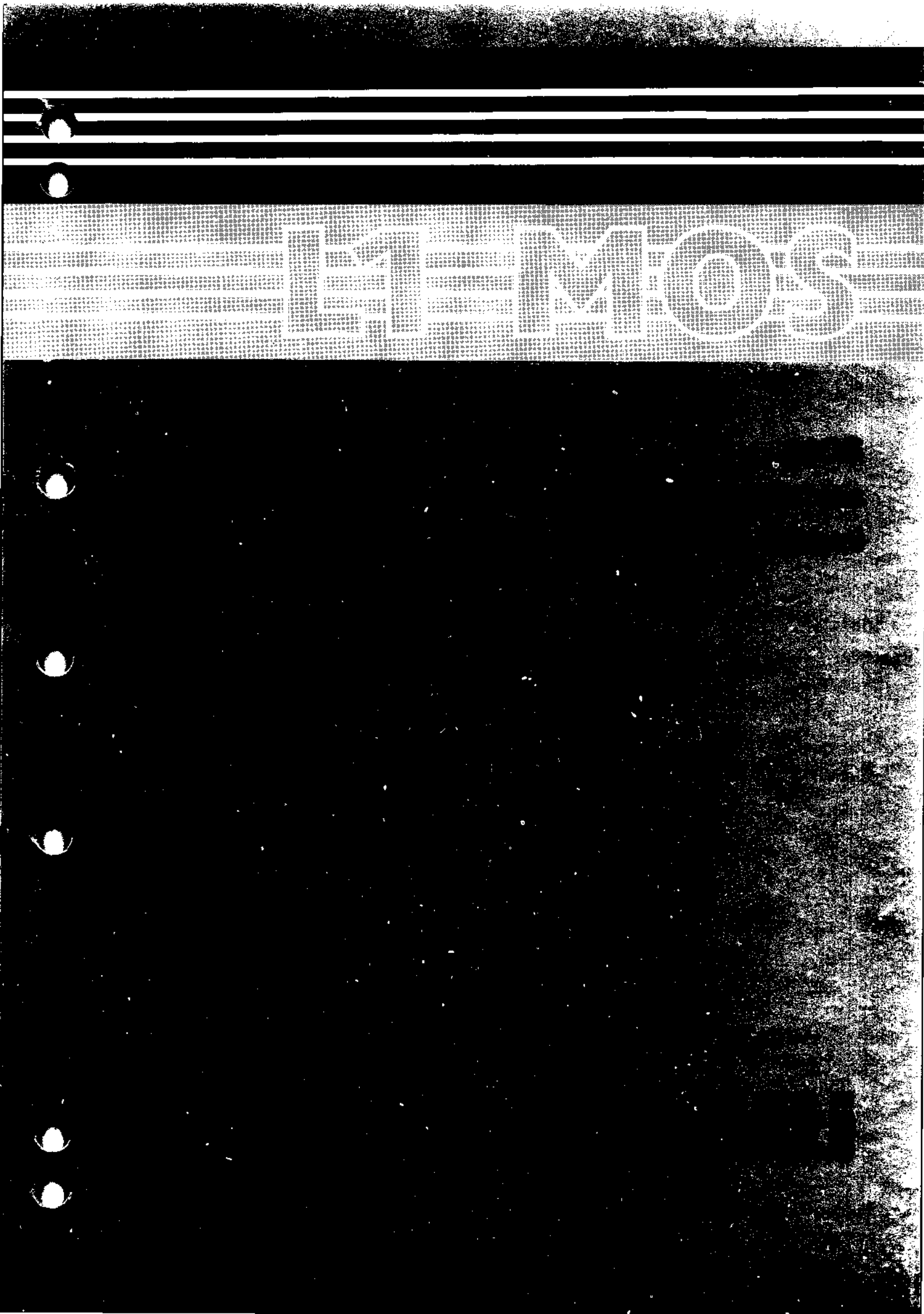

REMOS

L1 MOS

MOS
File System Primitives
Reference Manual

olivetti

PREFACE

This is a reference manual for the primitives which form the external interface of the File System Management (FSM) subsystem of MOS.

It is addressed to systems programmers and system specialists and assumes a prior knowledge of PASCAL+.

SUMMARY

The manual has two chapters. The first chapter contains some general information and an analytical index. The second chapter contains preliminary information on the descriptions of the FSM primitives followed by these descriptions in alphabetical order.

REFERENCES

Read first ...
Introduction to System Programming
Code 4004660 E

For further information ....
PASCAL+ Language - Reference Manual
Code 4002460 R

PASCAL+ Program Preparation and Execution
Code 4002480 T

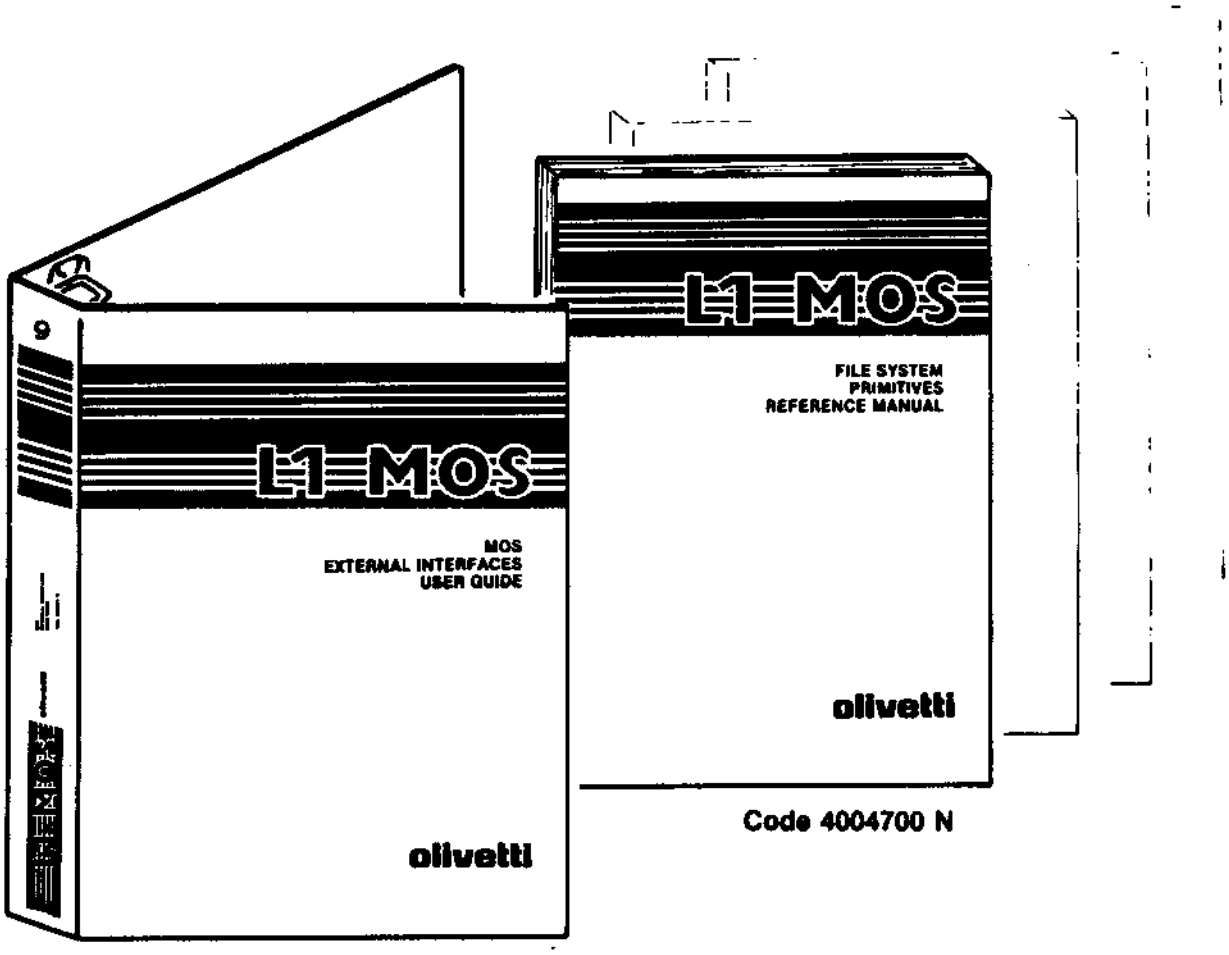
Glossary / Glossario
Code 4002140 H (vol.1)

SHELL Commands Reference Manual
Code 4002770 Q (vol.3)

First Edition: June, 1984
Release 3.0 and 4.0

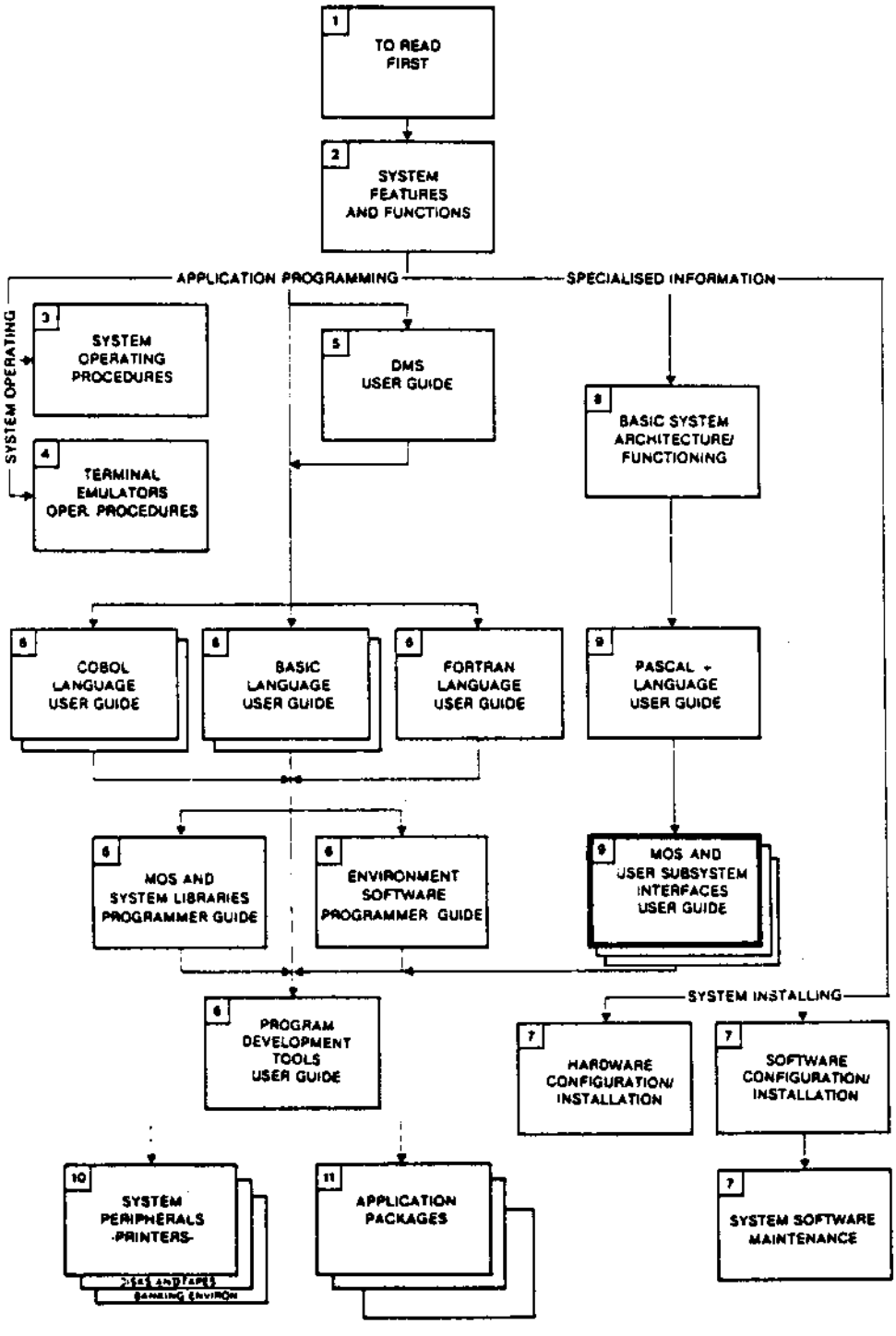
PUBLICATION ISSUED BY:

