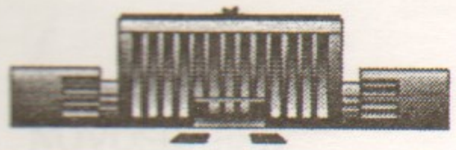
Siberian Branch of Russian Academy of Science
BUDKER INSTITUTE OF NUCLEAR PHYSICS

I.A. Gaponenko, A.A. Salnikov

INFORMATION MANAGEMENT SYSTEM FOR SND EXPERIMENT

Budker INP 98-39

http://www.inp.nsk.su/publications



Novosibirsk
1998

Information management system for SND experiment

I.A. Gaponenko and A.A. Salnikov

Budker Institute of Nuclear Physics SB RAS
630090 Novosibirsk, Russia

Abstract

This preprint describes the information management system designed for the on-line system of SND experiment. It was built with the client-server model, optimized for the access time, and comprises a vital part of the DAQ. We present here basic design issues, programming interface together with some examples. The system was used successfully in the DAQ of the SND and CMD-2 experiments.

1 Introduction

The idea of IMAN[1] appeared during the development of the on-line system of the SND experiment[1]. The processes comprising the on-line system needed to exchange the information at the rate different from the event processing rate, this information is basically an "environment" of the data taking and processing (beam conditions, rates, luminosity, etc.) Access to this information should be extremely fast, ideally without any impact on the data taking. Starting with these requirements the information system called IMAN was designed and implemented. Below we describe the system in details, including its architecture, short description of the client-side API, and also give some primitive examples of the code using this library.

2 Architecture overview

IMAN is built using the client-server model and therefore consists of two parts – server and client library. The server is a stand-alone process running under the control of VMS operating system, serving the requests from other processes (clients). The communication part of IMAN supports two transports between the server and client processes – VMS mailboxes and socket library (TCP/IP). This provides the maximum access speed for the clients running on the same host as the server process, and also supports clients running at the remote hosts. See figure 1 for the sketch of the interconnections between different processes.

[1]The name IMAN can be though as the abbreviation of "information management".

Information in the server is stored in the memory segment which is mapped to the file on disk (see figure 2.) This memory segment holds all information in the position-independent form, allowing mapping of the file content at any address in the server's address space. The control over the memory in this segment is performed with the original position-independent memory allocation scheme. Paging mechanism of VMS provides low overhead for saving information together with high reliability. Thus server is able to access all information very fast without the complicated file access methods.

The backup scheme for the data was designed and implemented using the same paging mechanism and memory allocation scheme, and is also extremely fast. It produces the exact copies of the data, which can be reused in the case of the program or computer crashes. Backups are made periodically, at the moments when the system's load is low, and practically do not disturb the normal operation of the whole system.

Information in server is stored in the native VAX/VMS formats for the integer and floating point numbers. When the data are sent across the network the integer numbers are converted to the "network byte ordering", client part of the library converts these data in the client's native format. Floating point numbers are sent in the VAX/VMS representation, and also converted into the client's native format by the special architecture-dependent functions in the client library.

To simplify data exchange algorithm and further increase the exchange speed, the amount of data passing between the client and server is limited to 2048 bytes. This limits the amount of the data which the items may contain. The sum of the domain name length, item name length and the item data should be less than 2040 bytes (some bytes are spared to the control information.)

3 Information inside IMAN

IMAN allows storage and further retrieval of small pieces of information. Each such piece is referred later in this paper as "item" and has some essential characteristics – name, type and length. Items are grouped together in domains – named collections of items – which have only one characteristic, the name. Each item belongs exactly to one domain, so every item has unique path consisting of domain name and item name (see figure 3.) For example, domain named "STATUS" can have items "MAGNETIC FIELD", "CURRENTS", "LUMINOSITY", etc.

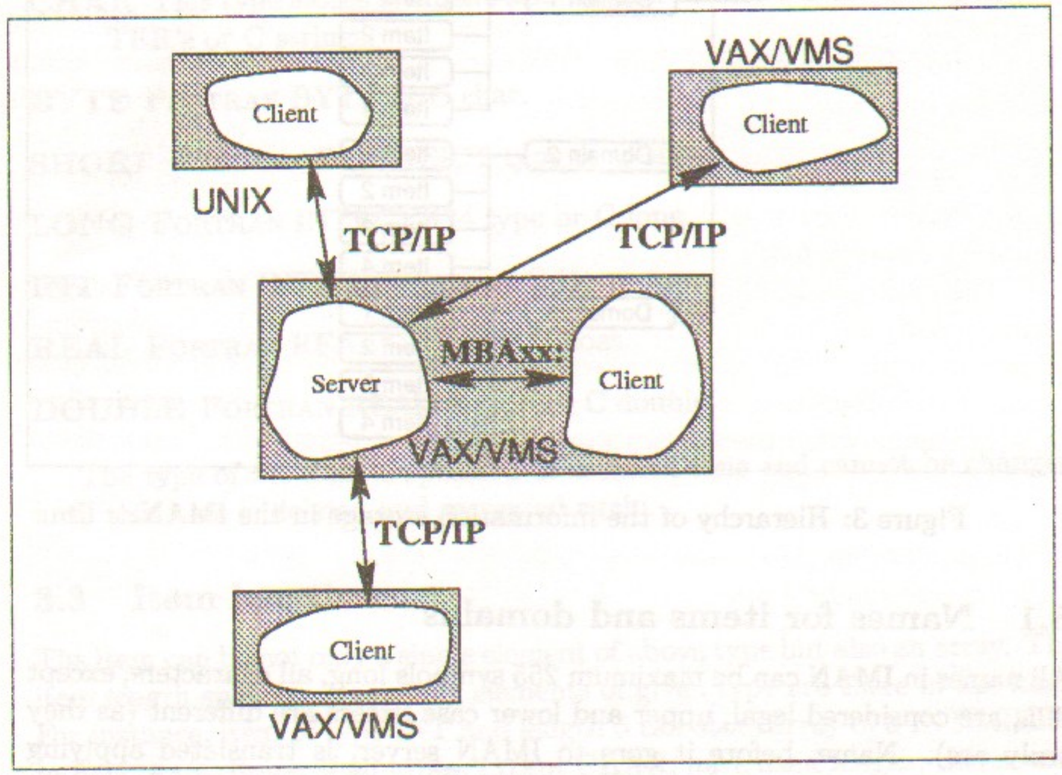


Figure 1: Sketch of the client-server model and the connection using two different transports.

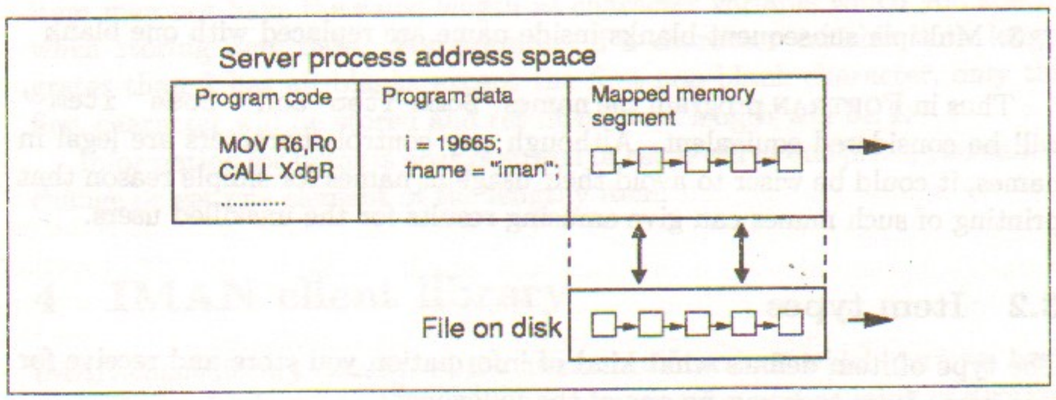


Figure 2: Mapping of the server address space onto the disk file using VAX/VMS paging scheme.

