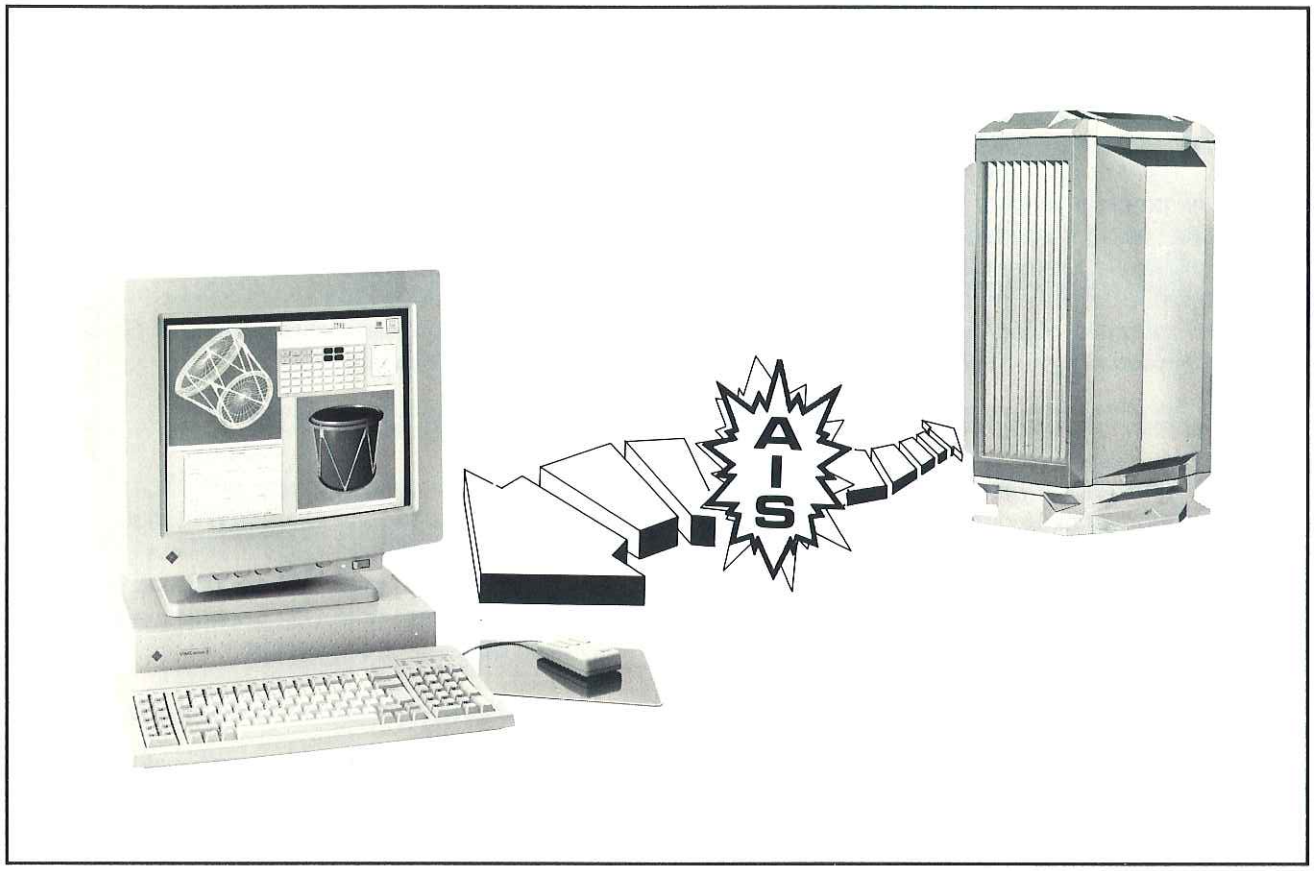
Cancelled
Per Trans 021-01
May 1997

I/A Series® Application Interface Software for Sun Microsystems SPARCstation and SPARCserver Products



Application Interface Software (AIS) provides an Application Program Interface (API) resident in a host computer. Host computer applications use AIS to gain access to objects, files, and/or virtual terminal interface in connected I/A Series systems. This application program interface, following international standards, insulates the customer application programs from host computer specifics. Customer applications deal with functional requirements instead of communication implementation details.