Ing. C. Olivetti & C., S.p.A.
Direzione Documentazione
77, Via Jervis - 10015 IVREA (Italy)



Code 4004700 N

Code 4004970 W


1
TO READ FIRST
2
SYSTEM FEATURES AND FUNCTIONS
APPLICATION PROGRAMMING
SPECIALISED INFORMATION
SYSTEM OPERATING
3
SYSTEM OPERATING PROCEDURES
4
TERMINAL EMULATORS OPER. PROCEDURES
5
DMS USER GUIDE
8
BASIC SYSTEM ARCHITECTURE/ FUNCTIONING
6
COBOL LANGUAGE USER GUIDE
6
BASIC LANGUAGE USER GUIDE
6
FORTRAN LANGUAGE USER GUIDE
9
PASCAL + LANGUAGE USER GUIDE
6
MOS AND SYSTEM LIBRARIES PROGRAMMER GUIDE
6
ENVIRONMENT SOFTWARE PROGRAMMER GUIDE
9
MOS AND USER SUBSYSTEM INTERFACES USER GUIDE
6
PROGRAM DEVELOPMENT TOOLS USER GUIDE
SYSTEM INSTALLING
7
HARDWARE CONFIGURATION/ INSTALLATION
7
SOFTWARE CONFIGURATION/ INSTALLATION
10
SYSTEM PERIPHERALS -PRINTERS-
11
APPLICATION PACKAGES
7
SYSTEM SOFTWARE MAINTENANCE

The interface between the File System Management (FSM) Subsystem of MOS and application programs is a group of primitives which allow data file handling on any type of peripheral.

There are two distinct types of file access:

- direct access for files on disk, i.e. for:
  - . Byte files
  - . Positional files
  - . Keyed files
- sequential access, for:
  - . Sequential peripherals (seen as sequential files)
  - . Byte files.

The FSM primitives provide these access methods, implement additional concepts such as volume and directory, and also provide other file management and I/O functions.

1. GENERAL INFORMATION

This chapter contains an analytical index of the FSM primitives and some observations of a general nature related to different functional subsets of the primitives.

ANALYTICAL INDEX

Name Handling Functions (module $BST unless otherwise indicated)

Primitives for creating files and directories.

Creating a byte file	- CREATEBYTE
Creating a positional file	- CREATEPOS ($POS module)
Creating a keyed file	- CREATEKEYED ($KEY module)
Creating a sequential file	- CREATESEQ
Creating a directory	- CREATEDIR

Primitive for deleting files.

Deleting a file	- REMOVE

Primitive for changing byte file type.

Changing internal type	- CHANGETYPE

Primitives for adding or removing keyed file indexes.

Adding an index	- ATTACH ($KEY module)
Removing an index	- DETACH ($KEY module)

Primitives for connecting objects.

Connecting a file	- CONNECT
Connecting a file without aliasing	- CONNECTLITERAL
Connecting parent directory	- CONNECTFATHER

Primitives for mounting and unmounting volumes.

Mounting a volume	- MOUNT
Unmounting a volume	- UNMOUNT

Byte Operations (module $BST)

Primitives for opening and closing a file.

File opening	- OPENBYTE
File closing	- CLOSEBYTE

Primitives for reading and writing.

Byte reading	- READBYTE
Byte writing	- WRITEBYTE
Attribute reading	- READINFOBYTE
Attribute writing	- WRITEINFOBYTE

Primitives for reducing file size, or for obtaining the first/last byte address of the file.

Specifying new bounds	- SETBOUNDSBYTE
Finding the first byte	- BEGINBYTE
Finding the last byte	- ENDBYTE

Primitive for computing byte addresses.

Computing a byte address	- COMPUTEBYTE

Positional Operations (module $POS)

Primitives for opening and closing a file.

File opening	- OPENPOS
File closing	- CLOSEPOS

Primitives for reading, writing and deleting.

Record reading	- READPOS
Record writing	- WRITEPOS
Attribute reading	- READINFOPOS
Attribute writing	- WRITEINFOPOS
Record deletion	- DELETEPOS

Primitives for record locking and unlocking.

Locking/Unlocking a record	- LOCKRECORDPOS
Unlocking all records	- UNLOCKALLRECORDS

Primitives for reducing file size, or for obtaining the first/last record of the file.

Specifying new bounds	- SETBOUNDSPOS
Finding the first record	- BEGINPOS
Finding the last record	- ENDPOS

Primitives for computing record addresses.

Computing a record address	- COMPUTEPOS
Transforming a record address.	- TRANSFORMPOS

Keyed Operations (module $KEY)

Primitives for opening and closing a file.

File opening	- OPENKEYED
File closing	- CLOSEKEYED

Primitives for reading, writing and deleting.

Record reading	- READKEYED
Record writing	- WRITEKEYED
Attribute reading	- READINFOKEYED
Attribute writing	- WRITEINFOKEYED
Record deletion	- DELETEKEYED

Primitives for locking a keyed record.

Record locking	- LOCKRECORDKEYED

Primitives for reducing file size, or for obtaining the first/last record address of the file.

Specifying new bounds	- SETBOUNDSKEYED
Finding the first record	- BEGINKEYED
Finding the last record	- ENDKEYED

Primitives for computing record addresses.

Computing a record address	- COMPUTEKEYED
Transforming a key	- TRANSFORMKEYED

Sequential Operations (module $BST)

Primitives for opening and closing a file.

File opening	- OPENSEQ
File closing	- CLOSESEQ

Primitives for reading and writing.

Record reading	- READSEQ
Record writing	- WRITESEQ

Primitives for setting and finding current position in a byte file.

Setting current position	- SEEKP
Finding current position	- SENSEP

General Functions (module $BST unless otherwise indicated)

Primitives for connecting and disconnecting identifiers.

Connecting a new identifier	- CONNECTAGAIN
Disconnecting an identifier	- DISCONNECT
Disconnecting all family's identifiers.	- CLEANUP

Primitives for retrieving and setting information about files.

Getting a local name	- GETNAME
Getting internal file identifiers (pointers)	- GETFILEPTR
Getting record field descriptions	- GETRECORDVIEW ($KEY module)
Getting the file type	- GETTYPE
Getting global root identifier	- GETROOT
Getting user information	- GETXTN
Putting user information	- PUTXTN

Primitives for managing input/output.

Reserving private buffers	- SETBUFFER-OPTIONS
Setting I/O mode	- SETIOOPTIONS

Auxiliary Functions (File Macro Operations) (module $AUX where indicated, otherwise individual modules)

Directory clearing	- CLEARDIR	($AUX)
File compacting	- COMPACT	($AUX)
File converting	- CONVERT	
File and directory copying	- COPY	($AUX)
Alias file creation	- CREATEALIAS	
Getting a device identifier	- GETDEVID	
Getting a pathname	- GETPATHNAME	
Directory listing	- LISTDIR	
Copying a bootstrapper	- MAKEBOOT	
Creating an environment	- MAKESTD24	

Creating a volume	- MAKEVOLUME	
Prying for information	- PRY	
Renaming an object	- RENAME	($AUX)

OBSERVATIONS

Operations on Files

Byte operations are associated with working on files seen as an unstructured stream of bytes and are intended for use on byte files. All byte operations can be carried out on positional and keyed files, but care must be taken as the support files are not updated and an inconsistent database may result.

Positional operations are associated with working on files seen as a numbered set of records that is, on direct access files with a record structure. Positional operations can therefore be carried out on keyed files as well as positional files but care must be taken as the indexes are not updated.

Keyed operations are associated with working on files via keys. All keyed operations are carried out exclusively on keyed files.

Sequential operations are used to access data in its order of entry. Most sequential operations can be carried out on sequential and byte files, but the current position primitives are relevant only to byte files.

Shell Commands

Most name-handling primitives (except the Connect functions) and most auxiliary primitives (except GetDevId and GetPathName) correspond directly to Shell commands:

Shell:	FSM:
	Name handling primitive:
CHTYPE	ChangeType
DELINDEX	Detach
MKBST	CreateByte
MKDIR	CreateDir
MKINDEX	Attach
MKKEYED	CreateKeyed
MKPOS	CreatePos
MNT	Mount
REMOVE	Remove
UNMNT	Unmount

	Auxiliary primitive:
CLEARDIR	ClearDir
COMPACT	Compact
CONVERT	Convert
COPY	Copy
MKALIAS	CreateAlias
MKBOOT	MakeBoot
MKENV	MakeStd24
MKVOL	MakeVolume
PRY	Pry
RENAME	Rename
SHDIR	ListDir

2. PRIMITIVE DESCRIPTIONS

This chapter contains descriptions of all the File System Management primitives, in alphabetical order, preceded by a general explanation of the layout of these descriptions and lists of PASCAL+ types for reference.

Layout

Each description contains:

- a statement of the function
- a schematic view, in which the parameters are shown according to the following rules:
  - The parameter preceding the primitive name is returned by the primitive.
  - The parameters following the primitive name are input parameters or the output ones whose addresses must be supplied to the primitive. (The parameters enclosed between '*' are variables).
- a table of the parameters
- a table of the diagnostics (if any)
- a list of characteristics (if any)
- a list of related primitives (if any)
- an example of calling the primitive.

Note

Pathnames in Release 4 onwards have no limitation on their length (up to and including Release 3, max. length = 60 characters).

Examples of Calling Primitives

In the examples:

- 'systypes.i' contains all system types; 'fstypes.i' contains all file system types and 'AMS.d' the definition of all the primitives.
- The comment explains how to invoke a primitive if only the 'systypes.i' module is included.

Type Definitions

The system types are shown below and a listing of the PASCAL+ definitions of the file system types is then provided.

T_systemId,T_id — These are internal identifiers of file system objects.

T_address — This type defines addresses which cannot be interpreted by the user.