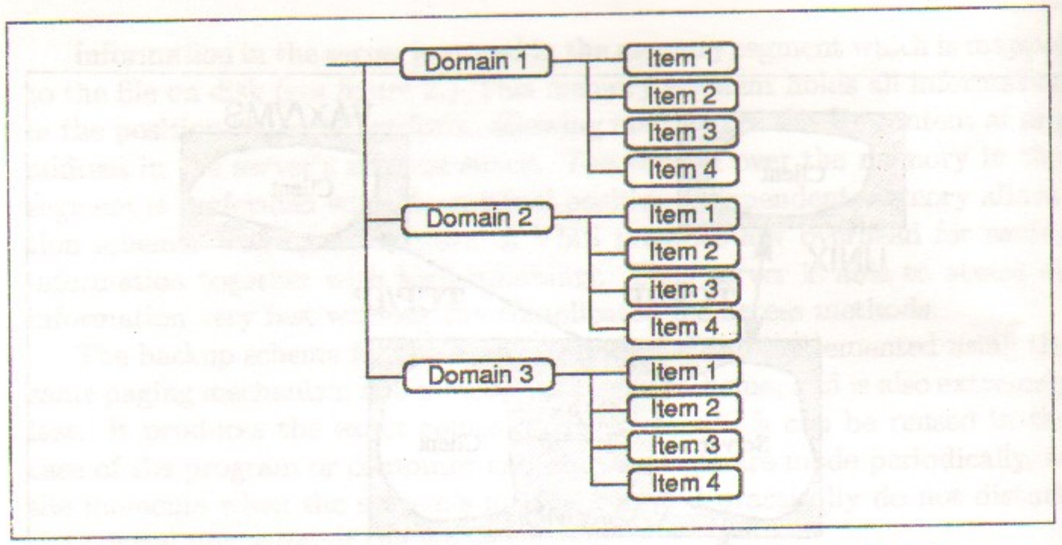


Figure 3: Hierarchy of the information storage in the IMAN.

3.1 Names for items and domains

All names in IMAN can be maximum 255 symbols long, all characters, except NUL, are considered legal, upper and lower case letters are different (as they truly are). Name, before it goes to IMAN server, is translated applying certain rules:

1. All TABs and newline characters are substituted with blanks
2. Leading and trailing blanks are removed
3. Multiple subsequent blanks inside name are replaced with one blank

Thus in FORTRAN program the names `'Some item'` and `' Some  item '` will be considered equivalent. Although any control characters are legal in names, it could be wiser to avoid their usage in names for simple reason that printing of such names can give amusing results for the unskilled users.

3.2 Item types

The type of item defines what kind of information you store and receive for this item. Item type can be one of the following[2]:

[2]The types were defined for the present-day 32-bits computers and, generally speaking, are not portable.

CHAR This type stores symbolic information such as FORTRAN CHARACTER's or C strings.

BYTE FORTRAN BYTE or C char.

SHORT FORTRAN INTEGER*2 type or C short.

LONG FORTRAN INTEGER*4 type or C long.

INT FORTRAN INTEGER type or C int.

REAL FORTRAN REAL*4 type or C float.

DOUBLE FORTRAN REAL*8 type or C double.

The type of the item is specified at creation time and cannot be changed until this item is deleted and recreated again.

3.3 Item length

The item can be not only a single element of above type but also an array. The item length specifies how many elements of given type are there in the item. For instance, item of type INT and length 5 can store array of 5 INTEGER's in FORTRAN. Items of all types, except CHAR, have fixed length, this means that length of the item is specified at creation time and cannot be changed later. Items of type CHAR have floating length, this means that the length of the item changes with data stored in it. When storing data for the items of type CHAR, trailing blanks are stripped. Thus length of the CHAR type item may not have the same length as character variable which you specify when storing item data. For example, if a character variable with length grater than 1 has all blanks except the first non-blank character, only this first character will be stored and resulting item length will be 1.

Information for items is always stored or retrieved in one piece, you cannot change or get one element of the lengthy item.

4 IMAN client library

IMAN client library consists of the number of functions which perform basic operations with items and domains, such as creation and deleting of domains and items, storing and retrieving item data and querying item type and length. Following sections describe this functions in details.

Functions in client library can be divided in two groups:

1. For "everyday" use. These functions operate with existing items and domains and perform such operations as changing item data, retrieving item data, changing current domain. These functions are:

IMAN_PutItem
Change item data. Changes data stored with specified item.

IMAN_GetItem
Get item data. Retrieves data stored with specified item.

IMAN_GetItemType
Get item type and length. Returns type and length for specified item.

IMAN_SetDomain (VAX/VMS only)
Set current domain. All operations on items are performed for domain set with this function. If you did not set domain any operation on item will result in IMAN_SETDOM error code (see Appendix A.)

2. Rarely used functions. These functions perform management operations such as creation and deletion of domains and items, receiving list of existing items and domains. These functions are as well available for users but they are also gathered in simple management task IMAM, which is built on top of the client library and has well-defined user interface and built-in help. See Appendix 5. These functions are:

IMAN_CreateDomain
Create domain. Creates one empty domain with specified name.

IMAN_CreateItem
Create item. Creates item in current domain and specifies its name, type, length and data it contains.

IMAN_DeleteDomain
Delete domain. Deletes specified domain and all items in it.

IMAN_DeleteItem
Delete item. Deletes specified item in current domain.

4.1 Client library on VAX/VMS.

4.1.1 Linking.

To gain access to the client functions you should link to the client library and, optionally, to the file with messages for the VAX/VMS error handling facility. Client library is a shareable image and resides in directory pointed by logical name `IMAN`. So to link against IMAN library you should create option file for the linker and place reference to shareable image and message file like in example below.

```
$ EDIT LINK.OPT
IMAN:IMAN_MSGS.OBJ    ! IMAN message file, optional
IMAN:IMAN_SHARE/SHARE ! IMAN shareable library
^Z
* EXIT
$ LINK SOME_OTHER_STUFF,LINK.OPT/OPT
```

You can also use existing option file with contents just as in example above, which resides in IMAN directory. In this case you can type

```
$ LINK SOME_OTHER_STUFF, IMAN:IMAN.OPT/OPT
```

4.1.2 Checking for errors.

Most functions in IMAN library return condition values which conform to VMS programming standards and specifies whether this function finished successfully or not. In latter case you can also retrieve information about error occurred in function call. Condition values returned by the library functions are defined in the files `IMAN:IMANDEF.FOR` for FORTRAN programs and `IMAN:IMANDEF.H` for C/C++ programs. All condition values returned by the library functions are described in details in Appendix A.

You can also retrieve the message describing error in standard VMS way with functions `lib$signal()` or `sys$getmsg()`, this requires including IMAN message file `IMAN:IMAN_MSGS.OBJ` during linking procedure. I strongly recommend for programmers to test the condition value returned after each call like in example below:

```
integer *4 stat
integer *4 IMAN_GetItem

stat = IMAN_GetItem ( 'Some item', data )
if ( .not. stat ) call lib$signal ( %val(stat) )
```

4.1.3 IMAN functions.

In this section all functions of the IMAN client library are described in alphabetic order. For each function its calling sequence in VMS format and FORTRAN format, types of parameters and possible return codes are presented.

IMAN_BeginBatching

Sets batching mode of communication.

VMS format:

```
IMAN_BeginBatching
```

Fortran interface:

```
CALL IMAN_BeginBatching
```

Description:

In default operating mode the client closes connection to server after each call to client routine. This produces some overhead for opening connection again before next call. It simplifies programming but can be unwise when an application performs many operations with the database. This routine changes behavior of the communication part so that it will not close connection before return to the user and client remains connected to the server until explicitly detached with the routine IMAN_EndBatching.

IMAN_CreateDomain

Creates one new empty info domain with the specified name.

VMS format:

```
IMAN_CreateDomain domain
```

Arguments:

domain

type: character string
access: read only
mechanism: by descriptor - fixed length

This argument specifies name of the domain to be created.

Fortran interface:

```
INTEGER *4 STAT
CHARACTER *(*) DOMAIN
INTEGER *4 IMAN_CreateDomain
STAT = IMAN_CreateDomain ( DOMAIN )
```

Return codes:

IMAN_SUCCESS
Successful completion.

IMAN_TOOLONG
Name specified is too long.

IMAN_DOMEXIST
Domain with such name already exists.

IMAN_SERVMEM

IMAN_PROTOCOL

IMAN_SERVERBUG

IMAN_BUG
See Appendix A for description of this errors.

IMAN_CreateItem

Creates new info item with specified type and length in current domain.

VMS format:

```
IMAN_CreateItem item, type, length, data
```

Arguments:

item

type: character string
access: read only
mechanism: by descriptor - fixed length
This argument specifies the name of the item to be created.

type

type: character string
access: read only
mechanism: by descriptor - fixed length

This argument specifies the type of the item to be created. See section "Item types" about possible item types.

length

type: longword
access: read only
mechanism: by reference

This argument specifies length of the item in units of item type. It is ignored for the item type "CHAR" but must be specified (any number). Item length for the type "CHAR" is determined by the length of the string you specify to store with the item. See section "Item length" also.

data

type: unspecified
access: read only
mechanism: by reference

This argument declares the initial data stored with the item. For the type "CHAR" this argument should be fixed-length descriptor, for other types - address of array or single element of the corresponding type.

