EDOC078

EUROGAM PROJECT

NSF DATA ACQUISITION SYSTEM

Spectrum Access Server Protocol

Edition 1.2
Aug 1992

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

Introduction

This document describes the Spectrum Access Server (SAS) protocol specification used to provide access to the distributed spectrum servers which form part of the NSF data acquisition system.

The protocol is used to implement the Spectrum Access procedures defines in *EDOC062* together with the spectrum format defined in *EDOC061*.

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

Version 1.0 June 1991 - Initial version

Version 1.1 Sept 1991

Version 1.2 Aug 1992 - Allow authorisation with no password if client machine is listed in user's .rhosts.sas, /etc/hosts.sas or /etc/hosts.equiv.

Use

Throughout this document *capability* is an 8 byte (64 bit) opaque field generated by the server and used by the client as an access token.

Throughout this document the spectrum being accessed is identified by the string variable *pathname*. For on-line spectrum servers this is the name of the spectrum within the server. For off-line spectrum servers (ie disc-based spectra) this is either an absolute pathname identifying a disc file known to the spectrum server or is a pathname which is relative to the 'home' directory of the authenicated user issuing the request.

Program Specification

RPC Information

The SAS service uses only the Null authentication except for procedure 1 which can use Unix authentication protocol requests.

SAS is supported on the UDP/IP transport only

SAS is RPC Program 28000230

This is Version 1 of the protocol

SAS uses UDP port number 10230

XDR Information

Basic Data Types

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

typedef  enum {
        OK = 0
        authentication_failure = 1
        already_in_use = 2
        capability_invalid = 3
        pathname_invalid = 4
        parameter_invalid = 5
        not_a_spectrum = 6
        not_a_directory = 7
        data_access_error = 8
} stat;

typedef  string  user<32>;
typedef  string  password<32>;
typedef  string  pathname<255>;

typedef  string  timedate[20];

typedef  unsigned  map;

typedef  opaque  capability[8];

typedef  signed  base[8];
typedef  unsigned  range[8];
typedef  unsigned  size[8];

typedef  unsigned  dimension;
typedef  unsigned  arraytype;
typedef  unsigned  arraylayout;
typedef  opaque  countvalue;
typedef  unsigned  stringtype;
typedef  unsigned  stringid;
```

Actually *countvalue* is not opaque, the data type is determined by *type*.

```
typedef struct {
        arraylayout layout;
        arraytype type;
} attrib;
```

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. Authorise

Requests authorisation for the client to access the spectrum server. For off-line spectrum servers (ie for disc-based spectra) a User Identifier must be supplied. If the procedure is called with Null authentication an appropriate password must also be supplied to authenicate the user. If the procedure is called with Unix authentication and the request comes from a machine listed in one of the files .rhosts.sas in the user's home directory, /etc/hosts.equiv or /etc/hosts.sas, all on the server machine, a password need not be supplied. The User Identifier is then used to define a 'home' disc directory. Pseudo users may be used for access to shared spectrum databases. For on-line spectrum servers the User Identifier and password may be null length strings.

REQUEST

User name and *password* are strings which provide user authenication. *Mode* indicates whether read only access (mode=0) or read/write access (mode=1) is required.

user name length
user name
password length
password
mode

RESPONSE - success

If the authorisation request is successful *capability* is returned as an access token to be used in all subsequent procedure requests.

success = 0
capability

RESPONSE - failure

procedure status

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

Procedure status=	1	user authentication error
	2	server already in use

XDR Specification