T_reply — These are the classes of error reply, see Error Types below.

```
                    **********************
                    *                    *
                    * FILE SYSTEM TYPES  *
                    *                    *
                    **********************

T_accessRights = (nill_, r_, a_, ra_, w_, rw_, aw_,
                  raw_, x_, rx_, ax_, rax_, wx_, rwx_,
                  awx_, rawx_)

T_blkdev = (vdisk_, flop_, mflop_, hdisk_, sc_, null_);

T_bounds = (bof_, eof_);

T_bstf = (data_, dir_, vol_, iotable_, alias_, program_)

T_clearDir = (clearsubdir_, contclearsubdir_);

T_convert = (byte_, positional_, keyed_);

T_copyOptions = (replace_, dontreplace_, append_);

T_exactness = (eq_, ge_, gt_, next_, previous_);

T_existAction = (rename_, abort_);

T_field = packed record
              fieldNumber : integer;
              fieldCoords : T_fieldCoords;
                fieldKind : T_fieldKind
          end;

T_fieldCoords = packed record
                    start : integer;
                    length: integer
                end;

T_fieldKind = (pm_, su_, sd_);

T_file = record
           case fileType : T_sys of
                  bst_ : (bs_ : T_bstf);
                  pos_ : (ps_ : T_posf);
                  key_ : (ky_ : T_keyf);
                blkDev_ : (bk_ : T_blkDev);
                seqDev_ : (sq_ : T_seqdev);
           end;

T_iotype = (async_, sync_, directwv_, directw_, directv_,
            unchanged_);

T_keyf = (keyin_, keyex_);

T_listDir = (listsubdir_, dontlistsubdir_, listonly_);

T_lockLevel = (n_, e_, m_);
```

```
T_pageSize = (p512_, p1024_, p2048_, p4096_);

T_posf = posyd_, posnd_, posydp_);

T_readInfo = packed record
                 allocationUnit : longinteger;
                     createTime : longinteger;
                     lastaccess : longinteger;
                       userType : integer
             end;

T_readInfoK = packed record
                     readInfo : T_readInfoP;
                 splitStrategy : T_splitStrategy
              end;

T_readInfoP = packed record
                     readInfo : T_readInfo;
                  recordLength : integer
              end;

T_recordDeletion = (yd_, nd_, ydp_);

T_seqDev = (ct_, op_, fr_);

T_splitStrategy = (s50_, s90_, s100_);

T_sys = (private_, pmmp_, bet_, pos_, key_, blkDev_,
         seqDev_, wstp_, frtp_, loctp_);

T_timeout = 0..15;

T_volumeInfo = packed record
                        volSize : longinteger;
                  releaseNumber : integer;
                   releaseLevel : integer;
                       nrFiles : integer;
                      userCode : integer
               end;

T_writeInfo = packed record
                  writeAllocationUnit : boolean;
                      writeUserType : boolean;
                    allocationUnit : longinteger;
                          userType : integer
              end;

T_writeInfoK = packed record
                          writeInfo : T_writeInfo;
                  writeSplitStrategy : boolean;
                       splitStrategy : T_splitStrategy
               end;

T_writeMode = (rewrite_, newwrite_, alwayswrite_);
```

Error Types

A listing of the PASCAL+ definitions of the error types which concern the File System Management and the error class type, T_reply, is given below.

```
T_class = (CORRECT,SYSTEM,DEBUG,PMMWARNING,PMMERROR,
           FSWARNING,FSERROR,WSWARNING,WSERROR,RUNTIME,
           USRCLASS,LMWARNING,LMERROR,THWARNING,THERROR,
           FRWARNING,FRERROR);

T_correctCode = (OKAY);

T_sysErrCode = (TERMINALDOWN,DIAGNOSTICERROR,RECEIVEERROR,
                HARDWAREERROR,SYSTEMERROR,TABLEOVERFLOW,
                OUTOFDISKSPACE,FATALERROR,DEFAULT,KERNSEGFAULT,
                SEGLENFAULT,INHIBSEGFAULT,XQTFAULT,INVALIDINST,
                PRIVILEGEDINST,FATALSEGFAULT,NOPROCEDURE,NOROUTING,
                RESULTSDOUBTFUL,OPERATIONABORTED);

T_dbgErrCode = (RECORDTOOLONG,INVALIDOPERATION,INVALIDID,
                INVALIDPARAMETERS,ERRORINTHESTRING,SECURITYVIOLATION,
                NOTYETIMPLEMENTED,BADSCOPE,BADSTATE,ILLEGINDEX,
                CHANDESTROY,CHANBUSY,CHILDALIVE,NEOBJECT,NOWRITERS);

T_fsWarnCode = (BYTESTREAMOPERATION,POSITIONALOPERATION,
                INCONSISTENTDATABASE,NOPRIMARYINDEX,DUPLICATEDKEY,
                INCOMPLETERECORD);

T_fsErrCode  = (TIMEOUT,RECORDLOCKED,DEVICENOTREADY,INVALIDPATHNAME,
                NAMEALREADYEXISTS,NAMENOTFOUND,NOTEMPTYDIRECTORY,
                ENDOFFILE,OUTOFBOUNDS,RECORDNOTFOUND,
                RECORDALREADYEXISTS,KEYNOTFOUND,KEYALREADYEXISTS);

T_reply = record
            case class  : T_class of
             CORRECT    : (correctCode : T_correctCode);
             SYSTEM     : (sysErrCode  : T_sysErrCode);
             DEBUG      : (dbgErrCode  : T_dbgErrCode);
             PMMWARNING : (pmmWarnCode : T_pmmWarnCode);
             PMMERROR   : (pmmErrCode  : T_pmmErrCode);
             FSWARNING  : (fsWarnCode  : T_fsWarnCode);
             FSERROR    : (fsErrCode   : T_fsErrCode);
             WSWARNING  : (wsWarnCode  : T_wsWarnCode);
             WSERROR    : (wsErrCode   : T_wsErrCode);
             RUNTIME    : (rtCode      : T_rtCode);
             USRCLASS   : (usrCode     : T_usrCode);
             LMWARNING  : (lmWarnCode  : T_lmWarnCode);
             LMERROR    : (lmErrCode   : T_lmErrCode);
             THWARNING  : (thWarnCode  : T_thWarnCode);
             THERROR    : (thErrCode   : T_thErrCode);
             FRWARNING  : (frWarnCode  : T_frWarnCode);
             FRERROR    : (frErrCode   : T_frErrCode);
          end(T_reply);
```

FS Modules

Primitive definition modules corresponding to the individual File System Management modules are available for inclusion in programs in addition to the general module AMS.d. Thus the use of AMS.d can be avoided in programs where only a subset of primitives is required, if there is a risk of Symbol Table overflow. The list of primitive definition modules and the corresponding FSM modules is given below.

Primitive definition module	FSM module(s)
AMS.d	All
AMSBS.d	$BST (Byte and sequential operations and byte-level name-handling and general functions).
AMSP.d	$POS (Positional operations plus one name-handling function)
AMSK.d	$KEY (Keyed operations plus three name-handling functions and one general function)
AUX.d	$AUX (FSM Macros ClearDir, Compact, Copy and Rename).

Note

The remaining FSM macros are separate modules.

This function allows a new index to be associated with a keyed file.

reply
Attach
parentDirId,*fileName*,*field*,pageSize,splitStrategy

Parameter	I/O	System-Defined Type Name	Type	Description
parentDirId	I	T_systemId		System identifier of the directory within which 'fileName' can be found.
fileName	I		packed array [min..max: integer] of char	This parameter can be a sequence of names which already exist, which are separated by '/' and terminated with ' '. The length of each intermediate component must not exceed 14 characters, and the last one only 12. The sequence does not begin with '/'

field	I/O	T_field	packed record	This parameter contains the following information:
fieldNumber	O		integer	sequence number of the created index
fieldCoords	I	T_field-Coords	packed record	coordinates of the key:
start			integer	- beginning of the key
length			integer	- length of the key.
fieldKind	I	T_field-Kind	enumerated	Index type : su_:secondary index with unique keys sd_:secondary index with duplicated keyes pm_:primary index.
pageSize	I	T_page Size	enumerated	This parameter must contain one of the following values: p512_, p1024_, p2048_, p4096_. See 'CreateKeyed'.
splitStrategy	I	T_split-Strategy	enumerated	This parameter must contain one of the following values: s50_, s90_, s100_ See 'CreateKeyed'.
reply	O	T_reply	variant record	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	No more indexes can be added (six already exist).
	OUTOFDISKSPACE	No more space on disk.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDOPERATION	Invalid operation.
	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not available.
	INVALIDPATHNAME	Invalid pathname.
	NAMENOTFOUND	File name does not exist.

Characteristics

1 The file must possess at least one index in order to be considered a keyed file.

2 This primitive can be used to create a primary index.If the keys are external,the file must be empty.

3 Up to five secondary indexes can be associated with a keyed file (and one primary index).

4 It is possible to modify the splitStrategy parameter via WriteInfoKeyed.

5 The current view of the records in the file may be obtained by using the GetRecordView primitive.

6 The user must specify the parameter ' fieldNumber ' to access the keyed file's records using a particular index.

Note

See the following primitives:

- WriteInfoKeyed
- GetRecordView

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var  parentDirId    : T_systemId;
     fileName       : packed array [ 1..17 ] of char;
     field          : T_field;
     pageSize       : T_pageSize;
     splitStrategy  : T_splitStrategy;
     fieldNumber    : integer;
     reply          : T_reply;


(* type

   T_fieldCoords = packed record
                       start : integer;
                      length : integer
                   end;

   T_fieldKind = ( pm_, su_, sd_ );

   T_field = packed record
                 fieldNumber : integer
                 fieldCoords : T_fieldCoords;
                   fieldKind : T_fieldKind
             end;

   T_pageSize = ( p512_, p1024_, p2048_, p4096_ );

   T_splitStrategy = ( s50_, s90_, s100_ );

   function Attach(     parentDirId : T_systemId;
                           fileName : packed array [ min..max :
                                                integer ] of char;
                    var         field : T_field;
                               pageSize : T_pageSize;
                      splitStrategy : T_splitStrategy) : T_reply;
   external; *)

begin
 .
 .
 fileName:='usr/franco/carlo ';
 field.fieldCoords.start:=8;
 field.fieldCoord.length:=8;
 field.fieldKind:=su_;
 reply:=Attach(parentDirId,fileName,field,p512_,s50_);
 fieldNumber:=field.fieldNumber;
 .
end.
```

This function returns the byte offset of the first valid byte of the file indicated by the system identifer.
The primitive does not return any error code.

```
posByte
BeginByte
systemId,*startAddress*
```

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
startAddress	O	T_address		Address of the first valid byte in the file. This value cannot be interpreted by the user.
posByte	O		longinteger	Byte offset of the first valid byte in the file.

Characteristics

1 The byte offset returned by the primitive may be used as input to ReadByte and WriteByte.

2 The returned address can be an input for ComputeByte and SetBoundsByte.

3 It is not necessary to open the file before invoking this primitive.

Note

See the following primitives:

- ReadByte
- ComputeByte
- SetBoundsByte
- WriteByte

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'APS.d'

var  systemId     : T_systemId;
     startAddress : T_address;
     posbyte      : longinteger;

(* type

   dummyRec = record
              end;

   T_address = ^dummyRec;

   function BeginByte(        systemId : T-systemId;
                      var startAddress : T_address) : longinteger;

   external; *)


begin
 .
 .
 posByte:=BeginByte(systemId,startAddress);
 .
 .
end.
```

This function returns the position in a specified file of the record corresponding to the smallest key which is present in a particular index. The file is identified by the system identifier and by the index sequence number. In output,. the primitive also returns the key and its address within the index.

posRecord
BeginKeyed
systemId,fieldNumber,*beginAddress*,*beginKey*,keyStartIndex

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
fieldNumber	I		integer	Integer indicating index sequence number.
beginAddress	O	T_address		Address of the key within the specified index. This value cannot be interpreted by the user.

beginKey	0	packed array [min..max: integer] of dataItem	Array containing the record key. 'dataItem'is defined by the user and its length must be of 8 bits, e.g. char, byte (-128...127).
keyStartIndex	I	integer	Integer within the "beginKey" range. The key will be transferred into "beginKey" starting from the position specified by this parameter.
posRecord	0	longinteger	Record position.