Fortran interface:

```
INTEGER *4 STAT
CHARACTER *(*) ITEM
CHARACTER *(*) TYPE
INTEGER LENGTH
... DATA
INTEGER *4 IMAN_CreateItem
STAT=IMAN_CreateItem(ITEM,TYPE,LENGTH,DATA)
```

Return codes:

IMAN_SUCCESS

Successful completion.

IMAN_SETDOM

Current domain is not set yet. Use function IMAN_SetDomain to establish current domain.

IMAN_NODOMAIN

Current domain does not exist.

IMAN_TOOLONG

Specified data or name are too long.

IMAN_ITEMEXIST

Item with such name already exists.

IMAN_SERVMEM

IMAN_PROTOCOL

IMAN_SERVERBUG

IMAN_BUG

See Appendix A for description of this errors.

IMAN_DeleteDomain

Deletes specified domain and all items in it.

VMS format:

```
IMAN_DeleteDomain domain
```

Arguments:

domain

type: character string
access: read only
mechanism: by descriptor - fixed length

This argument specifies name of the domain to be deleted.

Fortran interface:

```
INTEGER *4 STAT
CHARACTER *(*) DOMAIN
INTEGER *4 IMAN_DeleteDomain
STAT = IMAN_DeleteDomain ( DOMAIN )
```

Return codes:

IMAN_SUCCESS
Successful completion.

IMAN_TOOLONG
Name specified is too long.

IMAN_NODOMAIN
Domain with such name does not exist.

IMAN_PROTOCOL

IMAN_SERVERBUG

IMAN_BUG
See Appendix A for description of this errors.

IMAN_DeleteItem

Deletes specified info item in the current domain.

VMS format:

```
IMAN_DeleteItem item
```

Arguments:

item
type: character string
access: read only
mechanism: by descriptor - fixed length
This argument specifies name of the item to be deleted.

Fortran interface:

```
INTEGER *4 STAT
CHARACTER *(*) ITEM
INTEGER *4 IMAN_DeleteItem
STAT = IMAN_DeleteItem ( ITEM )
```

Return codes:

IMAN_SUCCESS
Successful completion.

IMAN_SETDOM
Current domain is not set yet. Use function IMAN_SetDomain to establish current domain.

IMAN_NODOMAIN
Current domain does not exist.

IMAN_TOOLONG
Specified item name is too long.

IMAN_NOITEM
Item with such name does not exist.

IMAN_PROTOCOL

IMAN_SERVERBUG

IMAN_BUG
See Appendix A for description of this errors.

IMAN_EndBatching

Resets batching mode of communication.

VMS format:

```
IMAN_EndBatching
```

Fortran interface:

```
CALL IMAN_EndBatching
```

Description:

This routine cancels the effect of the IMAN_BeginBatching routine and closes connection with the server.

IMAN_GetItem

Retrieves information stored in the item.

VMS format:

```
IMAN_GetItem item, data
```

Arguments:

item

type: character string
access: read only
mechanism: by descriptor - fixed length
This argument specifies the name of the item.

data

type: unspecified
access: write only
mechanism: by reference
This argument receives data stored with the item. For the type "CHAR" this argument should be address of fixed-length descriptor, for other types - address of array or single element of the corresponding type.

Fortran interface:

```
INTEGER *4 STAT
CHARACTER *(*) ITEM
... DATA
INTEGER *4 IMAN_GetItem
STAT=IMAN_GetItem(ITEM,DATA)
```

Return codes:

IMAN_SUCCESS
Successful completion.

IMAN_SETDOM
Current domain is not set yet. Use function IMAN_SetDomain to establish current domain.

IMAN_NODOMAIN
Current domain does not exist.

IMAN_NOITEM
Item with such name does not exist.

IMAN_TOOLONG
Specified name is too long.

IMAN_TRUNC
Length of item exceeds user supplied character variable length. Result is truncated.

IMAN_PROTOCOL

IMAN_SERVERBUG

IMAN_BUG
See Appendix A for description of this errors.

IMAN_GetItemType

Returns type and length of the specified item.

VMS format:

```
IMAN_GetItemType item, type, length
```

Arguments:

item

type: character string
access: read only
mechanism: by descriptor - fixed length
This argument specifies name of the item.

type

type: character string
access: write only
mechanism: by descriptor - fixed length
This argument receives type of the item. See section "Item types" about possible item types.

length

type: longword
access: write only
mechanism: by reference
This argument receives length of the item in units of the item type.

Fortran interface:

```
INTEGER *4 STAT
CHARACTER *(*) ITEM
CHARACTER *(*) TYPE
INTEGER LENGTH
INTEGER *4 IMAN_GetItemType
STAT=IMAN_GetItemType(ITEM,TYPE,LENGTH)
```

Return codes:

IMAN_SUCCESS
Successful completion.

IMAN_SETDOM
Current domain is not set yet. Use function IMAN_SetDomain to establish current domain.

IMAN_NODOMAIN
Current domain does not exist.

IMAN_NOITEM
Item with such name does not exist.

IMAN_TOOLONG
Specified name is too long.

IMAN_TRUNC
Length of item type string exceeds user supplied character variable length. Result is truncated.

IMAN_PROTOCOL

IMAN_SERVERBUG

IMAN_BUG
See Appendix A for description of this errors.

IMAN_ListDomains

Lists all existing domains.

VMS format:

```
IMAN_ListDomains context, domain, retlen
```


Arguments:

context
type: longword
access: read-write
mechanism: by reference
This argument specifies a list operation context value. Before first call to the routine it must be set to zero. After each call it is updated and should not be changed by the user.

domain
type: character string
access: write only
mechanism: by descriptor - fixed length
This argument receives name of the domain.

retlen
type: longword
access: write only
mechanism: by reference
This argument receives length of the domain name.

Fortran interface:

```
INTEGER *4 STAT
INTEGER CONTEXT
CHARACTER *(*) DOMAIN
INTEGER RETLEN
INTEGER *4 IMAN_ListDomains
STAT=IMAN_ListDomains(CONTEXT,DOMAIN,RETLEN)
```

Description:

This routine returns names of the existing domains. It should be called repeatedly to receive names consequently. Before first call to the routine context variable must be set to zero. When routine finds next domain name it returns code IMAN_SUCCESS and updates the context variable. When there are no more domains to list routine returns code IMAN_NOMORE, empty domain name and zero RETLEN. You can zero the context variable to reset search context to the beginning of domains list at any time.

Return codes:

IMAN_SUCCESS
Successful completion.

IMAN_NOMORE
Successful completion. No more domains in the list.

IMAN_TRUNC
Length of domain name string exceeds user supplied character variable length. Result is truncated.

IMAN_PROTOCOL

IMAN_SERVERBUG

IMAN_BUG
See Appendix A for description of this errors.

IMAN_ListItems

Lists all existing items in the current domain.

VMS format:

```
IMAN_ListItems context, item, retlen
```

Arguments

context
type: longword
access: read-write
mechanism: by reference
This argument specifies a list operation context. Before first call to the routine it must be set to zero. After each call it is updated and should not be changed by the user.

item
type: character string
access: write only
mechanism: by descriptor - fixed length
This argument receives name of item.

retlen
type: longword
access: write only
mechanism: by reference
This argument receives length of item name.

Fortran interface:

```
INTEGER *4 STAT
INTEGER CONTEXT
CHARACTER *(*) ITEM
INTEGER RETLEN
INTEGER *4 IMAN_ListItems
STAT=IMAN_ListItems(CONTEXT,ITEM,RETLEN)
```

Description:

This routine returns names of existing items in the current domain. It should be called repeatedly to receive consequent names. Before first call to the routine the context variable must be set to zero. When routine finds next item name it returns code `IMAN_SUCCESS` and updates the context variable. When there are no more items to list routine returns code `IMAN_NOMORE`, empty item name and zero `RETLEN`. You can zero the context variable to reset search context to the beginning of items list at any time.

Return codes:

IMAN_SUCCESS
Successful completion.

IMAN_NOMORE
Successful completion. No more items in list.

IMAN_TRUNC
Length of item name exceeds user supplied character variable length. Result is truncated.

IMAN_SETDOM
Current domain is not set yet. Use function IMAN_SetDomain to establish current domain.

IMAN_NODOMAIN
Current domain does not exist.

IMAN_TOOLONG
Specified name is too long.

IMAN_PROTOCOL

IMAN_SERVERBUG

IMAN_BUG
See Appendix A for description of this errors.

IMAN_PutItem

Changes information stored in the item.