```
procedure   Authorise(id,pass,mode)   returns (reply)
            user   id ;
            password   pass ;
            unsigned   mode ;

            union switch (stat status)   {
               OK:
               capability  cap;
               default:
               struct   {}
            }  reply;
```

Procedure 2.   Create Spectrum

REQUEST

If the access *capability* is valid creates a spectrum in the server with name given by *pathname*.
The remaining fields are as defined in the document *EDOC061*. Undefined fields should be set to -1 or all one bits.

capability
pathname length
pathname
dimension
dimension 1 base
dimension 2 base
:
dimension 8 base
dimension 1 range
dimension 2 range
:
dimension 8 range
data array 1 layout
data array 1 type
data array 2 layout
data array 2 type

The array layout and array type parameters may be defined here or left undefined (set to -1 or all one bits) and defined latter using the Change Array Options procedure. If *data array 1 layout* and *data array 1 type* are defined here for an on-line spectrum server then space will be allocated for the data array at this point. In the case of disc based spectra space for the data arrays will not be allocated until the first Write Spectrum procedure.

RESPONSE

procedure status

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

Procedure status=	0	success
	3	invalid capability
	4	invalid pathname
	5	invalid parameter value
	7	pathname not a directory

XDR Specification

```
procedure   CreateSpectrum(cap,path,dim,bse,rng,layout1,type1,layout2,type2)
                   returns (status)
            capability  cap;
            pathname  path;
            dimension  dim;
            base  bse;
            range  rng;
            arraylayout  layout1;
            arraytype  type1;
            arraylayout  layout2;
            arraytype  type2;
            stat  status;
```

Procedure 3. Delete Spectrum

REQUEST

capability
pathname length
pathname

If the access *capability* is valid deletes the spectrum in the server with name given by *pathname*.

RESPONSE

procedure status

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

Procedure status=	0	success
	3	invalid capability
	4	invalid pathname
	6	pathname not a spectrum

XDR Specification

```
procedure   DeleteSpectrum(cap,path)   returns (status)
            capability  cap;
            pathname  path;
            stat  status;
```

Procedure 4. Look Up Spectrum

REQUEST

If the access *capability* is valid and the spectrum defined by *pathname* exists attribute information extracted from the spectrum header is returned.

capability
pathname length
pathname

RESPONSE - success

The fields returned are as defined in the document *EDOC061*. Undefined fields are returned set to -1 or all one bits.
The field *data array 1 base address* will only be defined for on-line spectra.

success = 0
dimension
creation time and date
modification time and date
dimension 1 base
:
dimension 8 base
dimension 1 range
:
dimension 8 range
information string map
annotation string map
calibration string map
efficiency string map
data array 1 layout
data array 1 type
data array 2 layout
data array 2 type
data array 1 base address

The string maps provide a convenient indication of the existance or not of the various data strings. If string n exists then bit n (with the most significant bit being 0 and the least being 31) is set to 1 otherwise it is set to 0. For example if information strings 1,2 and 3 are defined then the information string map will have the value @70000000. This avoids the need for a *read string* procedure to be used for non-existant strings.

RESPONSE - failure

procedure status

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

Procedure status=	3	invalid capability
	4	invalid pathname
	6	pathname not a spectrum

XDR Specification

```
procedure   LookUpSpectrum(cap,path)   returns (reply)
            capability  cap ;
            pathname  path ;

            union switch (stat status)  {
               OK:
               dimension  dim;
               timedate  creation;
               timedate  modification;
               base  bse;
               range  rng;
               map  information;
               map  annotation;
               map  calibration;
               map  efficiency;
               attrib  array1;
               attrib  array2;
               unsigned  address;
               default:
               struct  {}
            }  reply;
```

Procedure 5. Read String

REQUEST

If the access *capability* is valid and the spectrum defined by *pathname* exists then the string of type *string type* identified by *string number* is returned.

capability
pathname length
pathname
string type
string number

string type	=	0	information
		1	annotation
		2	calibration
		3	efficiency

String number is in the range 1 to 16 for information strings and 1 to the spectrum dimension for others.

RESPONSE - success

The requested string is returned. If the requested string is not defined (but the *type* and *number* are within valid range) then a null length string is returned.

success = 0
string length
string

RESPONSE - failure

procedure status

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

Procedure status=	3	invalid capability
	4	invalid pathname
	5	invalid parameter value
	6	pathname not a spectrum

XDR Specification