Characteristics

1 The record selected in this way can be read with the ReadPos primitive.

2 The address can be used in input for ComputeKeyed or SetBoundsKeyed.

3 It is not necessary to open the file before invoking this primitive.

Note

See the following primitives:

- ReadPos
- ComputeKeyed
- SetBoundsKeyed

xample

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'APS.d'

var   systemId                      : T_systemId;
      fieldNumber,keyStartIndex     : integer;
      beginAddress                  : T_address;
      beginKey                      : packed array [ 1..16 ] of dataItem;
      posRecord                     : longinteger;

(* type

   dataItem = char;

   dummyRec = record
              end;

   T_address = ^dummyRec;

   function beginKeyed(        systemId  : T_systemId;
                           fieldNumber : integer;
                     var beginAddress : T_address;
                     var      beginKey : packed array [ min..max :
                                           integer ] of dataItem;
                        keyStartIndex : integer) : longinteger;
   external; *)

begin
 .
 .
 posRecord:=BeginKeyed(systemId,0,beginAddress,beginKey,1);
 .
 .
end.
```

This function returns the position and the address of the first valid record of the file identified by the system identifier. The primitive does not return any error.

```
start
BeginPos
systemId,*startAddress*
```

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
startAddress	O	T_address		Address of the first valid record. This address cannot be interpreted by the user.
start	O		long-integer	Record position.

Characteristics

1 The position of the record returned by the primitive can be used as input to ReadPos.

2 The address of the record can be used in input for ComputePos and SetBoundsPos.

3 It is not necessary to open the file before invoking this primitive.

Note

See the following primitives:

- ReadPos
- ComputePos
- SetBoundsPos

Example

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var   systemId     : T_systemId;
      startAddress : T_address;
      start        : longinteger;

(* type

   dummyRec = record
              end;

   T_address = ^dummyRec;

   function BeginPos(         systemId : T_systemId;
                      var startAddress : T_address) : longinteger;
   external; *)

begin
 .
 .
 start:=BeginPos(systemId,startAddress);
 .
 .
end.
```

This function changes the internal type of the byte file specified by the file name.

reply
ChangeType
parentDirId,*fileName*,fileOwner,newType

Parameter	I/O	System-Defined Type Name	Type	Descrition
parentDirId	I	T_systemId		Identifier of directory containing "fileName".
fileName	I		packed array [min..max: integer] of char	This parameter may be a sequence of existing names, separated by a "/" It must not begin with a "/" and must end with a ' ' or null. Each component may have a maximum of 14 characters.
reserved	I		long-integer	Reserved for future use.

newType	I	T_bstf	enumerated	Specifies the required file type to be recognised by the system. See characteristic 2.
reply	O	T_reply	variant record	Diagnostics.

Diagnostics — The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table overflow.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.
	INVALIDOPERATION	Invalid operation.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not ready.
	INVALIDPATHNAME	Invalid pathname.

Characteristics

1 The primitive will not work (INVALIDOPERATION) if the file is open.

2 The possible values for the parameter "newType" are:

- data_ : data file
- dir_ : directory
- vol_ : volume
- iotable_ : program context
- alias_ : alias file
- program_ : program directory
- loadable_ : program

3 Any changes to the structure of the data within the file to make it conform to the new type should be made explicitly before calling this primitive, as the only function performed by the primitive is to change the type field in the file descriptor.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'ARS.d'

var  parentDirId : T_systemId;
       fileName : packed array [1..20] of dataItem;
       reserved : longinteger;
        newType : T_bstf;
          reply : T_reply

(*  type

    dataItem = char;

    T_bstf = (data_,dir_,vol_,iotable_,alias_,program_);

    function ChangeType( parentDirId : T_systemId;
                         var fileName : packed array [min..max : integer]
                                        of dataItem;
                             reserved : longinteger;
                              newType : T_bstf) : T_reply;
    external; *)

begin
 .
 .
 newType:=dir_;
 fileName:='pippo ';
 reply:=ChangeType(parentDirId,,fileName,0,newType);
 .
 .
end.
```

CLEANUP (DISCONNECT ALL FAMILY'S FILES)

This function closes and disconnects all files associated with the family whose system identifier is given in input.

reply
Cleanup
systemId

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		Family identifier.
reply	O	T_reply	variant record	Diagnostics.

Diagnostics — The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation correctly executed.
SYSTEM	SYSTEMERROR	System error.
DEBUG	INVALIDID	Invalid identifier.

Characteristic	This function is intended for system use when a family is destroyed; it ensures that system table entries for files associated with the family are cleared.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var  systemId : T_systemId;
     reply    : T_reply;

(* function CleanUp(var systemId : T_systemId) : T_reply;

   external; *)

begin
 .
 .
 reply:=CleanUp(systemId);
 .
 .
end.
```

This function removes the contents of the directory specified by the directory name.

reply
ClearDir
parentDirId,*dirName*,clearSubDir

Parameter	I/O	System-Defined Type Name	Type	Description
parentDir	I	T-SystemId		Identifier of directory containing the directory to be cleared.
dirName	I		packed array [min..max: integer] of char	This parameter may be a sequence of existing names, separated by a "/". It must not begin with a"/" and must end with a ' ' or null. Each component may have a maximum of 14 characters.
clearSubDir	I	T_clearDir	enumerated	This parameter specifies whether all non-empty subdirectories are removed recursively or not - clearsubdir - dontclearsubdir_

reply	0	T-reply	variant record	Diagnostics.

Diagnostics — The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table overflow.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDOPERATION	Invalid operation.
	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not ready.
	INVALIDPATHNAME	Invalid pathname.
	NAMENOTFOUND	Directory name does not exist.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var   parentDirId  : T_systemId;
        dirName    : packed array [1..20] of dataItem;
      clearSubDir  : T_clearDir;
            reply  : T_reply;

(*  type

    T_clearDir = (clearsubdir_, dontclearsubdir_);

    function ClearDir(     parentDirId : T_systemId;
                       var   dirName : packed array [min..max : integer]
                                       of dataItem;
                         clearSubDir : T_clearDir) : T_reply;

    external; *)

begin
 .
 .
 dirName:='pippo ';
 reply:=ClearDir(parentDirId,dirName,clearsubdir_);
 .
 .
end.
```

This function closes the file specified by the system identifier, releases all the existing locks on the file, and updates the last access and modify times.

```
reply
CloseByte
*systemId*
```

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I/O	T_systemId		Identifier of file to be closed.
reply	O	T_reply	variant record	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.

Characteristics

1 On completion of this primitive, no I/O operation can be requested on the file. It is necessary to reopen the file for future operations.

2 The identifier given in input must be a session-id; in output the associated connect-id is returned.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'APS.d'

var  systemId : T_systemId;
     reply    : T_reply;

(* function CloseByte(var systemId : T_systemId) : T_reply;

   external; *)


begin
 .
 .
 reply:=CloseByte(systemId);
 .
 .
end.
```

This function closes the keyed file specified by the system identifier releases all the existing locks on the file, and updates the last access and modify times.

```
reply
CloseKeyed
*systemId*
```

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I/O	T_systemId		Identifier of the file to be closed.
reply	O	T_reply	variant record	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.

Characteristics

1 On completion of this primitive, no I/O operation can be executed on the file. Closing a file will ensure that all data written to it will be written to the disk. The user must reopen the file for future operations.

2 The identifier given in input must be an I/O session-id; in output the associated connect-id is returned.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var  systemId : T_systemId;
     reply    : T_reply;


(* function CloseKeyed(var systemId : T_systemId) : T_reply;

   external; *)


begin
 .
 .
 reply:=CloseKeyed(systemId);
 .
 .
end.
```

CLOSEPOS (CLOSE A POSITIONAL FILE)

This function closes the positional file specified by the system identifier releases all the existing locks on the file, and updates the last access and modify times.

```
reply
ClosePos
*systemId*
```

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I/O	T_systemId		Identifier of the file to be closed.
reply	O	T_reply	variant record	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.

Characteristics

1 On completion of this primitive, no I/O operation can be requested on the file. Closing a file will ensure that all data written to it will be written to the disk. It is necessary to reopen the file for future operations.

2 The identifier given in input must be an I/O session-id; in output the associated connect-id is returned.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var  systemId : T_systemId;
     reply    : T_reply;

(* function ClosePos(var systemId : T_systemId) : T_reply;

   external; *)

begin
 .
 .
 reply:=ClosePos(systemId);
 .
 .
end.
```

reply
CloseSeq
systemId

This function closes the sequential file specified by the system identifier, releases all the existing locks on the file, and updates the last access and modify times. A sequential file corresponds to a sequential peripheral.

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I/O	T_systemId		Identifier of the file to be closed.
reply	O	T_reply	variant record	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.

Characteristics

1 On completion of CloseSeq, no I/O operation is allowed. It is necessary to reopen the file for future operations.

2 The identifier given in input must be an I/O session-id; in output the associated connect-id is returned.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined
#include 'fstypes.i'
#include 'APS.d'

var  systemId : T_systemId;
     reply    : T_reply;

(* function CloseSec(var systemId : T_systemId) : T_reply;

   external; *)

begin
 .
 .
 reply:=CloseSeq(systemId);
 .
 .
end.
```

reply
Compact
parentDirId, *fileName*

This function compacts the file specified by the file name.

Parameter	I/O	System-Defined Type Name	Type	Description
parentDirId	I	T_systemId		Identifier of the directory containing the file name.
fileName	I		packed array [min..max: integer] of char	This parameter may be a sequence of existing names, separated by a "/" It must not begin with a "/" and must end with a ' ' or null. Each component may have a maximum of 14 characters.
reply	O	T_reply	variant record	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	OUTOFDISKSPACE	Not enough space left on disk.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDOPERATION	Operation not permitted.
	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not available.
	INVALIDPATHNAME	Invalid pathname.
	NAMENOTFOUND	File name does not exist.

Characteristics

1 Compacting a file results in a single file extent.

2 Compacting a packed positional or a keyed file also removes logically deleted (keys and) records.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'APS.d'

var   parentDirId : T_systemId;
        fileName : packed array [1..20] of char;
           reply : T_reply;

(* function Compact(parentDirId : T_systemId;
                    fileName : packed array [min..max : integer]
                                 of char) : T_reply;
   external; *)

begin
 .
 .
 fileName:='pippo
 reply:=Compact(parentDirId,fileName);
 .
 .
 end.
```

COMPUTEBYTE (COMPUTE A BYTE OFFSET ADDRESS)

This function computes on address within the file identified by the system identifier. The primitive does not return an error.

```
ret
ComputeByte
systemId,presentAddress,nrByte,*nextAddress*
```

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
present-Address	I	T_address		Address within the file. This value is returned by EndByte or BeginByte or ComputeByte.
nrBytes	I		long-integer	Numerical value which indicates how many bytes 'nextAddress' is from 'present-Address'.
nextAddress	O	T_address		Address of the requested byte offset.

ret	0	long-integer	Byte offset.

Characteristics

1 The primitive computes the byte offset address required, on the basis of the address provided in input, which has been increased/decreased according to the number of bytes specified by the 'nrByte' parameter.
The addresses cannot be interpreted by the user.