VMS format:

```
IMAN_PutItem item,data
```

Arguments:

item
type: character string
access: read only
mechanism: by descriptor - fixed length
This argument specifies the name of the item.

data
type: unspecified
access: read only
mechanism: by reference
This argument specifies a new data to store with the item. For the type "CHAR" this argument should be an address of the fixed-length descriptor, for other types - address of the array or single element of the corresponding type.

Fortran interface:

```
INTEGER *4 STAT
CHARACTER *(*) ITEM
... DATA
INTEGER *4 IMAN_PutItem
STAT=IMAN_PutItem(ITEM,DATA)
```


Return codes:

IMAN_SUCCESS
Successful completion.

IMAN_SETDOM
Current domain is not set yet. Use function IMAN_SetDomain to establish current domain.

IMAN_NODOMAIN
Current domain does not exist.

IMAN_NOITEM
Item with such name does not exist.

IMAN_TOOLONG
Specified name or data are too long.

IMAN_PROTOCOL

IMAN_SERVERBUG

IMAN_SERVMEM

IMAN_BUG
See Appendix A for description of this errors.

IMAN_SetDomain

Sets current domain.

VMS format:

```
IMAN_SetDomain domain
```

Arguments

domain
type: character string
access: read only
mechanism: by descriptor - fixed length
This argument specifies name of current domain.

Fortran interface:

```
INTEGER *4 STAT
CHARACTER *(*) DOMAIN
INTEGER *4 IMAN_SetDomain
STAT=IMAN_SetDomain(DOMAIN)
```

Description:

This routine sets a current domain name. It does not check if such domain exists, so if you set non-existing domain name you will get return code `IMAN_SETDOM` later from calls to the routines which operate on items.

Return codes:

IMAN_SUCCESS
Successful completion.

IMAN_TOOLONG
Specified name is too long.

4.2 Client library on UNIX.

4.2.1 Differences from VAX/VMS library.

The library for UNIX was implemented in C language, and can be used in C or C++ only. The names of the routines are basically the same, but the format of the data is changed, for example FORTRAN strings are replaced with the C strings.

Contrary to the VMS library, UNIX version requires manual connection to the server with the IMAN_Connect function. Almost all functions performing data exchange with the server require also so called "connection id" obtained from the IMAN_Connect call, which is a standard C file descriptor.

Another difference consists in the absence of the `IMAN_SetDomain` function in the UNIX library. Instead, the domain name must be specified in each function call requiring this information.

4.2.2 Linking.

To gain access to the client functions you should link to client library in the following way:

```
cc ....  -Limandir -liman ...
```


where `imandir` is the path name of the directory where the `libiman.a` file resides.

4.2.3 Checking for errors.

Most functions in IMAN library on UNIX return condition value, which are listed in the `imandef.h` include file. We strongly recommend for programmers to test the condition value returned after each call like in example below:

```
#include "imandef.h"
int stat ;

stat = IMAN_GetItem ( fd, "Domain", "Some item", data ) ;
if ( stat ) {
  /* error condition occurred */
}
```

4.2.4 IMAN functions.

In this section all functions of the IMAN client library are described in alphabetic order. For each function its calling sequence, types of parameters and possible return codes are presented.

IMAN_Connect

This function establishes a connection between client and server processes and returns a file descriptor for this connection.

Interface:

```
int IMAN_Connect( const char* host, int port );
```

Arguments:

host zero-terminated character string, read only
This argument specifies name of the host on which server is running.

port integer number, read only
This argument specifies the port number at which server is expecting connections from clients.

Calling example:

```
int fd;
fd = IMAN_Connect ( "vxsnd", 5234 ) ;
if ( fd < 0 ) {
  /* error condition */
}
```

Return values:

This function returns a file descriptor for the connection with the server process, which is a positive number. In the case when connection to server fails, it returns a negative number, one of the following codes (defined in `imandef.h` file):

IMAN_SERVERHOST
Error while resolving server host name.

IMAN_SOCKET
Error while creating TCP/IP family socket.

IMAN_CONNECT
Error while trying to connect to server. Usually means that no server is running or wrong port number.

IMAN_Close

Close connection to the server opened previously with the IMAN_Connect function.

Interface:

```
int IMAN_Close( int fdescr );
```

Arguments:

fdescr integer number, read only
This argument specifies the file descriptor for the connection opened previously with the IMAN_Connect function.

Calling example:

```
int stat;
stat = IMAN_Close ( fd ) ;
```


Return values:

This function always returns 0.

IMAN_CreateDomain

Creates one new empty info domain with the specified name.

Interface:

```
int IMAN_CreateDomain( int fdescr, const char* domain ) ;
```

Arguments:

fdescr integer number, read only
This argument specifies the file descriptor for the connection opened previously with the IMAN_Connect function.

domain zero-terminated character string, read only
This argument specifies name of the domain to be created.

Calling example:

```
int stat;
stat = IMAN_CreateDomain ( fd, "Domain" ) ;
```

Return codes:

IMAN_SUCCESS
Successful completion.

IMAN_TOOLONG
Name specified is too long.

IMAN_DOMEXIST
Domain with such name already exists.

See Appendix A for description of other types of errors.

IMAN_CreateItem

Creates new info item with specified type and length.

Interface:

```
int IMAN_CreateItem( int fdescr, const char* domain, const
char* item, const char* type, int length, const void* data );
```

Arguments:

fdescr integer number, read only
This argument specifies the file descriptor for the connection opened previously with the IMAN_Connect function.

domain zero-terminated character string, read only
This argument specifies name of the domain in which the new item to be created.

item zero-terminated character string, read only
This argument specifies the name of the item to be created.

type zero-terminated character string, read only
This argument specifies the type of the item to be created. See section "Item types" about possible item types.

length integer number, read only
This argument specifies length of the item in units of item type. It is ignored for the item type "CHAR" but must be specified (any number). Item length for the type "CHAR" is determined by the length of the string you specify to store with the item. See section "Item length" also.

data varying type, read only
This argument declares the pointer to the initial data stored with the item. For the type "CHAR" this argument should be a pointer to the zero-terminated string, for other types - address of array or single datum of the corresponding type.

Calling example:

```
int stat;
int data1 = 512; /* data to store */
float data2[5] = {12,13,17,28,345};
stat = IMAN_CreateItem( fd, "Domain", "Item1",
                        "INT", 1, &data1 ) ;
stat = IMAN_CreateItem( fd, "Domain", "Item2",
                        "REAL", 5, data2 ) ;
```


Return codes:

IMAN_SUCCESS
Successful completion.

IMAN_NODOMAIN
Specified domain does not exist.

IMAN_TOOLONG
Specified data or name are too long.

IMAN_ITEMEXIST
Item with such name already exists.

See Appendix A for description of other types of errors.

IMAN_DeleteDomain

Deletes specified domain and all items in it.

Interface:

```
int IMAN_DeleteDomain( int fdescr, const char* domain );
```

Arguments:

fdescr integer number, read only
This argument specifies the file descriptor for the connection opened previously with the IMAN_Connect function.

domain zero-terminated character string, read only
This argument specifies name of the domain to be deleted.

Calling example:

```
int stat;
stat = IMAN_DeleteDomain ( "Domain" ) ;
```

Return codes:

IMAN_SUCCESS
Successful completion.

IMAN_TOOLONG
Name specified is too long.

IMAN_NODOMAIN
Domain with such name does not exist.

See Appendix A for description of other types of errors.

IMAN_DeleteItem

Deletes specified info item.

Interface:

```
int IMAN_DeleteItem( int fdescr, const char* domain,
const char* item );
```

Arguments:

fdescr integer number, read only
This argument specifies the file descriptor for the connection opened previously with the IMAN_Connect function.

domain zero-terminated character string, read only
This argument specifies the name of the domain in which the item resides.

item zero-terminated character string, read only
This argument specifies name of the item to be deleted.

Calling example:

```
int stat;
stat = IMAN_DeleteItem ( fd, "Domain", "Item1" ) ;
```

Return codes:

IMAN_SUCCESS
Successful completion.

IMAN_NODOMAIN
Specified domain does not exist.

IMAN_TOOLONG
Specified item name is too long.

IMAN_NOITEM
Item with such name does not exist.

See Appendix A for description of other types of errors.

IMAN_GetItem

Retrieves information stored with item.

Interface:

```
int IMAN_GetItem( int fdescr, const char* domain, const char*
item, void* data );
```

Arguments:

fdescr integer number, read only
This argument specifies the file descriptor for the connection opened previously with the IMAN_Connect function.

domain zero-terminated character string, read only
This argument specifies the name of the domain in which the item resides.

item zero-terminated character string, read only
This argument specifies the name of the item.

data unspecified type, write only
This argument receives data stored with the item. For the type "CHAR" this argument should be address of the string of the sufficient length, for other types - address of array or single element of the corresponding type.