Host computer programmers write application programs using AIS functions to perform such functions as process variable access, file transfer, and terminal interface. The AIS routines work in conjunction with one or more Information Network Interface (INI) modules configured in connected I/A Series systems. AIS works with Information Network Interface modules direct or network connected to the host computer system. The Host computer vendor determines appropriate Host system hardware and software to support the connection.

Application functions available with AIS:

- Non-connected named process variable access
- Connected named process variable access
- Exception based, named process variable access
- Named file upload and download

FOXBORO®
A SIEBE COMPANY

- Named file partial upload
- Named file partial download
- Programmatic virtual terminal interface
- Redirection of virtual terminal input and output to and from files
- Menu-driven virtual terminal access
- Menu-driven file transfer

AIS optionally provides circuit optimization and data object path redundancy on multiple INI systems. The AIS package, by configuration, apportions process variable lists among several Information Network Interface modules. AIS also uses configurable alternate INI paths for object access upon failure of a particular INI. This maintains operating object access services originally provided by the failed INI path.

AIS additionally offers application program development tools. This package includes utility programs for exercising the package functions and for comparing results during application development. Several optimizations are configurable for tuning the performance of the host computer application under development.

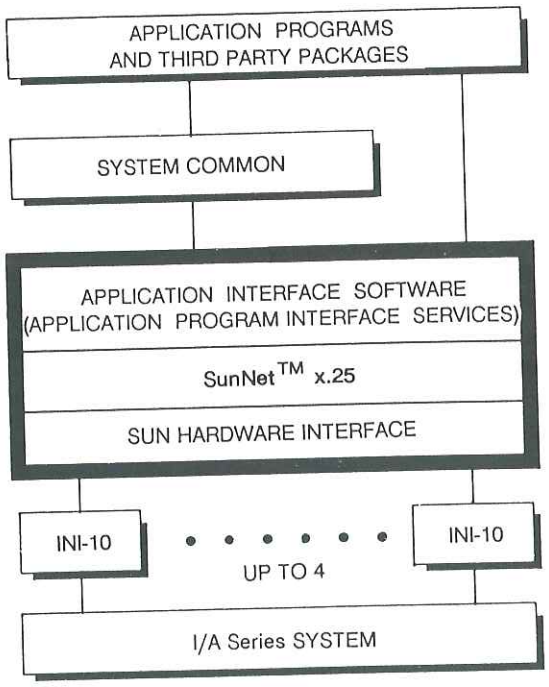


Figure 1. Layered Application Program Interface

AIS SERVICES

The Application Interface Software (AIS) accesses data objects, files residing in the I/A Series stations, and Application Processor terminal sessions for user-written application programs resident in Host computers. Figure 2 illustrates the relationship between Host computer programs and I/A Series services.

AIS runs in a Host computer to support redundant program access of I/A Series objects via INIs. It includes C-callable functions used by the Host programs to read global data from I/A Series Processors, to write data to I/A Series Processors, and to execute terminal sessions in I/A Series Application Processors.

ACCESSING DATA OBJECTS

The methods of object access are:

- Unbuffered read/write of data objects
- Buffered read/write of data objects
- Buffered read/write with continuous updates
- Buffered read/write with continuous updates and queuing of all changes

Unbuffered Read/Write of Data Objects

This access method is appropriate when the data objects are accessed infrequently. Data objects of all value types can be read or written. Both connectable and non-connectable data objects may be accessed. Related functions are:

-uread
-uwrite
-sread
-swrite

UNBUFFERED READ OF NON-STRING DATA OBJECTS

The caller of the uread function specifies the gateway paths to be used and a set of data objects by name. The function returns the value, status, and error code for each non-string data object.