2 The address returned by the primitive can only be used subsequently in input by ComputeByte itself or by SetBoundsByte. The byte position returned can be used in input by ReadByte.

3 The main use of this primitive is in conjunction with SetBoundsByte in order to empty the file.

4 It is not necessary to open the file before invoking this primitive.

Note

See the following primitives:

- BeginByte
- EndByte
- SetBoundsByte
- ReadByte

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var  systemId                       : T_systemId;
     presentAddress,nextAddress     : T_address;
     nrBytes,ret                    : longinteger;


(* type

   dummyRec = record
              end;

   T_address = ^dummyRec;

   function ComputeByte(          systemId : T_systemId;
                          presentAddress : T_address;
                                  nrBytes : longinteger;
                      var    nextAddress : T_address) : longinteger;
   external; *)


begin
 .
 .
 ret:=ComputeByte(systemId,presentAddress,T,nextAddress);
 .
 .
end.
```

This function computes the address within an index, the position and the key of a record within a keyed file. The file is identified by the system identifier and the index by its sequence number.
The primitive does not return any error code.

nextRecord
ComputeKeyed
systemId,fieldNumber,presentAddress,nrKeys,*nextAddress*,*nextKey*,
keyStartIndex

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
fieldNumber	I		integer	Integer indicating index sequence number.
presentAddress	I	T_address		Known address within the index.
nrKeys	I		long-integer	Numerical value which indicates how many keys 'nextAddress' is from 'present-Address'.

nextAddress	O	T_address		Address in index of the requested key. It can be used in input by a subsequent ComputeKeyed or SetBoundsKeyed.
nextKey	O		packed array [min...max: integer] of dataItem	Array containing the requested key. 'dataItem' is defined by the user and its length must be of 8 bits, e.g. char, byte (-128...127).
keyStartIndex	I		integer	Integer within the 'nextKey' range. The key will be transferred into 'nextKey' starting from the position specified by this parameter.
nextRecord	O		long-integer	Value indicating the position of the record corresponding to the returned key. This value may only be used in input by ReadPos.

Characteristics

1 The output address is computed on the basis of an address returned by either TransformKeyed or ComputeKeyed or BeginKeyed or EndKeyed, which has been increased/decreased according to the 'nrKeys' parameter. These addresses cannot be interpreted by the user.

2 The address thus obtained can be subsequently used in input by ComputeKeyed or by SetBoundsKeyed.

3 It is possible to read the record via ReadPos. By alternating ComputeKeyed with ReadPos, it is possible to read the file sequentially.

4 The record position should not be used in input to WritePos or DeletePos, as indexes will not be updated and inconsistency could result.

5 This primitive in conjunction with SetBoundsKeyed empties the file completely.

6 It is not necessary to open the file before invoking this primitive.

Note

See the following primitives:

- SetBoundsKeyed
- TransformKeyed
- EndKeyed
- BeginKeyed
- ReadPos

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var   systemId                     : T_systemId;
      fieldNumber,keyStartIndex    : integer;
      presentAddress,nextAddress   : T_address;
      nrKeys,nextRecord            : longinteger;
      nextKey                      : packed array [ 1..16 ] of dataItem

(* type

   dataItem = char;

   dummyRec = record
              end;

   T_address = ^dummyRec;

   function ComputeKeyed(          systemId : T_systemId;
                                fieldNumber : integer;
                             presentAddress : T_address;
                                    nrKeys : longinteger;
                           var  nextAddress : T_address;
                           var      nextKey : packed array [ min..ma
                                              integer ] of dataItem
                              keyStartIndex : integer ) : longinteger
   external; *)


begin
 .
 .
 nextRecord:=ComputeKeyed(systemId,C,presentAddress,I,nextAddress,
                                                      nextkey,1);
 .
 .
end.
```

COMPUTEPOS
(COMPUTE THE ADDRESS OF A RECORD IN A POSITIONAL FILE)

This function computes the address and the position of a record within the file identified by the system identifier.
The primitive does not return an error.

nextRecord
ComputePos
systemId,presentAddress,nrRecords,*nextAddress*

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
presentAddress	I	T_address		Record address within the file.
nrRecords	I		long-integer	Numerical value which indicates how many records 'nextAddress' is from 'present-Address'.
nextAddress	O	T_address		Record address corresponding to 'presentAddress' +/- nrRecords.

nextRecord	0	long-integer	The position of the record corresponding to the new address.

Characteristics

1 The primitive computes the address of a record, as the address provided in input (calculated by TransformPos, EndPos, ComputePos or BeginPos) increased or decreased according to the number of records indicated by 'nrRecords'. The addresses cannot be interpreted by the user.

2 The returned address may subsequently be used in input to ComputePos or to SetBoundsPos.

3 By alternating this primitive with ReadPos, it is possible to read the file sequentially.If the file does not contain any deleted records and all its records are contiguous, it can be read sequentially using the ReadPos primitive alone.

4 This primitive in conjunction with SetBoundsPos empties the file completely.

5 It is not necessary to open the file before invoking this primitive.

Note

See the following primitives:

- SetBoundsPos
- TransformPos
- EndPos
- BeginPos
- ReadPos

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var  systemId                       : T_systemId;
     presentAddress,nextAddress     : T_address;
     nrRecords,nextRecord           : longinteger;

(* type

   dummyRec = record
              end;

   T_address = ^dummyRec;

   function ComputePos(          systemId : T_systemId;
                          presentAddress : T_address;
                              nrRecords : longinteger;
                       var    nextAddress : T_address) : longinteger;
   external; *)


begin
 .
 .
 nextRecord:=ComputePos(systemId,presentAddress,3,nextAddress);
 .
 .
end.
```

CONNECT (CONNECT AN OBJECT)

This function associates a system identifier with a named object.
The identifier returned must be provided in subsequent operations
on that particular object until a Disconnect is issued.
The named object is specified in input by the name objectName which
is relative to a reference structure specified by referenceId.

The reference structure can be a standard file system directory, a
program context or any generic reference structure. The nature of
the pathname in objectName depends on the reference structure.
If this is a program context, the first component of objectName
is a name local to the program.

reply
Connect
referenceId,*objectName*,accessRights,*systemId*,*userType*

Parameter	I/O	System-Defined Type Name	Type	Description
referenceId	I	T_systemId		System identifier. This indicates the structure within which the 'objectName' can be found.

objectName	I		packed array min.. [max integer] of char	This parameter can be a sequence of names separated by '/'. It must not begin with '/' and it must terminate with a ' '. This parameter could also be the name of an alias.

accessRights	1	T_accessRights	enumerated	This parameter indicates the operations allowed on the file. It may be nill_ or any combination of the following basic values: r_: available for reading (file) or listing contents (directory) a_: it is possible to append data but it is not possible to modify or remove existing data (for file) or it is possible to create files (for a directory). w_: it is possible to modify data without any restrictions (file) or it is possible to create and remove files (directory) x_: it is possible to execute it (if a file)or connect files within it (if a directory).

systemId	0	T_systemId		Identifier associated with 'fileName'. It is returned by the system.
userType	0		integer	Integer returned by the system. It corresponds to that provided by the user at create time. It is not interpreted by the FS.
reply	0	T_reply	variant record	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System Error.
	TABLEOVERFLOW	System Table Overflow.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not available.
	INVALIDPATHNAME	Invalid pathname.
	NAMENOTFOUND	Name provided does not exist.

Characteristics

1 A 'Connect' to a directory or to a volume can only have "accessRights :=r_".

2 The input referenceId must be a connect-id not a session-id.

3 If objectName specifies an alias file, the aliased file will be connected.

4 Failure may occur if you try to connect a file within a volume that is already open.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var  referanceId,systemId : T_systemId;
     objectName           : packed array [ 1..6 ] of char;
     accessRights         : T_accessRights;
     userType             : integer;
     reply                : T_reply;

(* type

   T_accessRights = (nill_, r_, a_, ra_, w_, rw_, aw_, raw_, x_, rx_,
                     ax_, rax_, wx_, rwx_, awx_, rawx_);

   function Connect(       referenceId : T_systemId;
                    var     objectName : packed array [ min..max :
                                              integer ] of char;
                           accessRights : T_accessRights;
                    var        systemId : T_systemId;
                    var        userType : integer) : T_reply;
   external; *)


begin
 .
 .
 objectName:='pippo ';
 reply:=Connect(parentDirId,objectName,w_,systemId,userType);
 .
 .
end.
```

This function obtains a copy of the system identifier associated with an already connected object.

```
reply
ConnectAgain
inputId, *outputId*
```

Parameter	I/O	System-Defined Type Name	Type	Description
inputId	I	T_systemId		Identifier of the already connected object.
outputId	O	T_systemId		Copy of identifier for the object.
reply	O	T_reply		Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.

Characteristics

1 The new identifier refers to the same object and possesses the same access rights as the input identifier but has an independent existence.

2 This primitive is intended for use by system and environment management programs.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AFS.d'

var  inputId,outputId      : T_systemId;
     reply                 : T_reply;

(* function ConnectAgain(  inputId : T_systemId;
                       var   outputId : T_systemId) : T_reply;
   external; *)

begin
 .
 .
 reply:=ConnectAgain(inputId,outputId);
 .
 .
end.
```

This function associates a system identifier with the parent directory of a named object and also returns the last element of the object's pathname.

The named object is specified in input by:
- the system identifier of another named object (obtained by a previous Connect operation) that indicates where the search for the object is to be made
- a pathname

The reference structure searched may be a directory or any generic memory structure. The internal structure of objectName depends on the reference structure.

reply
ConnectFather
referenceId,*objectName*,accessRights,*fatherId*,*lastName*,*offset*

Parameter	I/O	System-Defined Type Name	Type	Description
referenceId	I	T_systemId		System identifier. This indicates the structure within which "objectName" can be found.
objectName	I		packed array [min..max: integer] of char	This parameter can be a sequence of existing names separated by slashes and ending in blank or null. Each element is limited to 14 characters.

accessRights	I	T_access-Rights	enumerated	This parameter indicates the operations allowed on the file. See Connect.
fatherId	0	T_systemId		Identifier associated with the parent directory of lastName.
lastName	0		packed array [min2..max2: integer] of char	Last element of pathname 'objectName', left justified and filled with null characters.
offset			integer	Location in array of last meaningful character of lastName.
reply	0	T_reply	variant record	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not available.
	INVALIDPATHNAME	Invalid pathname.

Characteristics

1 If the named object is an alias file, the parent directory of the aliased file is connected

2 This primitive has the same characteristics as the Connect primitive

Note

See also:

Connect

Example

```
#include 'systypes.i'
type dataItem = char;      (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var  referenceId          : T_systemId;
     objectName           : packed array [ 1..6 ] of char;
     accessRights         : T_accessRights;
     fatherId             : T_systemId;
     lastName             : packed array [ 1..6 ] of char;
     offset               : integer;
     reply                : T_reply;