```
procedure   ReadString(cap,path,stype,snumber)   returns (reply)
            capability  cap ;
            pathname  path ;
            stringtype  stype ;
            stringid  snumber ;

            union switch (stat status)  {
               OK:
               string  data <>
               default:
               struct  {}
            } reply;
```

Procedure 6. Write String

REQUEST

If the access *capability* is valid and the spectrum defined by *pathname* exists then the string of type *string type* identified by *string number* is defined with the contents of the string parameter *string*.

capability
pathname length
pathname
string type
string number
string length
string

Any existing string is overwritten and an existing string may be undefined by use of a *write string* procedure request with a null length string.

String type and *string number* are as defined for the *read string* procedure.

RESPONSE

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

procedure status

Procedure status=	0	success
	3	invalid capability
	4	invalid pathname
	5	invalid parameter value
	6	pathname not a spectrum

XDR Specification

```
procedure   WriteString(cap,path,stype,snumber,data)   returns (status)
            capability  cap ;
            pathname  path ;
            stringtype  stype ;
            stringid  snumber ;
            string  data <>;
            stat  status ;
```

Procedure 8. Read Names

REQUEST

If the access *capability* is valid those names identified to the server by *pathname* are returned with a total size of up to *count* bytes.

capability
pathname length
pathname
cookie
count

For on-line spectrum servers which have only a single level name structure *pathname* should be a null length string and all names known to the server will be returned. For off-line spectrum servers (ie disc-based spectra) *pathname* must define a directory file and the names of the spectrum files and directory files within that directory will be returned. All other files will be ignored.

RESPONSE - success

Each entry contains the name of the spectrum or file, the type of file and a cookie which is an opaque pointer to the next entry. The cookie can be used in the next *read name* 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. *Eod* has a value of TRUE if there are no more entries and FALSE if there are more. *Valid* is used to mark the end of the entries.

success = 0
valid=true
name length
name
entry type
cookie
:
valid=false
eod

entry type	=	1	spectrum
		2	directory file

RESPONSE - failure

procedure status

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

Procedure status=	3	invalid capability
	4	invalid pathname
	5	invalid parameter value
	7	pathname not a directory

XDR Specification

```
procedure   ReadNames (cap,path,k,n)   returns (reply)
            capability  cap;
            pathname  path;
            cookie  k;
            unsigned  n;

            union switch (stat status)  {
               OK:
               typedef  union switch (boolean valid)  {
                  TRUE:
                  struct {
                     string  name <255>;
                     unsigned  entrytype;
                     cookie  kk;
                     entry  nextentry;
                  }
                  FALSE:
                  struct {}
               } entry;
               boolean  eod;
               default:
               struct  {}
            }  reply;
```

Procedure 9. Read Spectrum

REQUEST

If the access *capability* is valid and the spectrum defined by *pathname* exists then returns *count* items of data from the spectrum.
The fields are as defined in the document *EDOC061*. Undefined fields should be set to -1 or all one bits.
The elements of the arrays *base*, *range*, and *size* defined must be consistent with the dimension of the spectrum.
If *size* are undefined then it is assumed that no scaling of the data is required and the spectrum is returned as it would be if *size* were equal to *range*.
Array number = 1 to access the spectrum and = 2 to access the error spectrum.

capability
pathname length
pathname
array number
offset
count
base1
base2
:
base8
range1
range2
:
range8
size1
size2
:
size8
arraytype

The arrays *base* and *range* define a region of counts within the spectrum to be returned and the counts may then be considered to form a sequence of data items of size given by the type of the spectrum array. Counts are returned starting at *offset* data items into the defined region. The first data item is at *offset* zero.

The data returned is scaled such that the region defined by *base* and *range* is returned in an array defined by *size* and *arraytype*.

RESPONSE - success

The data is returned in the array *data*. *Flag* will normally be set to zero but will be set to -1 (all one bits) if truncation/overflow occurred while scaling the data in order to return it within the array requested and/or in the type requested.

success = 0
flag
data length
data

Requests that a region of the spectrum array be returned with the option that the size of the array elements and the format of each data item may be different from the source spectrum. Normally this will be such that the amount of data to be returned will be less than the original data.

RESPONSE - failure

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

procedure status

Procedure status=	3	invalid capability
	4	invalid pathname
	5	invalid parameter value
	6	pathname not a spectrum

XDR Specification