Calling example:

```
int stat;
float data[5];
stat = IMAN_GetItem( fd, "Domain", "Item2", data ) ;
```

Return codes:

IMAN_SUCCESS
Successful completion.

IMAN_NODOMAIN
Specified domain does not exist.

IMAN_NOITEM
Item with such name does not exist.

IMAN_TOOLONG
Specified name is too long.

IMAN_TRUNC
Length of item exceeds user supplied character variable length. Result is truncated.

See Appendix A for description of other types of errors.

IMAN_GetItemType

Returns type and length of the specified item.

Interface:

```
int IMAN_GetItemType( int fdescr, const char* domain, const
char* item, char* type, int* length );
```

Arguments:

fdescr integer number, read only
This argument specifies the file descriptor for the connection opened previously with the IMAN_Connect function.

domain zero-terminated character string, read only
This argument specifies the name of the domain in which the item resides.

item zero-terminated character string, read only
This argument specifies the name of the item.

type zero-terminated character string, write only
This argument receives type of the item. See section "Item types" about possible item types.

length integer, write only
This argument receives length of the item in units of the item type.

Calling example:

```
int stat;
char type[8];
int length;
stat = IMAN_GetItemType( fd, "Domain", "Item2", type,
                         &length ) ;
```

Return codes:

IMAN_SUCCESS
Successful completion.

IMAN_NODOMAIN
Specified domain does not exist.

IMAN_NOITEM
Item with such name does not exist.

IMAN_TOOLONG
Specified name is too long.

IMAN_TRUNC
Length of item type string exceeds user supplied character variable length. Result is truncated.

See Appendix A for description of other types of errors.

IMAN_ListDomains

Lists all existing domains.

Interface:

```
int IMAN_ListDomains( int fdescr, int* context, char* domain
);
```

Arguments:

fdescr integer number, read only

This argument specifies the file descriptor for the connection opened previously with the IMAN_Connect function.

context pointer to integer number, read-write

This argument specifies a list operation context value. Before first call to the routine it must be set to zero. After each call it is updated and should not be changed by the user.

domain zero-terminated character string, write only

This argument receives name of the domain.

Calling example:

```
int stat;
int context;
char domain[256];
stat = IMAN_ListDomains( fd, &context, domain ) ;
```

Description:

This routine returns names of the existing domains. It should be called repeatedly to receive names consequently. Before first call to the routine context variable must be set to zero. When routine finds next domain name it returns code IMAN_SUCCESS and updates the context variable. When there are no more domains to list routine returns code IMAN_NOMORE, empty domain name and zero RETLEN. You can zero the context variable to reset search context to the beginning of domains list at any time.

Return codes:

IMAN_SUCCESS

Successful completion.

IMAN_NOMORE

Successful completion. No more domains in the list.

IMAN_TRUNC

Length of domain name string exceeds user supplied character variable length. Result is truncated.

See Appendix A for description of other types of errors.

IMAN_ListItems

Lists all existing items in the specified domain.

Interface:

```
int IMAN_ListItems( int fdescr, const char* domain,
int* context, char* item );
```

Arguments

fdescr integer number, read only

This argument specifies the file descriptor for the connection opened previously with the IMAN_Connect function.

domain zero-terminated character string, read only

This argument specifies the name of the domain in which the items reside.

context pointer to integer number, read-write

This argument specifies a list operation context. Before first call to the routine it must be set to zero. After each call it is updated and should not be changed by the user.

item character string, write only

This argument receives name of item.

Calling example:

```
int stat;
int context;
char item[256];
stat = IMAN_ListItems( fd, "Domain", &context, item ) ;
```

Description:

This routine returns names of existing items in the current domain. It should be called repeatedly to receive consequent names. Before first call to the routine the context variable must be set to zero. When routine finds next item name it returns code IMAN_SUCCESS and updates

the context variable. When there are no more items to list routine returns code IMAN_NOMORE, empty item name and zero RETLEN. You can zero the context variable to reset search context to the beginning of items list at any time.

Return codes:

IMAN_SUCCESS
Successful completion.

IMAN_NOMORE
Successful completion. No more items in list.

IMAN_TRUNC
Length of item name exceeds user supplied character variable length. Result is truncated.

IMAN_NODOMAIN
Specified domain does not exist.

IMAN_TOOLONG
Specified name is too long.

See Appendix A for description of other types of errors.

IMAN_PutItem

Changes information stored with item.

Interface:

```
int IMAN_PutItem( int fdescr, const char* domain, const char*
item, const void* data );
```

Arguments:

fdescr integer number, read only
This argument specifies the file descriptor for the connection opened previously with the IMAN_Connect function.

domain zero-terminated character string, read only
This argument specifies the name of the domain in which the item resides.

item zero-terminated character string, read only
This argument specifies the name of the item.

data unspecified type, read only
This argument specifies a new data to store with the item. For type "CHAR" this argument should be address of the zero-terminated string, for other types - address of array or single element of the corresponding type.

Calling example:

```
int stat;
float data[5] = {12,13,17,28,345};
stat = IMAN_PutItem( fd, "Domain", "Item2", data ) ;
```

Return codes:

IMAN_SUCCESS
Successful completion.

IMAN_NODOMAIN
Specified domain does not exist.

IMAN_NOITEM
Item with such name does not exist.

IMAN_TOOLONG
Specified name or data are too long.

See Appendix A for description of other types of errors.

5 IMAM management task

5.1 IMAM management task on VMS

IMAM is a simple task built on top of IMAN client library for interactive control of IMAN functions. It has command-based user interface and built-in help facility. To start IMAM you simply type

```
$ RUN IMAN:IMAM
IMAM>
```

"IMAM> " here represents program prompt. In response to it user may type any of the commands described below. Usually commands map to specific client library functions. Follows an alphabetic list of all commands with brief description.

CREATE

CREATE command creates one new item inside current domain (see SET/DOMAIN command). Format:

```
CREATE/TYPE=type ITEM_NAME INITIAL_DATA
```

type is keyword one of the CHAR, BYTE, SHORT, LONG, INT, REAL, DOUBLE. ITEM_NAME specifies name of the item to be created. It is specified according standard VMS rules - name consisting of characters other than uppercase letters, digits, dollar sign ($) and underscore (_) must be surrounded by quotation marks ("). INITIAL_DATA specifies data to be stored with created item. Its format depends on type of created item. For type CHAR this should be string of characters (again in standard VMS format), for other types this is a comma-separated list of numbers.

CREATE/DOMAIN

This command creates new empty domain. Format:

```
CREATE/DOMAIN DOMAIN_NAME
```

DOMAIN_NAME specifies name of the domain too be created.

DELETE

This command deletes specified item in current domain. Format :

```
DELETE ITEM_NAME
```

ITEM_NAME specifies name of item to be deleted.

DELETE/DOMAIN

This command deletes specified domain and all items in this domain. Format:

```
DELETE/DOMAIN DOMAIN_NAME
```

DOMAIN_NAME specifies name of the domain to be deleted.

LIST

This command displays list of items inside the current domain. Format:


```
LIST [pattern]
```

When pattern is specified only the names are displayed which match this pattern. Without pattern all items are displayed. You may use wildcard characters in pattern to select group of items.

LIST/DOMAINS

This command displays list of existing domains. Format:

```
LIST/DOMAINS [pattern]
```

When pattern is specified only the names are displayed which match this pattern. Without pattern all domains are displayed. You may use wildcard characters in pattern to select group of domains.

HELP

This command invokes help facility to display information on selected topic. Format:

```
HELP [topic]
```

PUT

This command changes data stored with specified item. Format:

```
PUT ITEM_NAME DATA
```

ITEM_NAME specifies name of the item for which to change stored data. DATA specifies new data to be stored with item. See CREATE command on how to specify item name and data.

SET/DOMAIN

This command changes current domain. Format:

```
SET/DOMAIN DOMAIN_NAME
```

DOMAIN_NAME is name of the domain to be established as the current domain.

SHOW

This command displays all information about specified item. Format:

```
SHOW ITEM_NAME
```

ITEM_NAME specifies name of the item for which information to be displayed.

5.2 IMAM management task on UNIX

IMAM task on UNIX is a bit simpler than on VMS and does not allow to update the information in the database, nor to create new domains or items. The list of available commands can be obtained during the session or from the following example (assuming all configurations correct and the directory with the binary is in the shell's path):

```
% imam
imam> help

IMAM commands available :
    'ld [regex]'  - list domains
    'cd domain'   - set current domain
    'ls [regex]'  - show item list in current domain
    'ex [regex]'  - examine one or more items in current domain
    'help' or '?' - this message

 regex - regular expression specifying domain or item name

imam>
```

Using imam you can quickly examine the set of items in one domain or list the contents of the database. Note that it uses regular expressions, see ed(1) for the reference.