(* type

   T_accessRights = (nill_, r_, a_, ra_, w_, rw_, aw_, raw_, x_, rx_,
                     ax_, rax_, wx_, rwx_, awx_, rawx_);

   function ConnectFather( referenceId : T_systemId;
                     var     objectName : packed array [ min..max :
                                          integer ] of char;
                           accessRights : T_accessRights;
                     var       fatherId : T_systemId;
                     var       lastName : packed array [ min..max :
                                          integer ] of char;
                     var         offset : integer) : T_reply;
   external; *)


begin
 .
 .
 objectName:='pippo ';
 reply:=ConnectFather(referenceId,objectName,raw_,fatherId,lastName,offset);
 .
 .
end.
```

This function is identical to Connect, except that it does not follow any aliasing that may exist in the last element of objectName. It associates a system identifier with a named object. The identifier returned must be provided in subsequent operations on that object until a Disconnect is issued.
The object is specified in input by:
- the system identifier of a reference structure object (obtained whit a similar Connect)
- a pathname relative to the reference structure.

The reference structure can be a directory, program directory or any generic memory structure.

reply
ConnectLiteral
referenceId,*objectName*,accessRights,*systemId*,*userType*

Parameter	I/O	System-Defined Type Name	Type	Description
referenceId	I	T_systemId		System identifier. this indicates the structure within which the 'objectName' can be found.

objectName	I		packed array [min.. max: integer] of char	This parameter can be a sequence of names separated by '/'. It must not begin with '/' and it must terminate with a ' '. This parameter could also be the name of an alias.
accessRights	I	T_accessrights	enumerated	This parameter indicates the operations allowed on the file.It may be nill_, or any combination of the values r_ read, a_ append, w_ write, x_ execute. (see Connect primitive)
systemId	O	T_systemId		Identifier associated with 'fileName'. It is returned by the system.
userType	O		integer	Integer returned by the system.It corresponds to that provided by the user at 'create' time. It is not interpreted by the FS.
reply	O	T_reply	variant record	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table overflow.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not available.
	INVALIDPATHNAME	Invalid pathname.
	NAMENOTFOUND	Name provided does not exist.

Characteristics

1 A 'Connect" to a director or to a volume can only have "accessRights :=r_".

2 The input referenceId must be a connect-id not a session-id.

3 If objectName specifies an alias file, this file (and NOT the aliased file) will be connected. This is the only difference between ConnectLiteral and Connect.

4 Failure may occur if you try to connect a file within a volume that is already open.

5 This primitive is intended for use by system and environment management programs.

Note

See the following primitive:

- Connect

CONVERT (CONVERT A FILE TO A DIFFERENT TYPE)

This function converts the direct access file specified by the file name, which is relative to directory parentDirId, to a specified type.

```
reply
Convert
parentDirId,*fileName*,newType,recordLength,recordDeletion,coords,
pageSize,splitStrategy
```

Parameter	I/O	System-Defined Type Name	Type	Description
parentDirId	I	T_systemId		Identifier of the directory containi fileName.
fileName	I		packed array [min..max: integer] of char	This parameter can be a sequence of names separated by slashes and end ing in blank or null.Each element may have at most 14 characters. If thefile is or willbe positional or keyed, the last element may have at most 12 characters.
newType	I	T_convert	enumerated	The new type can be byte_, positional or keyed_.

recordLength	1		integer	This parameter specifies the number of bytes per record for conversion from byte to positional.
recordDeletion	1	T_recordDeletion	enumerated	This parameter specifies the required record management policy for conversion from byte to positional It may be: - yd_: records may be deleted - nd_: records will never be deleted - ydp_:records may be deleted and are packed.
coords	1	T_fieldCoords	packed record	Record which contains the following information for conversion from positional to keyed:
start			integer	- start of key
length			integer	- length of key
pageSize	1	T_pageSize	enumerated	Number of bytes per page in the B*tree for the associated indexes for converting pos to key. Possible values are: p512_,p1024_, p2048_,p4096_.

splitStrategy	I	T_split-Strategy	enumerated	Strategy to be adopted for splitting B*tree pages when they are full, for converting pos to key. Possible values are: s50_ : for randomly generated keys s90_ : when a few insertions are expected for sequentially generated keys. s100_:
reply	I	T_reply	variant record	Diagnostics.

Diagnostics The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	OUTOFDISKSPACE	Not enough space left on disk.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDOPERATION	Operation not permitted.
	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not available.
	INVALIDPATHNAME	Invalid pathname.
	NAMENOTFOUND	File name does not exist.
	KEYALREADYEXISTS	Primary index cannot be created as duplicate keys not allowed.

Characteristics

1 Conversion is permitted only to the next/previous level of file, i.e. bst to pos, pos to key, key to pos, pos to bst.

2 If possible, the conversion will be reversible, but pos to byte cannot be reversed if record deletion was permitted, as the information on deleted records will have been lost.

3 Conversion from byte to pos requires the input parameters recordLength and recordDeletion; from pos to key requires coords, pageSize and splitStrategy. Downward conversions (key to pos, pos to byte) require no parameters.

4 When conversion is from pos to key a primary index is automatically created, and the defined keys must therefore be unique or this conversion will fail (KEYALREADYEXISTS error).

Note

See also the following primitives:

- CreatePos
- CreateKeyed

Example

```
#include "systypes.i"
type dataItem = char;          (* this type is user-defined *)
#include "fstypes.i"
#include "AMS.d"

var      parentDirId  : T_systemId;
            fileName  : packed array [1..6] of char;
             newType  : T_convert;
        recordLength  : integer;
      recordDeletion  : T_recordDeletion;
              coords  : T_fieldCoords;
            pageSize  : T_pageSize;
         splitStrategy  : T_splitStrategy;
               reply  : T_reply;

(* type

   T_convert = ( byte_, positional_, keyed_ );

   T_recordDeletion = ( yd_, nd_, ydp_ );

   T_fieldCoords =  packed record
                      start : integer;
                     length : integer
                    end;

   T_pageSize = ( p512_, p1024_, p2048_, p4096_ );

   T_splitStrategy = ( s50_, s90_, s100_ );

   function Convert (      parentDirId  : T_systemId;
                     var      fileName  : packed array [1..6] of char;
                              newType  : T_convert;
                         recordLength  : integer;
                        recordDeletion  : T_recordDeletion;
                                coords  : T_fieldCoords;
                              pageSize  : T_pageSize;
                          splitStrategy  : T_splitStrategy) : T_reply;
   external; *)

begin
 .
 .
fileName:='pippo ';
reply:=Convert(parentDirId,fileName,positional_,256,nd_,coords,
               pageSize,splitStrategy);
 .
 .
end.
```

This function copies a file to a destination in accordance with a specified option.

reply
Copy
sourceDirId,*sourceFileName*,destDirId,*destFileName*,options

Parameter	I/O	System-Defined Type Name	Type	Description
sourceDirId	I	T_systemId		Identifier of the directory containing sourceFileName.
sourceFileName	I		packed array [min..max: integer] of char	This parameter can be a sequence of existing names, separated by slashes and ending in blank or null.Each element must have at most 14 characters. If the file is positional or keyed,.the last element must have at most 12 characters.
destDirId	I	T_systemId		Identifier of the directory containing destFileName.

destFileName	I		packed array [min2.. max2 : integer] of char	see sourceFileName
options	I	T_copy-Options	enumerated	Type of copy operation. Possible values: replace_, dontreplace_, append_, see Characteristic 1.
reply	O	T_reply	variant record	Diagnostics.

Diagnostics The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	OUTOFDISKSPACE	Not enough space left on disk.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDOPERATION	Operation not permitted.
	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not available.
	INVALIDPATHNAME	Invalid pathname.
	NAMEALREADYEXISTS	Directory/File cannot be created.
	NAMENOTFOUND	Directory/File name does not exist

Characteristics

1 In general, the replace_ option causes overwriting (if the destination already exists) while dontreplace_ does not permit overwriting (it can cause an error if the destination already exists.) The append_ option causes data to be appended to a file or peripheral.

2 Further characteristics particular to various combinations of source and destinations in conjunction with different options, are shown in the table below.

Source	Destination	Option	Notes
byte or sequential file, volume or peripheral	peripheral	any	No other types of source permitted. Option dontreplace_ never causes abort.
byte or sequential file	byte or sequential file or peripheral	append_	No other source/ destination combinations permitted.
file (any type) or peripheral	directory or volume	replace_ or dontreplace_	New file created in directory or volume with last element of name equal to last element of source name. Option applied to source.

directory	directory or volume	replace_ or dontreplace_	Source directory is recursively copied into directory or volume. New directory has last element of name equal to last element of source name. Option applied to source.
directory	file(any type)	replace_ or dontreplace_	Destination file removed and new directory created with name equal to destFileName. Option has no effect.
volume	byte file or peripheral	replace_ or dontreplace_	Source volume copied onto file or peripheral and then removed. Option has no effect.
volume	sequential file or volume	replace_ or dontreplace_	Destination file or volume removed and new volume created with name equal to destFileName. Option has no effect.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var            sourceDirId  : T_systemId;
             sourceFileName : packed array [ 1..6 ] of char;
   destDirId,destFileName   : T_systemId;
                    options : T_copyOptions;
                      reply : T_reply;

(* type

   T_copyOptions = ( replace_, contreplace_, append_ );

   function Copy(           sourceDirId  : T_systemId;
                  var  sourceFileName    : packed array [ min..max ] integer
                                           of char;
                            destDirId    : T_systemId;
                  var     destFileName   : T_systemId;
                               options   : T_copyOptions) : T_reply;
   external; *);


begin
 .
 .
 sourceFileName:='pipin ';
 destFileName:='pipot '
 reply:=Copy(sourceDirId,sourceFileName,destDirId,destFileName,replace_);
 .
 .
end.
```

This function creates an alias file for a specified file.

```
reply
CreateAlias
parentDirId,*alias*,*aliasedFile*
```

Parameter	I/O	System-Defined Type Name	Type	Description
parentDirId	I	T_systemId		Identifier of directory containing aliasedFile.
alias	I		packed array [min..max: integer] of char	This parameter can be a sequence of existing names, separated by slashes and ending in blank or null. Each element can have at most 14 characters.
aliasedFile	I		packed array [min2..max2: integer] of char	Name of file to be aliased. This pathname must begin with "/" local rot).
reply	O	T_reply	variant record	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	OUTOFDISKSPACE	Not enough space left on disk.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatable with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not available.
	INVALIDPATHNAME	Invalid pathname.
	NAMEALREADYEXISTS	File name cannot be created.

Characteristics

1 The created file is a byte file of internal type alias, with the name specified by the 'alias' parameter. It contains the name 'aliased file' and is used to access 'aliased file'.

2 The aliasing mechanism operates only on intermediate elements of path names in general, but on all elements in Connect. Thus a Connect request on an alias file will connect the aliased file. When the mechanism does not operate, the alias file itself is accessed and not the aliased file. Thus Remove (alias) removes the alias and not the aliased file.