UNBUFFERED WRITE OF NON-STRING DATA OBJECTS

The caller of the uwrite function specifies the gateway paths to be used, a set of data objects by name, and the new values for the data objects. All data objects that are accessible are updated. An error code is returned for each data object.

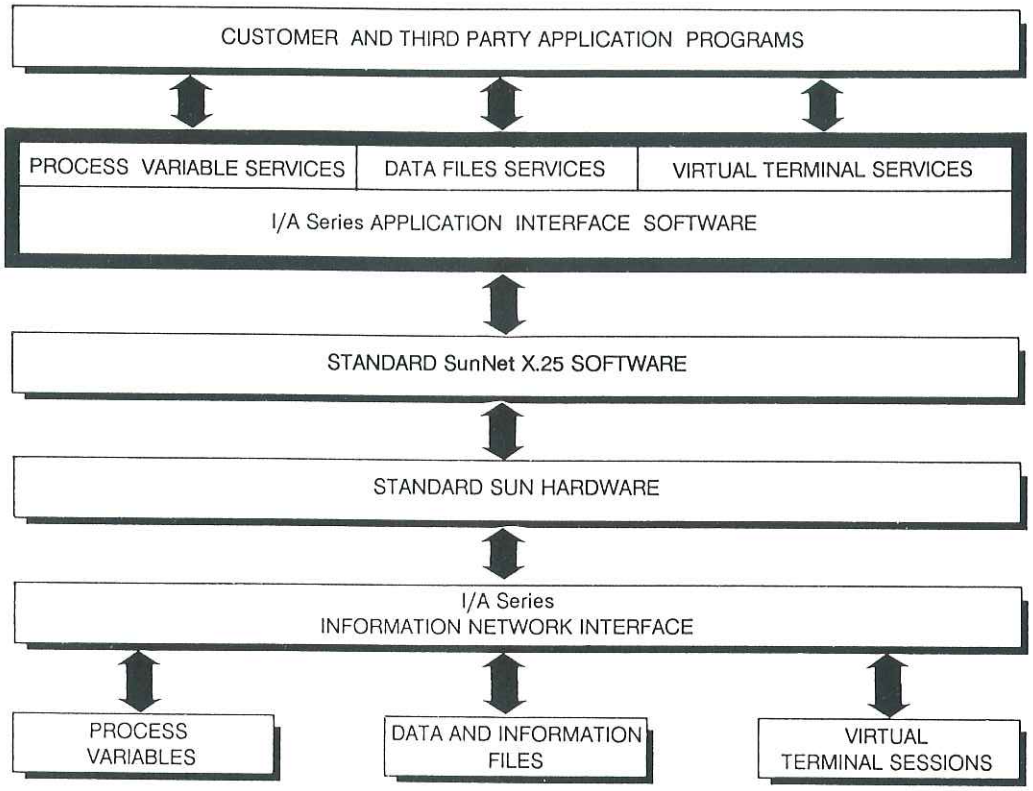


Figure 2. Host Services Relationship

UNBUFFERED READ OF STRING DATA OBJECTS
One data object may be read at a time.
The caller of the sread function specifies the gateway path to be used and the name of a data object. The function returns the string, the status, and an error code for the data object.

UNBUFFERED WRITE OF STRING DATA OBJECTS
One string data object may be written at a time.
The caller of the swrite function specifies the gateway path to be used, the name of a data object, and the string. The function returns the status and an error code for the data object.

Buffered Read/Write of Data Objects

This access method is appropriate when data objects are being accessed frequently. Only connectable data objects can be accessed.
Related functions are:

-bopen
-sbopen
-bread
-bwrite
-clsset

SPECIFICATION OF DATA OBJECTS FOR BUFFERED READ/WRITE
The bopen function establishes the change-driven connections for a set of data objects. The data objects are then accessible by using the bread and bwrite functions. The caller optionally specifies the gateway paths. The caller specifies a set of data objects by name, change value (delta), assumed value type, and an access mode (read-only or read/write). The function makes the connections to the data objects and returns a data set number to be used in other calls to reference the set of data objects, a status, and an error code per data object informing the caller if and why a connection cannot be made.

