	register not known
	8	application not responding
	10	application detected failure

XDR Specification

```
procedure   InitialiseRegister (cap, res, reg)   returns (status)
            capability  cap;
            resource   res;
            register   reg;
            stat   status;
```

Procedure 8. Inquire Registers

If the resource specified is known by the server and the access capability is valid those register names known by the server for the resource are returned with a total size of up to count bytes. The pattern string supplied is used to select the set of registers to be returned from the total available.

capability
resource name length
resource name
pattern length
pattern
cookie
count

RESPONSE - success

Each entry contains the name of the register and a cookie which is an opaque pointer to the next entry. The cookie is used in the next Inquire Registers call to get more entries starting at a given point. The special cookie zero (all bits zero) can be used to get the entries starting at the beginning. The eod flag has a value of TRUE if there are no more entries; valid is used to mark the end of the entries.

success = 0
valid = 1
register name length
register name
cookie
:
valid = 0
eod

RESPONSE - failure

Procedure status is a 4 byte field which is non zero and indicates the cause of the failure.

procedure status

Procedure status=	1	resource not known
	3	capability invalid
	9	invalid cookie
	13	count too small

XDR Specification

```
procedure   InquireRegisters (cap, res, patt, k, n)      returns (reply)
            capability  cap;
            resource  res;
            pattern  patt;
            cookie  k;
            unsigned  n;
            union switch (stat status)  {
               OK:
               reglist  regs;
               boolean  eod;
               default:
               struct  {}
            }  reply;
```

Procedure 9. Read Registers

If the resource specified is known to the server and the access capability is valid the read procedure defined by the object associated with each register selected from the total available using the supplied pattern is envoked.

capability
resource name length
resource name
pattern length
pattern
cookie
count

RESPONSE - success

Each entry contains the name of the register, the data returned by the read procedure and a cookie which is an opaque pointer to the next entry. The cookie is used in the next Read Registers call to get more entries starting at a given point. The special cookie zero (all bits zero) can be used to get the entries starting at the beginning. The eod flag has a value of TRUE if there are no more entries; valid is used to mark the end of the entries.

success $= 0$
valid $= 1$
register name length
register name
register value length
register value
cookie
:
valid $= 0$
eod

RESPONSE - failure

procedure status

Procedure status is a 4 byte field which is non zero and indicates the cause of the failure.

Procedure status=	1	resource not known
	3	capability invalid
	8	application not responding
	9	invalid cookie
	10	application detected failure
	13	count too small

XDR Specification

```
procedure   ReadRegisters (cap, res, patt, k, n)        returns (reply)
            capability  cap;
            resource  res;
            pattern  patt;
            cookie  k;
            unsigned  n;
            union switch (stat status)  {
               OK:
               regvallist  regvals;
               boolean  eod;
               default:
               struct  {}
            }  reply;
```

Procedure 10. Write Registers

If the resource specified is known to the server and the access capability is valid the write procedure defined by the object associated with each register selected from the total available using the supplied pattern is envoked with the data supplied.

capability
resource name length
resource name
pattern length
pattern
register value length
register value

RESPONSE

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

procedure status

Procedure status=	0	Ok
	1	resource not known
	3	capability invalid
	8	application not responding
	10	application detected failure

XDR Specification

```
procedure   WriteRegisters (cap, res, patt, v)     returns (status)
            capability  cap;
            resource  res;
            pattern  patt;
            registervalue  v;
            stat  status;
```

Procedure 11. Initialise Registers

If the resource specified is known to the server and the access capability is valid the initialise procedure defined by the object associated with each register selected from the total available using the supplied pattern is envoked.

capability
resource name length
resource name
pattern length
pattern

RESPONSE

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

procedure status

Procedure status=	0	Ok
	1	resource not known
	3	capability invalid
	8	application not responding
	10	application detected failure

XDR Specification

```
procedure   InitialiseRegisters (cap, res, patt)     returns (status)
            capability  cap;
            resource  res;
            pattern  patt;
            stat  status;
```

Procedure 12. Read Attribute

If the register specified is known by the server for the resource specified and the access capability is valid the attribute string associated with the register is returned.

capability
resource name length
resource name
register name length
register name

RESPONSE - success

Attribute length is a 4 byte field which defines the length of the following attribute data in bytes.

success $= 0$
attribute value length
attribute value

Attribute value contains the data component of the attribute. Data in this field conforms to the XDR standard for the particular data type.