```
procedure   ReadSpectrum(cap,path,number,offset,count,bse,rng,s,type)   returns (reply)
            capability  cap ;
            pathname   path ;
            unsigned   number;
            unsigned   offset;
            unsigned   count;
            base   bse ;
            range   rng ;
            size   s ;
            arraytype   type ;

            union switch (stat status)   {
               OK:
               unsigned   flag;
               string   data <>;
               default:
               struct   {}
            }  reply;
```

Procedure 10. Write Spectrum

REQUEST

If the access *capability* is valid and the spectrum defined by *pathname* exists then the supplied data is written to the spectrum.
The fields are as defined in the document *EDOC061*. Undefined fields should be set to -1 or all one bits.
The elements of the arrays *base* and *range* defined must be consistent with the dimension of the spectrum.
Array number = 1 to access the spectrum and = 2 to access the error spectrum.
The data to be written is supplied in the array *data* and *arraytype* defines the format of each data item in this array.

capability
pathname length
pathname
array number
offset
count
base1
base2
:
base8
range1
range2
:
range8
arraytype
data length
data

The arrays *base* and *range* define a region of counts within the spectrum to be written and the counts may then be considered to form a sequence of data items of size determined by *arraytype*. *count* counts are written starting at *offset* data items into the defined region. The first data item in a region is at *offset* zero. Type conversion takes place if *arraytype* differs from the type of the spectrum array being written.

Requests that a region of the spectrum array be written with the option that the format of each data item supplied may be different from that of the target spectrum.

RESPONSE

procedure status
flag

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

If *procedure status* = 0 then *flag* is also returned which will normally be set to zero but will be set to -1 (all one bits) if truncation/overflow occurred while converting the data between the type supplied and the type defined for the data array.

Procedure status=	0	success
	3	invalid capability
	4	invalid pathname
	5	invalid parameter value
	6	pathname not a spectrum

XDR Specification

```
procedure   WriteSpectrum(cap,path,number,offset,count,bse,rng,type,data)   returns (reply)
            capability  cap ;
            pathname  path ;
            unsigned  number;
            unsigned  offset;
            unsigned  count;
            base  bse;
            range  rng;
            arraytype  type ;
            string  data <>;

            union switch (stat status)  {
               OK:
               unsigned  flag;
               default:
               struct  {}
            }  reply;
```

Procedure 11. Change Array Options

REQUEST

If the access *capability* is valid and the spectrum defined by *pathname* exists then the array layout and array type for the data array defined by *array number* may be set.

capability
pathname length
pathname
array number
array layout
array type

If a data array already exists then it will be destroyed by this operation. By setting the array layout and array type to undefined (-1 or all one bits) any existing data array may be just destroyed.

If *array layout* and *array type* are defined here for an on-line spectrum server then space will be allocated for the data array at this point. In the case of disc based spectra space for the data arrays will not be allocated until the first Write Spectrum procedure.

RESPONSE

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

procedure status

Procedure status=	0	success
	3	invalid capability
	4	invalid pathname
	5	invalid parameter value
	6	pathname not a spectrum

XDR Specification

```
procedure   ChangeArrayOptions(cap,path,number,layout,type) returns (status)
            capability   cap;
            pathname   path;
            unsigned   number;
            arraylayout   layout;
            arraytype   type;
            stat   status;
```

Procedure 12. UnAuthorise

REQUEST

Reverse the action of a previous Authorise procedure call. The capability will no longer be accepted.

capability

RESPONSE

success=0

XDR Specification

```
procedure   UnAuthorise(cap) returns (status)
            capability  cap;
            stat        status;
```

Procedure 13. Generate Error Report

REQUEST

An explanation of the last error condition associated with this capability is returned as a text string.

capability

RESPONSE

success=0
text length
text

XDR Specification

```
procedure   GenerateErrorReport(cap) returns (status,text)
            capability  cap;
            stat        status;
            string      text;
```