BUFFERED READ OF DATA OBJECTS
The caller of the bread function specifies a set of data objects by data set number. The function returns the value and status for each data object in the data set.

BUFFERED WRITE OF DATA OBJECTS
The caller of the bwrite function specifies a set of data objects by data set number and the new values. All data objects that are accessible are written. An error code is returned for each data object in the data set.

Buffered Read/Write with Continuous Update

This access method is appropriate when you want to have values updated automatically in memory or files without doing any bread calls. Related calls are:

-copen
-scopen
-bread
-bwrite
-clsset

The copen call establishes the change-driven connections and starts the updating process. The caller optionally specifies the gateway path(s) to be used. The caller specifies a set of data objects by name, change value (delta), assumed value type, access mode (read-only or read/write), and an array to return the indexes into shared memory arrays where the value and status are stored. The function returns a data set number to be used in the bread, bwrite, and clsset calls.

When using scopen, it is also possible to specify the frequency of checking the delta. By updating the write value table in shared memory with wrtval, it is possible to invoke a change-driven write of the value to the I/A Series object.

The readval, readwval, readsta, readccnt, readwcnt, mreaidx, and mreawidx calls return read and write value, status, read and write change count, and multiple read and write information from the AIS value, status and change count shared memory arrays.

Note that bread calls, though possible, are unnecessary, since value and status information is automatically updated to shared memory arrays.

Buffered Read/Write with Continuous Update and Queuing of Changes

This access method is appropriate when you want the application program to be made aware of every change in value greater than or equal to the delta and/or every change in status of object in the set. Related functions are:

-qopen
-sqopen
-qread
-bread
-bwrite
-clsset

The qopen call establishes the change-driven process variable object connections and starts the updating and queuing process. The caller optionally specifies the gateway path(s) to be used. The caller specifies a set of data objects by name, change value (delta), assumed value type, and an array to return the indexes into the shared memory array where the value and status are stored. The function returns a data set number to be used in the qread, bread, bwrite, and clsset calls.

When using sqopen, it is also possible to specify the frequency of checking the delta. By updating the write value table in shared memory with wrtval, it is possible to invoke a change-driven write of the value to the I/A Series process variable object.

The readval, readwval, readsta, readccnt, readwcnt, mreaidx, and mreawidx calls return read and write value, status, read and write change count, and multiple read and write information from the AIS value, status and change count shared memory arrays.

Note that bread calls, though possible, are unnecessary, since value and status information is automatically updated to shared memory arrays.

EXTRACTING CHANGES FROM A CHANGE QUEUE
The caller of the qread function specifies a set of data objects by data set number. The function returns the changes (value and status) from its queue for that data set. Each change is accompanied by a data set entry number. Some data objects may have no changes associated with them; others may have several changes. The maximum number of changes to be returned is specified by the caller. The actual number of changes is returned to the caller. A return code indicates the state of the queue.

CLOSING A SET OF DATA OBJECTS
Data sets previously opened with the bopen, sbopen, copen, scopen, qopen, or sqopen call can be closed with the clsset call.

Obtaining Information about a Data Set

AIS provides routines that can optionally be used to aid in developing an application. These routines return information that is available when the data set is opened. Providing this information relieves the application from housekeeping. Related functions are gsinfo, getnam, getscn, gsnent, get_set_name, and get_set_num.

Redundancy

The goal of the redundancy is to keep as many objects accessible as possible when INI(s) fail.

To supply redundancy it is necessary to have two or more INIs in the same group with some free lists. The redundancy algorithms keep as many lists open on the preferred INI as possible, or, as a second choice within a group. The result is that lists are moved from a failing preferred INI to a functional INI, then back to their preferred INI as soon as the preferred INI returns to a functional state ("OK").