RESPONSE - failure

Procedure status is a 4 byte field which is non zero and indicates the cause of the failure.

procedure status

Procedure status=	1	resource not known
	3	capability invalid
	4	register not known

XDR Specification

```
procedure   ReadAttribute (cap, res, reg)        returns (reply)
            capability  cap;
            resource  res;
            register  reg;
            union switch (stat status)  {
               OK:
               registerattribute  v;
               default:
               struct  {}
            } reply;
```

Procedure 13. Write Attribute

If the register specified is known by the server for the resource specified and the access capability is valid the attribute string associated with the register is defined using the data supplied.

capability
resource name length
resource name
register name length
register name
attribute value length
attribute value

RESPONSE

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

procedure status

Procedure status=	0	OK
	1	resource not known
	3	capability invalid
	4	register not known

XDR Specification

```
procedure   WriteAttribute (cap, res, reg, v)        returns (status)
            capability  cap;
            resource  res;
            register  reg;
            attributevalue  v;
            stat  status;
```

Procedure 14. Undefine Registers

If the resource specified is known to the server and the access capability is valid those registers selected from the total available using the supplied pattern are removed from those known by the server for the specified resource.

capability
resource name length
resource name
pattern length
pattern

RESPONSE

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

procedure status

Procedure status=	0	Ok
	1	resource not known
	3	capability invalid

XDR Specification

```
procedure   UndefineRegisters (cap, res, patt)      returns (status)
            capability  cap;
            resource  res;
            pattern  patt;
            stat  status;
```

Procedure 15. Read Attributes

If the resource specified is known by the server and the access capability is valid those register names and the associated attribute string known by the server for the resource are returned with a total size of up to count bytes. The pattern string supplied is used to select the set of registers to be returned from the total available.

capability
resource name length
resource name
pattern length
pattern
cookie
count

RESPONSE - success

Each entry contains the name of the register, its associated attribute string and a cookie which is an opaque pointer to the next entry. The cookie is used in the next Read Attributes call to get more entries starting at a given point. The special cookie zero (all bits zero) can be used to get the entries starting at the beginning. The eod flag has a value of TRUE if there are no more entries; valid is used to mark the end of the entries.

success = 0
valid = *1*
register name length
register name
attribute value length
attribute value
cookie
:
valid = *0*
eod

RESPONSE - failure

Procedure status is a 4 byte field which is non zero and indicates the cause of the failure.

procedure status

Procedure status=	1	resource not known
	3	capability invalid
	13	count too small

XDR Specification

```
procedure   ReadAttributes (cap, res, patt, k, n)      returns (reply)
            capability  cap;
            resource  res;
            pattern  patt;
            cookie  k;
            unsigned  n;
            union switch (stat status)  {
               OK:
               regattriblist  regattribs;
               boolean  eod;
               default:
               struct  {}
            } reply;
```

Procedure 16. Write Attributes

If the resource specified is known to the server and the access capability is valid the attribute string associated with each register selected from the total available using the supplied pattern is defined using the data supplied.

capability
resource name length
resource name
pattern length
pattern
attribute value length
attribute value

RESPONSE

procedure status

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

Procedure status=	0	Ok
	1	resource not known
	3	capability invalid

XDR Specification

```
procedure   WriteAttributes (cap, res, patt, v)    returns (status)
            capability  cap;
            resource  res;
            pattern  patt;
            registervalue  v;
            stat  status;
```