6 IMAB - X/Motif browser tool.

In order to utilize the benefits of the GUI, the accompanying X/Motif browsing tool, with the name abbreviated as IMAB, has been developed under UNIX systems as a part of the IMAN system. While preserving transparent access to a full set of the IMAN client's library functions, the browser in addition has a built-in monitor of the IMAN database contents. The monitor allows for the periodic checking of the items selected by users, and displaying them in a separate dialogue window using a simple formatting layout. This capability can be used for a quick debugging of users' applications without writing a complex code where it seems to be sufficient.

More specifically the interface provided by IMAB consists of a main window (see figure 4), monitor dialogue window (see figure 5), and several small input dialogues for modifying different parts of database contents (see figures 6-8.)

6.1 Main tool window.

The main window (figure 4) has a number of the active components providing a general control of the browser's operation and the navigation in the server's database. Among them (going from the top-left corner of the window) are:

- Task bar with several "standard" Motif-style pop-down menus: "File", "View", "Tools" and "Help".
- Server host selection entry.
- Two listboxes "DOMAINS" and "ITEMS" containing the lists of IMAN domains and items (when a domain is selected) respectively.
- An "Update" button bellow the listboxes (see bellow).
- A "Current Selection" box with a complete information concerning the selected item (when a domain and an item are both selected). This box has a slider to view the contents of the array-like items.
- "Domain" area with two buttons operating on the whole domains: "Add" - to add a new domain to the database and "Remove" - remove currently selected domain (together with all the items it may contain) from the database.
- "Item" area with a set of buttons operating on the individual items. The "Add" button brings up a dialogue window (figure 7) with a form describing the new item, its name, type, and length (when the item is an array), and an optional list of initialization values. "Remove" button is used to remove currently selected item from the database. "Modify" button allows to change the item's data "on the fly". The remaining button "Append them to Values Monitor List" is used to add the currently selected item to the list of items whose values are fetched periodically from the server's database and displayed in the monitor window.

The mentioned above "Update" button lets the browser to synchronize the state of currently selected components with the actual contents of the database. This feature is provided in a case when other applications working in parallel are modifying the database contents (adding/removing domains, adding/removing/modifying items).

6.2 Monitor window.

This dialogue window (figure 5) is brought up from the "Tools" pop-down menu of the main window. Unlike other dialogue windows grabbing the input, the "Values Monitor" operates in parallel with the main window.

The main part of the window is a formatted list of items reflecting their current state. The items are periodically updated with the interval given (in seconds) below the list. The items are added to the list from the main window during the navigation session. Three control buttons are provided in the right bottom corner of the monitor window in order to exclude items from the monitor list and to change their relative positions in the list.

The appearance of the long (array-like) items may be controlled with a slider on the window's bottom. The slider allows to specify an index of the first element of the item.

WARNING: Remember, please, that very long, very complex, and frequently updated monitor list may have a serious performance impact on the IMAN server you are connected to. So, try to avoid making the monitor list too long, especially with array-like items, or making very frequent updates.

6.3 Other dialogues.

There are also other dialogues which are quite simple and can be understood without description here. See figures below to get an idea how do they look like.

7 Installation and Set-up

7.1 Installation on VMS

Installation of IMAN is quite simple. First you need to select directory dedicated to IMAN. It can be any directory, possibly shared with other products but usual way is to create separate directory for it, for example DISK:[IMAN]. After you selected directory you should copy following files to this directory:

IMAN_STARTUP.COM

Command procedure for starting IMAN. After installation you should edit this file and change some parameters in it, see below for more information about these parameters. This file should be executed from the SYSTEM account, usually as a part of the startup procedure.

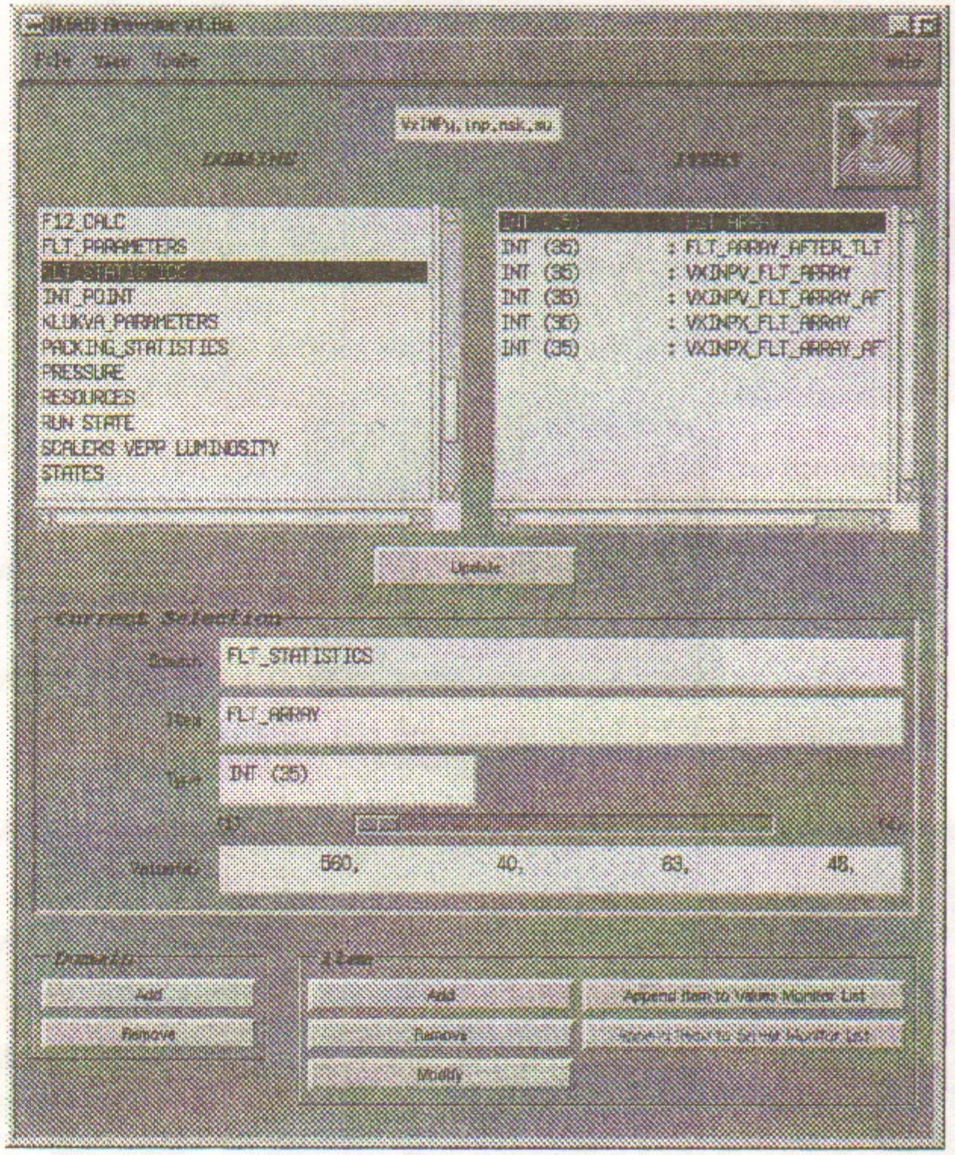


Figure 4: Main window of the browser tool.

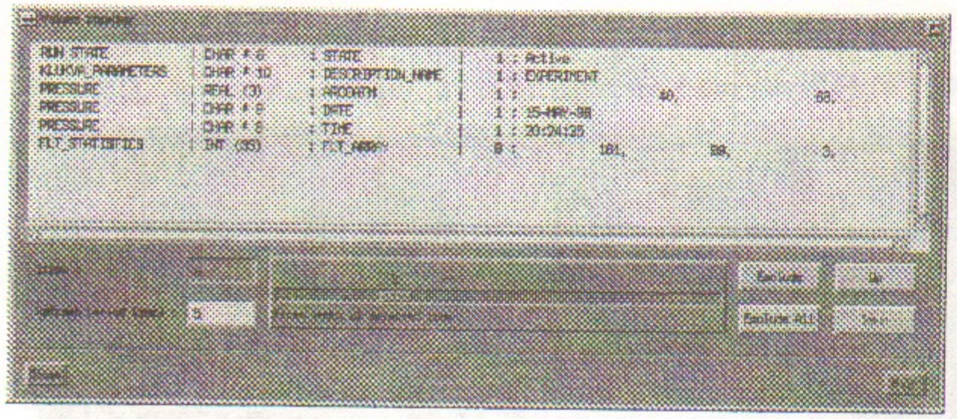


Figure 5: Monitor window of the browser tool.