The state of the INI is constantly monitored by the application program interface software. If one of the INIs changes state (from OK to FAILED or FAILED to OK), then the following actions are performed:

- All lists not currently opened on any INI are opened on their preferred INI. Interface software opens these lists if the INI is OK and has available resources. Unopened lists occur when other INIs have no available resources to back up a failed INI module.
- All lists currently opened on backup INIs are moved to their preferred INI. Interface software opens these lists if the preferred INI is OK and has available resources.
- All lists not currently opened by interface software on any INI module are opened on any INI module in the same group. Interface software opens these lists if the target INI is OK and has available resources.

ACCESSING I/A Series FILES

There are four methods of file access:

- Transferring a file from an I/A Series Application Processor to the Host.
- Transferring a file from the Host to an I/A Series Application Processor.
- Reading part of a file from an I/A Series Application Processors.
- Writing part of a file in an I/A Series Application Processor.

The file access allows the reading and writing of files. The application program access can be done through C language interface calls. Related functions are:

-iarfil
-iawfil
-pfread
-pfwrit

Reading a File from I/A Series Application Processors

The iarfil routine transfers the entire contents of an I/A Series Application Processor file to an I/A Series Open Information Server file. The caller specifies the I/A Series Application Processor name, I/A Series Open Information Server file name, gateway path, user id, group id, and I/A Series Application Processor logical name (as configured in the INI module). An error code is returned.

Writing a File to I/A Series Application Processors

The iawfil routine transfers the entire contents of an I/A Series Open Information Server file to an I/A Series Application Processor file. The caller specifies the I/A Series Application Processor file name, I/A Series Open Information Server file name, gateway path, user id, group id, and I/A Series Application Processor logical name (as configured in the INI module). An error code is returned.

Reading Part of a File from I/A Series Application Processors

The pfread routine transfers a specified portion of an I/A Series Application Processor file into the memory space of the application program. The caller specifies the I/A Series Application Processor file name, gateway path, userid, group id, logical name, the number of bytes to read, and the array in which to store the information.

An error code is returned.

Writing Part of a File in an I/A Series Application Processor

The pfwrit routine transfers data to a specified portion of an I/A Series Application Processor file. The caller specifies the I/A Series Application Processor file name, gateway path, userid, group id, logical name, the number of bytes to write, and the array from which to write the information. An error code is returned.

ACCESSING VIRTUAL TERMINAL SESSIONS

The virtual terminal access allows the use of UNIX features from I/A Series Host application programs using C language function calls. Related function calls are:

-vtinit
-vtprog
-vtcont
-vtend

Establishing a Session

The vtinit function establishes a session by logging into an I/A Series Application Processor. The caller specifies the gateway path, the letterbug of the chosen I/A Series Application Processor, the target account login name (set up by the site system administrator prior to use), account password, and the desired prompt string for the UNIX operating system shell. An error code is returned.

Processing UNIX Command Lines

The vtprog function issues UNIX command lines from a specified program array and returns responses to another specified array. The caller specifies the array containing the UNIX command lines, the number of characters to read from the command line array, and the response storage array. The number of characters stored in the response array and an error code are returned.

The vtcont function has the same arguments as vtprog, but its function is to "continue" the processing of UNIX command lines initiated with vtprog. A return code indicates when "continued" processing is necessary to complete the command(s).

Terminating a Session

The vtend function terminates a session with an I/A Series Application Processor. An error code is returned.

OPTIMIZING THE USE OF VIRTUAL CIRCUITS

By default, AIS functions which communicate to INIs establish and release the X.25 virtual circuit to the INIs each time they are called. This approach allows more applications to make use of the INI services.

If you have an application which makes frequent calls to such functions, you can optimize the application by specifying that virtual circuits are to be established once, kept for as long as needed, then released. Related functions are getcir and relcir.