Note

See also the following primitives:

- Connect

Example

```
#include "systypes.i"
type dataItem = char;          (* this type is user-defined *)
#include "fstypes.i"
#include "AMS.d"

var          parentDirId  : T_systemId;
                   alias  : packed array [ 1..6 ] of char;
             aliasedfile  : packed array [ 1..6 ] of char;
                   reply  : T_reply;

(* function CreateAlias(  parentDirId : T_systemId;
                                 alias : packed array [ min..max
                                         integer of char;
                           aliasedFile : packed array [ min..max
                                         integer of char) : T_reply;
   external; *)


begin
 .
 .
 alias:="outf "
 aliasedFile:="pippo "
 reply:=CreateAlias(parentDirId,alias,aliasedFile);
 .
 .
end.
```

This function creates a new byte file and assigns it the name contained in 'fileName'.

reply
CreateByte
parentDirId,*fileName*,allocationUnit,userType,*allocatedSize*

Parameter	I/O	System-Defined Type Name	Type	Description
parentDirId	I	T_systemId		System identifier. It indicates the directory within which 'fileName' will be created.
fileName	I		packed array [min..max: integer] of char	This parameter can be a sequence of existing names, separated by '/' and terminated with a ' '. It must not begin with '/' and must terminate with ' ' The names present in the sequence must be a maximum of 14 characters.

allocation-Unit	1		longinteger	Number of memory bytes which the system tries to allocate contiguously to the file.It is a hint for the system. This parameter is also used for subsequent file expansions.
userType	1		integer	Integer which the user assigns to the file as a personal identifier. It is not interpreted by the system. This value is returned by 'Connect'.
allocatedSize	0		longinteger	Number of bytes assigned by the system to the initial extent.
reply	0	T_reply	variant record	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	OUTOFDISKSPACE	No more space on disk.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not available.
	INVALIDPATHNAME	Invalid path name.
	NAMEALREADYEXISTS	File name already exists.

Characteristics

1 The correct use of 'allocationUnit' is to set it to the initial size required for the file.It can then be modified using WriteInfoByte so that any additional extents allocated are smaller. The system rounds the value to the next highest multiple of 512 bytes.

2 'fileName' may start with a blank or blanks. These will however be considered as insignificant by the system.

3 The new file will have internal type 'data_' but this can be changed using ChangeType.

4 Byte files can be sequentially handled using the sequential primitives.

5 If subsequent conversion to a positional or keyed file is likely, the last element of the file name should be limited to 12 characters.

Note

See the following primitives:

- WriteInfoByte
- Connect
- Change Type

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var  parentDirId                    : T_systemId;
     fileName                       : packed array [ 1..17 ] of char;
     allocationUnit,allocatedSize   : longinteger;
     userType                       : integer;
     reply                          : T_reply;

(* function CreateByte(        parentDirId : T_systemId;
                          var       fileName : packed array [ min..max :
                                                integer ] of char;

                              allocationUnit : longinteger;
                                    userType : integer;
                          var allocatedSize  : longinteger) : T_reply;
   external; *)

begin
 .
 .
fileName:='usr/milano/pippo ';
reply:=CreateByte(parentDirId,fileName,512,0,allocatedSize);
 .
 .
end.
```

This function creates a directory and assigns it the name contained in 'dirName' which is relative to the directory specified by the system identifier.

```
reply
CreateDir
parentDirId,*dirName*,userType
```

Parameter	I/O	System-Defined Type Name	Type	Description
parentDirId	I	T_systemId		System identifier indicating the directory in which 'dirName' will be created.
dirName	I		packed array [min..max: integer] of char	This parameter can be a sequence of existing names, separated by '/'. It must not begin with '/' and must terminate with ' ' The names in the sequence must be a maximum of 14 characters.

userType	I		integer	Integer which the user assigns to the directory as a personal identifier. It is not interpreted by the system. This value is returned by 'Connect'.
reply	O	T_reply	variant record	Diagnostics.

Diagnostics The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	OUTOFDISKSPACE	No more space on disk.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not available.
	INVALIDPATHNAME	Invalid path name.
	NAMEALREADYEXISTS	Directory cannot be created.

Characteristics

1 'fileName' may start with a blank or blanks, these will however be considered as not significant by the system.

2 The directory created is a byte file with internal type 'dir_'. This can be changed using ChangeType.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'XMC.d'

var  parentDirId : T_systemId;
     dirName     : packed array [ 1..9 ] of char;
     userType    : integer;
     reply       : T_reply;

(* function CreateDir(    parentDirId : T_systemId;
                      var     dirName : packed array [ min..max :
                                            integer ] of char;
                              userType : integer) : T_reply;
   external; *)


begin
 .
 .
 dirName:='usr/anna ';
 reply:=CreateDir(parentDirId,dirName,0);
 .
 .
end.
```

This function creates a new keyed file and assigns it the name contained in 'fileName' which is relative to the directory specified by 'parentDirId'.

```
reply
CreateKeyed
parentDirId,*fileName*,allocationUnit,userType,*allocatedSize*,
recordLength,primaryCoords,primaryPageSize,primarySplitStrategy
```

Parameter	I/O	System-Defined Type Name	Type	Description
parentDirId	I	T_systemId		System identifier. It indicates the directory within which 'fileName' will be created.
fileName	I		packed array [min..max: integer] of char	This parameter may be a sequence of existing names, separated by '/'. It must not begin with '/' and must end with a ' '. Each intermediate component may have a maximum of 14 characters. The last component is limited to 12 characters.

allocationUnit	I		longinteger	Number of memory bytes which the system tries to allocate to the file. This parameter is also used for subsequent file expansions.
userType	I		integer	Integer which the user assigns to the file as personal identifier. It is not interpreted by the system. This value is returned by 'Connect'.
allocatedSize	O		longinteger	Number of bytes assigned by the system to the initial extent.
recordLength	I		integer	Length of records.
primaryCoords	I	T_field-Coords	packed record	Record containing the following information:
start			integer	- start of the primary key within the record. If the parameter has a negative value, this indicates an external key
length			integer	- length of key.

primaryPageSize	I	T_pageSize	enumerated	This parameter contains one of following values: p512_, p1024_, p2048_, p4096_. These values indicate the page size ofthe B* tree for the primary index.
primarySplit-Strategy	I	T_split-Strategy	enumerated	This parameter contains one of the following values: s50_, s90_, s100_. It indicates the splitting strategy to be adopted when the B* - tree pages are full. s50_: for keys generated in a random manner S90_: when a few insertions are expected S100_:for sequentially generated keys.
reply	O	T_reply	variant record	Diagnostics.

Diagnostics

The primitive gives one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	OUTOFDISKSPACE	'allocationUnit' exceeds the available space on disk.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not ready.
	INVALIDPATHNAME	Invalid pathname.
	NAMEALREADYEXISTS	File cannot be created.

Characteristics

1 The allocationUnit and primarySplitStrategy parameters may be modified subsequently with WriteInfoKeyed.

2 The correct use of allocationUnit is to set it to the initial size required for the file. It can then be modified by using WriteInfoKeyed. The system rounds this value to be consistent with its allocation strategy, i.e. to a whole number of disk sectors.

3 The parameter allocationUnit is used by the system only to allocate memory for the data file. The system will generate a different allocationUnit for index files. It is based on the input allocation-Unit, record size and key size.

4 CreateKeyed associates one primary index with the file. The unique primary keys which may be internal or external are stored in the primary index. The system assigns to the primary index the sequence number 0. The primary key must not be changed during updates.

5 'fileName' may start with a blank or blanks, these will however be considered as not significant by the system.

6 A description of all the fields in the records may be obtained using GetRecordView.

Notes

See also the following primitives:

- WriteInfoKeyed
- GetRecordView
- Attach

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'APS.d'

var  parentDirId                    : T_systemId;
     fileName                       : packed array [ 1..5 ] of char;
     allocationUnit,allocatedSize   : longinteger;
     userType,recordLength          : integer;
     primaryCoords                  : T_fieldCoords;
     primaryPageSize                : T_pageSize;
     primarySplitStrategy           : T_splitStrategy;
     reply                          : T_reply;

(* type

   T_fieldCoords = packed record
                       start : integer;
                      length : integer
                   end;

   T_pageSize = ( p512_, p1024_, p2048_, p4096_ );

   T_splitStrategy = ( s50_, s90_, s100_ );

   function CreateKeyed(      parentDirId : T_systemId;
                         var      fileName : packed array [ min..max :
                                                integer ] of char;
                             allocationUnit : longinteger;
                                   userType : integer;
                         var  allocatedSize : longinteger;
                                recordLength : integer;
                               primaryCoords : T_fieldCoords;
                           primaryPageSize : T_pageSize;
                        primarySplitStrategy : T_splitStrategy) : T_reply;
   external; *)

begin
 .
 .
 fileName:='anna ';
 primaryCoords.start:=0;
 primaryCoords.length:=5;
 reply:=CreateKeyed(parentDirId,fileName,512,2,allocatedSize,
                                  10,primaryCoords,p512_,s50_);
 .
 .
end.
```

This function creates a new positional file and assigns it the name 'fileName' which is relative to the directory specified by parentDirId.

```
reply
CreatePos
parentDirId,*fileName*,allocationUnit,userType,*allocatedSize*,
recordLength,recordDeletion
```

Parameter	I/O	System-Defined Type Name	Type	Description
parentDirId	I	T_systemId		System identifier. It indicates the directory within which 'fileName' will be created.
fileName	I		packed array [min..max: integer] of char	This parameter may be a sequence of existing names separated by '/' and must be terminated by a ' '. Each intermediate component may have a maximum of 14 characters. The last component is limited to 12 characters. It must not begin with a '/'.
allocationUnit	I		longinteger	Number of memory bytes which the

			system tries to allocate contiguously to the file. It is a hint for the system. This parameter is also used for subsequent file expansions.
userType	1	integer	Integer which the user assigns to the file as a personal identifier. It is not interpreted by the system. This value is returned to the user by 'Connect'.
allocatedSize	0	longinteger	Number of bytes assigned by the system to the initial extent.
recordLength	1	integer	Record length.

recordDeletion	1	T_record-Deletion	enumerated	This parameter may contain one of the following values: yd_: it is possible to delete the file records. nd_: file records cannot be deleted. ydp_:it is possible to delete the records. The valid records are compacted within the allocated space.
reply	0	T_reply	variant record	Diagnostics.

Diagnostics

The primitives returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	OUTOFDISKSPACE	'allocationUnit' exceeds the available space on disk.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not available.
	INVALIDPATHNAME	Invalid path name.
	NAMEALREADYEXISTS	Name of file already exists.

Characteristics

1 The correct use of allocationUnit is to set it to the initial size of the file.It can then be modified using WriteInfoPos.The system rounds this value to the next highest multiple of 512 bytes.

2 'fileName' may start with a blank or blanks, these will however be considered as insignificant by the system.

Notes

See also the following primitives:

- WriteInfoPos
- Connect

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'API.d'

var  parentDirId                    : T_systemId;
     fileName                       : packed array [ 1..5 ] of char;
     allocationUnit,allocatedSize   : longinteger;
     userType,recordLength          : integer;
     recordDeletion                 : T_recordDeletion;
     reply                          : T_reply;

(* type

   T_recordDeletion = ( yd_, nd_, ycp_ )

   function CreatePos(       parentDirId : T_systemId;
                        var      fileName : packed array [ min..max :
                                                integer ] of char;
                           allocationUnit : longinteger;
                                 userType : integer;
                        var allocatedSize : longinteger;
                             recordLength : integer;
                           recordDeletion : T_recordDeletion) : T_reply;
   external; *)


begin
 .
 .
 fileName:='anna ';
 reply:=CreatePos(parentDirId,fileName,512,1,allocatedSize,10,yd_);
 .
 .
end.
```

This function allows a terminal, printer or free-running connection to another machine to be seen as a sequential file. It creates a sequential file and assigns to it the name contained in 'fileName' which is relative to the directory specified by parentDirId.

```
reply
CreateSeq
parentDirId,*fileName*,seqDevKind,userType,locationId
```

Parameter	I/O	System-Defined Type Name	Type	Description
parentDirId	I	T_systemId		System identifier. It indicates the directory within which 'fileName' will be created.
fileName	I		packed array [min...max: integer] of char	This parameter may be a sequence of existing names, separated by a '/'. It must not begin with a '/' and must end with a ' '. Each intermediate component may have a maximum of 14 characters.

seqDev	I	T_seqDev	enumerated	Parameter which indicates the type of device.It may contain one of the following values: ot_: terminal op_: printer fr_: free-running connection
userType	I		integer	Integer which the user assigns to the file as his own identifier.It is not interpreted by the system.
devId	I	T_systemId		Specifies the particular peripheral to which the created file will correspond.
reply	O	T_reply	variant record	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	OUTOFDISKSPACE	No more space on disk.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not available.
	INVALIDPATHNAME	Invalid pathname.
	NAMEALREADYEXISTS	File cannot be created.