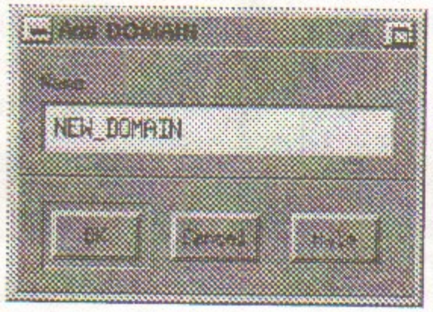


Figure 6: Domain creation dialog window of the browser tool.

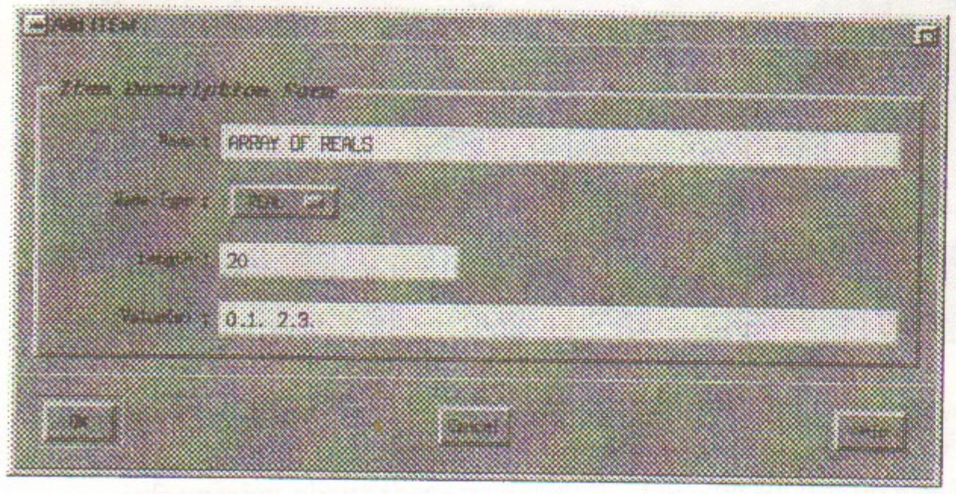


Figure 7: Item creation dialog window of the browser tool.

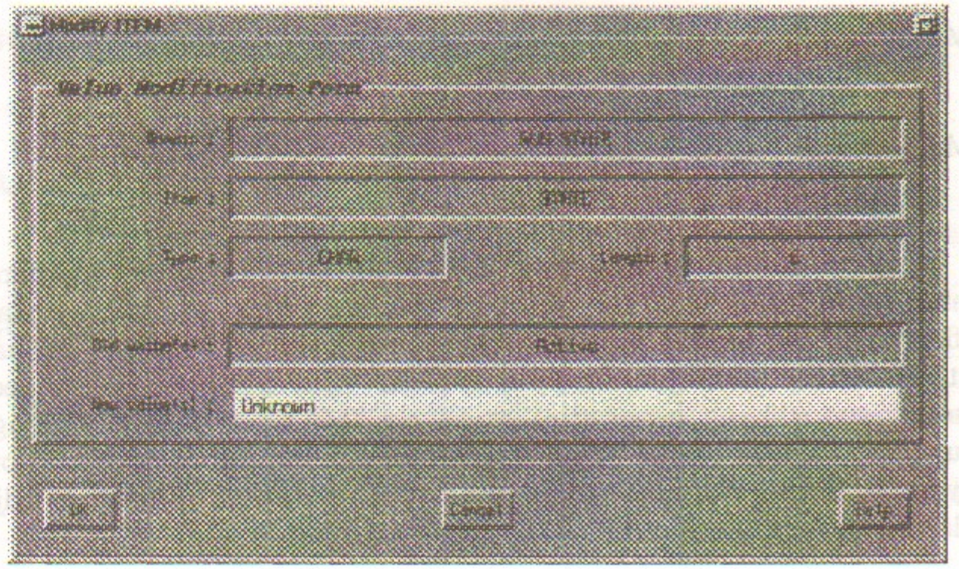


Figure 8: Item data modification dialog window of the browser tool.

IMAN_SERVER.EXE

Server process executable.

IMAN_SERVER.CONFIG

Server configuration file. See the section 7.2 "Server configuration" for the desription of parameters contained in this file.

IMAN_SHARE.EXE

Shareable client library.

IMAN_MSGS.OBJ

IMAN messages file.

IMAN.OPT

Option file for linker.

IMANDEF.H

IMANDEF.FOR

Include files with condition values.

IMAM.EXE

IMAM executable.

IMAM.HLB

IMAM help library.

Then this you should edit file `IMAN_STARTUP.COM` and change the parameters in the "configuration section" of this file. First you should change definition of the symbol `iman_directory` which should be set to the name of the directory where you have placed IMAN files. Then you should decide whether you will run server or client part of IMAN or both. According to your requirements you set value of symbols `client` and `server` to 0 or 1. If you decided to run client programs on your host then you should adjust following parameters:

install_client — set it to 1 or 0 according to your decision whether to install client shareble image or not.

transport — defines trasport for the communication of client programs with the server. Can be one of "LOCAL" (for VMS mailboxes) or "TCPIP" (for TCP/IP protocol). Usually when clients are running on the same node as server "LOCAL" is the best choice. "TCPIP" must be used when client is running on different than server host.

host_name — for the "TCPIP" transport defines the name of the host where the server is running.

port — TCP/IP port number for the communication with server. This value should correspond to the INET_PORT_NUMBER in the configutation file of the server with which you wish to communicate.

For the server process there is only one relevant parameter in this file:

queue — defines the name of the batch queue on which the server process will be executing. When this parameter is empty string then the server is started as a stand-alone process.

7.2 Server configuration

If you decided to run IMAN server at your node then, probably, you will need to adjust the server configuration, wich can be done with the parameters in `IMAN_SERVER.CONFIG` file. Follows a brief overview of these parameters:

SECTION_FILE

Defines name of file used for section mapping. Default name is `IMAN:IMAN_SECTION_FILE`.

BACKUP_FILE

Defines name of file used for storing backup information. Default name is `IMAN:IMAN_SECTION_FILE.BACKUP`.

SECTION_SIZE

Defines the initial size of section file in blocks. If section file does not exist it will be created by server with size specified by this parameter. If the file exists and its size is lower than value of this parameter, the file will be extended. If the file size is grater than the value supplied for parameter, its value will be ignored.

SECTION_SIZE_MAX

Defines maximum size of section in blocks. When server exhausts section memory it tries to extend it. To prevent server from unlimited expansion supply reasonable value for this parameter.

EXTEND_SIZE

Defines at how many blocks the section file will be expanded after exhausting available space.

UPDATE_TIME

Defines the time interval for the explicit updates of the section file. This time interval is specified in the standard VMS delta time format. Grater intervals reduce the overhead for updating and can improve performance, smaller intervals can be safer in case of computer crash.

BACKUP_TIME

Defines the time interval for making backup copies of the section file. This time interval is specified in the standard VMS delta time format. Server makes exact copies of section file at the specified periods of time. Although it saves only part of file which is actually used for storing information this can take a lot of time for big section files and produce delays in server responses to client requests.

BACKUPS_KEEP

Defines how many backup copies to keep. Server purges backup files after each backup. It will keep specified number of most recent backups.

HOST_ACCESS_LIST

List of the Internet names of hosts. Only the requests from the clients running at the hosts included in this list will be accepted.

INET_PORT_NUMBER

TCP/IP port number on which the server will be listening for incoming connections.

DEBUG_LEVEL

Specifies level of debugging messages. Set 0 to turn it off.

7.3 Installation on UNIX

Installation on UNIX platforms consists in the compilation of the object library and manager application (imam). Compilation is handled with the make utility, makefile was written for the GNU version of make. To build everything simply type:

```
% gmake all
```

(assuming that GNU make is installed with the name `gmake`.) This will produce both library and executable. You can copy them together with the include file in the appropriate place, like in example:

```
% cp libiman.a /usr/local/lib
% cp imandef.h /usr/local/include
% cp imam /usr/local/bin
```

(you may need to rerun `ranlib` on the library on some systems.)

8 Troubleshooting

Mostly, inconsistencies in the data structures can appear due to broken synchronization between structures in memory and section file. Using paging facility of system, information corruption in server can occur only when computer, on which IMAN server resides, crashes. In all other cases system will take care about consistency of information in server memory and disk file. Surely, data structures in server memory may become corrupted also due to bugs in server.