UTILITIES

SXOPEN Utility Program

The sxopen program opens an AIS data set with input from a file and output of the results to stdout.

AISDIS Utility Program

The aisdis program initially displays the status of configured INI modules and the total number of change messages per INI module. Secondary displays give the name, value, and status for the data objects of a specified data set. Aisdis also displays the number of changes per object either read from or written to a connected I/A Series system.

Aisdis is useful for tuning change and write deltas for data objects. Aisdis is also used to monitor change and write message load on INI modules.

AISTST Utility Program

AISTST exercises and inspects the AIS package. The following is provided:

- The capability to exercise all AIS functions except file access and virtual terminal access.

- The capability to report on the status of the package.

- The capability to turn traces on or off.

FILVAL Utility Program

The filval program updates the iaval.dat file with the latest value type, bad bit, value, status, and read change count for each object.

GWSTAT Utility Program

The gwstat utility program retrieves the status of INI modules and allows the user to connect or disconnect a gateway.

FILUTL Utility Program

The filutil program allows the user to copy:

- a Host file to an I/A Series file
- an I/A Series file to a Host file
- part of an I/A Series file to Host memory
- Host memory to part of an I/A Series file

IA2REM Utility Program

The ia2rem program transfers a file from an I/A Series Application Processor into a file on the Host computer.

REM2IA Utility Program

The rem2ia program transfers a file from the Host computer to an I/A Series Application Processor.

VT Utility Program

The vt program allows the user to enter UNIX command lines to stdin and receive prompts and responses on stdout and stderr.

SET2LOTU Utility Program

The set2lotu program produces a Lotus 1-2-3 (WK1) format worksheet file from a specified AIS data set. The first column contains the data object name, the second column contains the data object value, and the third column contains the data object status. Each worksheet row contains information for one data object.

The resulting worksheet file has as many rows as data objects in the source data set.

NAM2LOTU Utility Program

The nam2lotu program produces a Lotus 1-2-3 (WK1) format worksheet file from a specified list of I/A Series data object names. The first column contains the data object name, the second column contains the data object value, and the third column contains the data object status. Each worksheet row contains information for one data object.

IDX2LOTU Utility Program

The idx2lotu program produces a Lotus 1-2-3 (WK1) format worksheet file from a specified list of AIS data object indexes. The first column contains the data object name, the second column contains the data object value, and the third column contains the data object status. Each worksheet row contains information for one data object.

SETMLOTU Utility Program

The setmlotu program modifies a Lotus 1-2-3 (WK1) format worksheet file with values for set entries. It reads lines of input from stdin. Each line contains a column, row, and user entry. The setmlota program gets the value for the user entries in the set and modifies the column(s)/row(s) in the file.

NAMMLOTU Utility Program

The nammlotu program modifies a Lotus 1-2-3 (WK1) format worksheet file with values for objects specified by name. It reads lines of input from stdin. Each line contains a column, row, and object name. The nammlotu program gets the value for the objects and modifies the column(s)/ row(s) in the file.

IDXMLOTU Utility Program

The idxmlotu program modifies a Lotus 1-2-3 (WK1) format worksheet file with values for objects specified by index. It reads lines of input from stdin. Each line contains a column, row, and object index. Idxmlotu gets the value for the objects and modifies the column(s)/row(s) in the file.

SPECIFICATIONS

This software release provides support for the Sun Microsystems SPARCstation running SunOS Release 4.0 and later.

HOST HARDWARE

Sun Microsystems SPARCstation or SPARCserver.

Bulk Storage

4MB

HOST SOFTWARE

Sun Microsystems SunOS Release 4.1 or later.
Sun Microsystems SunNet x.25.

I/A Series MODULES

Select from one to twelve Information Network Interface (INI) modules.

APPLICATION INTERFACE SOFTWARE (AIS)

SUN AIS Software License works in conjunction with one to twelve Information Network Interface modules configured in connected I/A Series system(s).