Consistency check of information in server is not implemented yet in this release. Possibly in next major release we will add some checks to the server

code which will detect corrupted information and report such cases. This version of server tries to implement minimal security of information using simple backup scheme. Server makes copies of data structures to separate file in regular time intervals (using parameters `BACKUP_FILE`, `BACKUP_TIME` and `BACKUPS_KEEP` in `IMAN_SERVER.CONFIG` file). These backups can be freely interchanged with section file. So if information in section file become corrupted you can try to use backup file instead of original section file. It can be also recommended to save stable versions of section file (or backup file) with different name, because backup scheme of server does not guarantee even consistency of information in backup file and backup files can also be broken together with original section file.

A Error codes

IMAN functions in case of errors do not produce any messages, instead they return condition value which can be interpreted in standard way (with functions `sys$getmsg()` or `lib$signal()` on VMS). This condition values are defined in include files `IMAN:IMANDEF.FOR` and `IMAN:IMANDEF.H` on VMS or `imandef.h` on UNIX. Table below describes condition values in details, grouped by their severety level.

Severety: Success.

IMAN_SUCCESS

This code is returned by IMAN functions in case of successful completion. Reason for this is that IMAN works properly. No actions needed.

IMAN_NOMORE

No more domains or items. This value is returned in list operations when there are no more domain or items left to list. Action — stop calling list routines.

Severety: Warning.

IMAN_DOMEXIST

Warning. Domain exists. Returned while trying to create domain with name which already exists. Action - check if such domain really exists.

IMAN_ITEMEXIST

Item exists. Returned while trying to create item with name which already exists. Action - check if such item really exists.

IMAN_TRUNC

Character string truncated. Returned by functions which operate with character strings when returned value is longer than character variable supplied by user. Action - provide longer character variable.

Severety: Error.

IMAN_NARG

Illegal number of arguments. User supplied wrong number of arguments to one of the client routines. Action — check your code carefully.

IMAN_SETDOM

Domain is not set. Returned while trying to do operations on items without previously setting current domain name. Action — use IMAN_SetDomain to set current domain.

IMAN_CLOSED

Server closed connection. After some time of communication IMAN server closed connection. This is possibly networking problem on your site or bug in the server. Action — communicate with person which installed IMAN library on your node.

IMAN_COMM

Cannot communicate with server. Communication part of IMAN cannot establish connection with server. Possible cause - server is not running or network is down. Action - communicate with your system manager to locate possible failure.

IMAN_TOOLONG

Data too long to fit in buffer. Amount of data stored with item and all names in IMAN are limited in length. An attempt to exceed limit gives this return value. Action - reduce length of name or length of data stored with item.

IMAN_PROTOCOL

Protocol crashed. Communication part of IMAN detected errors in data flow protocol between client and server. Possible reasons — bug in IMAN library or network problems. Action — communicate with person which installed IMAN library on your node.

IMAN_ITEMTYPE

Illegal item type. Illegal item type specified when trying to create new item. Action — check the parameter decribing item type in the call to library functions.

IMAN_NODOMAIN

No such domain. This condition appears when user specifies non-existing name for domain. Action — check if domain with name specified really exists.

IMAN_NOITEM

No such item. This condition appears when user specifies non-existing name for item. Action — check if item with name specified really exists.

IMAN_SERVMEM

Server memory error. Server depository memory is limited and can exhaust when there are too many information stored. Action — remove all unneeded items or contact with person who installed IMAN on your site on subject of increasing memory limit.

IMAN_SHORTBUF

Client buffer too short. Inconsistency in buffer lengths between client and server. Action — report this to person who installed IMAN on your site.

Severity: Fatal

IMAN_TRANSPORT

Wrong transport name. Transport name as obtained from logical name `IMAN_TRANSPORT` is not legal transport name. Action — check logical name `IMAN_TRANSPORT`.

IMAN_SERVNAME

Cannot resolve server host name. Communication part of client library can not resolve host name on which server resides. This is possibly due to network problems when operating in multihost environment or errors in configuration while compiling IMAN. Action - communicate with person which installed IMAN library on your node.

IMAN_ILLPORT

Illegal port number. Logical name name `IMAN_PORT` specifies port number which is lower than 1024. Action — contact IMAN manager at your site.

IMAN_CREATSOCK

Cannot create socket. Communication part of IMAN cannot create socket for communicating with server. Possible cause is in networking software. Action - communicate with person which installed IMAN library on your node.

IMAN_SERVERBUG

Bug in server. Client part of IMAN detected some inconsistency in data received from server. We hope you will never get this return code. Action — report this to person who installed IMAN on your site.

IMAN_BUG

Bug in IMAN. Client part of IMAN detected some conditions which cannot be interpreted in reliable way. We hope you will never get this return code. Action — report this to person who installed IMAN on your site.

B Programming examples

Next follow some simple examples of code dealing with IMAN client functions.

B.1 Get item data from server

```
      program Example1

      integer*4 stat
      integer*4 iArray(5)
      integer*4 IMAN_GetItem, IMAN_SetDomain

      ...

c     ------ First we need to set current domain ------
      stat=IMAN_SetDomain('Domain #1')
      if ( .not. stat ) then
         call lib$signal(%val(stat))
         stop
      end if

c     --- Assume there exist "Item1" of type INT and length 5 ---
      stat=IMAN_GetItem('Item1',iArray)
      if ( .not. stat ) then
         call lib$signal(%val(stat))
         stop
      end if

      ...

      end
```

B.2 Change item data

```
      program Example2

      integer*4 stat
      integer*4 iArray(5)
      integer*4 IMAN_PutItem, IMAN_SetDomain

      ...

c     ------ First we need to set current domain ------
      stat=IMAN_SetDomain('Domain #1')
      if ( .not. stat ) then
         call lib$signal(%val(stat))
         stop
      end if

c     --- Assume there exist "Item1" of type INT and length 5 ---
      iArray(1) = 1
      ...
      stat=IMAN_PutItem('Item1',iArray)
      if ( .not. stat ) then
         call lib$signal(%val(stat))
         stop


      end if

      ...

      end
```

B.3 Set batching mode

```
      program Example3

      integer*4 stat
      integer*4 iArray(5)
      integer*4 IMAN_GetItem, IMAN_SetDomain

      ...

      call IMAN_BeginBatching

      stat=IMAN_SetDomain(...)
      if ( .not. stat ) then
         call lib$signal(%val(stat))
         stop
```

```
      end if

      stat=IMAN_GetItem('Item1',...)
      ...
      stat=IMAN_GetItem('Item2',...)
      ...
      stat=IMAN_GetItem('ItemN',...)
      ...
c     Close connection to server
      call IMAN_EndBatching

      ...

      end
```

B.4 Programming IMAN on UNIX

```
int fd ;
int stat ;
int data[5] ;

fd = IMAN_Connect ( "hostname", 3255 ) ;
if ( fd < 0 ) {
  exit ( errno ) ;
}

/*
  assuming that item with the name "Item1" in the domain
  "Domain" of the type "INT" and length 5 exists
*/
stat = IMAN_GetItem( fd, "Domain", "Item1", data ) ;

IMAN_Close (fd) ;
```

B.5 Getting the list of items

```
int fd ;
int stat ;
int context ;
char item[256] ;

fd = IMAN_Connect ( "hostname", 3255 ) ;
if ( fd < 0 ) {
  exit ( errno ) ;
}

while ( 1 ) {

  stat = IMAN_ListItems( fd, "Domain", &context, item ) ;
  if ( stat != IMAN_SUCCESS ) break ;

  printf ( "Item: %s\n", item ) ;

}

IMAN_Close (fd) ;
```


References

[1] D.A.Bukin et al., *Data Acquisition System of SND Experiment*, Proceedings of the International conference on Computing in High Energy Physics - CHEP97, Berlin, April 7-11, 1997.

Contents

I.A. Gaponenko, A.A. Salnikov

Information management system for SND experiment

И.А. Гапоненко, А.А. Сальников

Распределенная система обмена информацией (IMAN) для эксперимента СНД

Budker INP 98-39

Ответственный за выпуск А.М. Кудрявцев
Работа поступила 20.05. 1998 г.

Сдано в набор 22.05.1998 г.
Подписано в печать 22.05.1998 г.
Формат бумаги 60×90 1/16 Объем 2.8 печ.л., 2.3 уч.-изд.л.
Тираж 90 экз. Бесплатно. Заказ № 39

Обработано на IBM PC и отпечатано на
ротапринте ИЯФ им. Г.И. Будкера СО РАН,
Новосибирск, 630090, пр. академика Лаврентьева, 11.