Characteristics

1 A sequential file may only be handled by sequential primitives (which can also be used to handle byte files on disc sequentially).

2 'fileName' may start with a blank or blanks, these however will be considered as insignificant by the system.

Notes

See also the following primitive:

- Connect

Example

```
#include "systypes.i"
type dataItem = char;          (* this type is user-defined *)
#include "fstypes.i"
#include "AMS.d"

var  parentDirId : T_systemId;
     fileName    : packed array [ 1..6 ] of char;
     secDev      : T_secDev;
     userType    : integer;
     dev         : T_systemId;
     reply       : T_reply;

(* type

   T_seqDev = ( st_, op_, fr_ );

   function CreateSeq(       parentDirId : T_systemId;
                        var     fileName : packed array [ min..max :
                                           integer ] of char;
                              seqDev : T_secDev;
                            userType : integer;
                                 dev : T_systemId) : T_reply;
   external; *)

begin
 .
 .
 fileName:='pippo ';
 reply:=CreateSeq(parentDirId,fileName,st_,2,dev);
 .
 .
end.
```

This function deletes the record singled out by the primary key of the file specified by the system identifier.

```
reply
DeleteKeyed
systemId,*primaryKey*,keyStartIndex
```

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
primaryKey	I		packed array [min..max: integer] of dataItem	Array containing the primary key of the record to be deleted. 'dataItem' is defined by the user and its length must be of 8 bits, e.g. char, byte (-128..127).
keyStartIndex	I		integer	Integer within the 'primaryKey' range. It indicates the initial position of the key.
reply	O	T_reply	variant record	Diagnostics.

Diagnostics The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDOPERATION	Invalid operation.
	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameter.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not available.
	KEYNOTFOUND	The primary key does not exist.
	RECORDLOCKED	Record inaccessible because it is already locked by another process.
FSWARNING	INCONSISTENTDATABASE	Index not consistent with conterts of file.

Characteristics

1. DeleteKeyed verifies that there is no lock on the record before executing the operation. If there is a lock it returns an error (RECORD LOCKED).

2. The key length is equal to that established by CreateKeyed.

3. If the record with the specified key does not exist, an error is returned (KEYNOTFOUND).

4. If secondary indices for the file exist, the file must have been opened with r_ access (as well as w_) so that these indices can be read and updated (by deleting secondary keys associated with the deleted record).

5. If no primary index exists, it is not possible to delete records (INVALIDOPERATION).

Example

```
#include "systypes.i"
type dataItem = char;          (* this type is user-defined *)
#include "fstypes.i"
#include "AMS.d"

var  systemId       : T_systemId;
     primaryKey     : packed array [ 1..? ] of dataItem;
     keyStartIndex  : integer;
     reply          : T_reply;

(* type

   type dataItem = char;

   function DeleteKeyed(          systemId : T_systemId;
                          var   primaryKey : packed array [ min..max :
                                             integer ] of dataItem;
                             keyStartIndex : integer) : T_reply;

   external; *)

begin
 .
 .
 reply:=DeleteKeyed(systemId,primaryKey,1);
 .
 .
end.
```

DELETEPOS (DELETE A RECORD FROM A POSITIONAL FILE)

This function deletes the record specified by the user in the file singled out by the system identifier.

```
reply
DeletePos
systemId,position
```

Parameters	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
position	I		longinteger	Numerical value which indicates the position of the record.
reply	O	T_reply	variant record	Diagnostics.

Diagnostics The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDOPERATION	Invalid operation.
	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not available.
	RECORDLOCKED	Record inaccessible because it is already locked by another process.
	RECORDNOTFOUND	No record exists at the specified postition.

Characteristics

1 DeletePos verifies that there is no lock on the record before executing the operation. If there is a lock it returns an error.

2 If the record does not exist an error is returned.

Example

```
#include "systypes.i"
type dataItem = char;          (* this type is user-defined *)
#include "fstypes.i"
#include "AMI.d"

var  systemId : T_systemId;
     position : longinteger;
     reply    : T_reply;

(* function DeletePos( systemId : T_systemId;
                       position : longinteger) : T_reply;
   external; *)

begin
 .
 .
 reply:=DeletePos(systemId,1);
 .
 .
end.
```

This function removes a specific index associated with the keyed file indicated by "fileName". The memory area occupied by the index is returned immediately to the system.

```
reply
Detach
parentDirId,*fileName*,fieldNumber
```

Parameter	I/O	System-Defined Type Name	Type	Description
parentDirId	I	T_systemId		System identifier indicating the directory within which the 'fileName' can be found.
fileName	I		packed array [min..max: integer] of char	This parameter may be a sequence of existing names separated by '/' and must be terminated by a blank. Each intermediate component may have a maximum of 14 characters. The last component is limited to 12 characters. It must not begin with a '/'.

fieldNumber	I		integer	Integer which indicates the sequence number of the index to be removed. This number is returned by the Attach primitive.
reply	O	T_reply	variant record	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDOPERATION	Invalid operation.
	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not available.
	INVALIDPATHNAME	Invalid path name.
	NAMENOTFOUND	File name does not exist.

Characteristics

1 This primitive may remove a primary index.

2 After the operation has been carried out the file must still have at least one index.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var  parentDirId : T_systemId;
     fileName    : packed array [ 1..5 ] of char;
     fieldNumber : integer;
     reply       : T_reply;

(* function Detach (      parentDirId : T_systemId;
                    var       fileName : packed array [ min..max :
                                              integer ] of char;
                          fieldNumber : integer) : T_reply;
   external; *)


begin
 .
 .
 fileName:='anna ';
 reply:=Detach(parentDirId,fileName,fieldNumber);
 .
 .
end.
```

This function annuls the preceding 'Connect'.

reply
Disconnect
systemId

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		Indicates the file to be disconnected. (This parameter is returned by 'Connect').
reply	O	T_reply	variant record	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.

Notes

See also the following primitive:

- Connect

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'ANS.d'

var  systemId : T_systemId;
     fileName : packed array [ 1..5 ] of char;
     reply    : T_reply;

(* function Disconnect(var systemId : T_systemId) : T_reply;

   external; *)


begin
 .
 .
 fileName:='anna ';
 reply:=Disconnect(systemId);
 .
 .
end.
```

This function returns the byte offset of a file's last valid byte. The file is indicated by its system identifier. The primitive does not return errors.

posEnd
EndByte
systemId,*lastAddress*

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
lastAddress	O	T_address		Address of the last valid byte of the file.
posEnd	O		longinteger	Byte offset of the last valid byte of the file.

Characteristics

1 The byte offset returned by the primitive may be used as input by ReadByte.

2 The address returned may be used as input by ComputeByte or SetBoundsByte. It is not interpretable by the user.

3 It is not necessary to open the file before invoking this primitive.

Notes

See also the following primitives:

- ReadByte
- ComputeByte
- SetBoundsByte

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'APS.d'

var   systemId     : T_systemId;
      lastAddress  : T_address;
      posEnd       : longinteger;

(* type

   dummyRec = record
              end;

   T_address = ^dummyRec;

   function EndByte(        systemId : T_systemId;
                        var lastAddress : T_address) : longinteger;
   external; *)

begin
 .
 .
 posEnd:=EncByte(systemId,lastAddress);
 .
 .
end.
```

This function returns the position of the record corresponding to the highest key present in the specified index for a particular file. The file is indicated by its system identifier and the index by its sequence number (fieldNumber). The primitive also returns the key and its address within the indicated index. It does not return errors.

posRecord
EndKeyed
systemId,fieldNumber,*endAddress*,*endKey*,keyStartIndex

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
fieldNumber	I		integer	Index sequence number.
endAddress	O	T_address		Address of the key in the indicated index.
endKey	O		packed array [min..max: integer]of dataItem	Array containing the record key. 'dataItem' is defined by the user and its length must be of 8 bits, e.g. char, byte (-128..127).

keyStartIndex	1	integer	Integer within the 'endKey' range. The key is transferred into the specified key area starting from the location specified by this parameter.
posRecord	0	longinteger	The position of the last record.

Characteristics

1 The record thus selected may be read with ReadPos.

2 The returned address may be an input to ComputeKeyed or SetBoundsKeyed. It is not interpretable by the user.

3 It is not necessary to open the file before invoking this primitive.

Note

See also the following primitives:

- ReadPos
- ComputeKeyed
- SetBoundsKeyed

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var  systemId                    : T_systemId;
     fieldNumber,keyStartIndex   : integer;
     endAddress                  : T_address;
     endKey                      : packed array [ 1..16 ] of dataItem;
     posRecord                   : longinteger;

(* type

   dataItem = char;

   dummyRec = record
              end;

   T_address = ^dummyRec;

   function EndKeyed(         systemId : T_systemId;
                          fieldNumber : integer;
                  var    endAddress : T_address;
                  var        endKey : packed array [ min..max :
                                        integer ] of dataItem;
                      keyStartIndex : integer) : longinteger;
   external; *)

begin
 .
 .
 posRecord:=EndKeyed(systemId,0,endAddress,endkey,1);
 .
 .
end.
```

ENDPOS (FIND LAST RECORD POSITION OF A POSITIONAL FILE)

This function returns the position and the address of the last valid record in the file indicated by the system identifier.The primitive does not return errors.

last
EndPos
systemId,*lastAddress*

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
lastAddress	O	T_address		Address of the last valid record in the file.
last	O		longinteger	Position of the last record.

Characteristics

1 The record thus selected may be read using ReadPos.

2 The address returned may be used as input by ComputePos or SetBoundsPos. Is is not directly interpretable by the user.

3 It is not necessary to open the file before invoking this primitive.

Note

See also the following primitives:

- ReadPos
- ComputePos
- SetBoundsPos

Example

```
#include "systypes.i"
type dataItem = char;          (* this type is user-defined *)
#include "fstypes.i"
#include "AMS.d"

var  systemId    : T_systemId;
     lastAdcress : T_address;
     last        : longinteger;

(* type

   dummyRec = record
              end;

   T_address = ^dummyRec;

   function EndPos(       systemId : T_systemId;
                   var lastAddress : T_address) : longinteger;

   external; *)

begin
 .
 .
 last:=EndPos(systemId,lastAddress);
 .
 .
end.
```

GETDEVID (GET IDENTIFIER OF A DEVICE)

This function returns the device identifier of the device specified by the unit number.

reply
GetDevId
unitNo, *devId*

Parameter	I/O	System-Defined Type Name	Type	Description
unitNo	I		integer	Unit number of a sequential peripheral.
devId	O	T_systemId		System identifier of the device
reply	O	T_reply	variant record	Diagnostics

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDOPERATION	Operation not permitted.
	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var  unitNo   : integer;
     devId    : T_systemId;
     reply    : T_reply;

(* function GetDevId(      unitNo   : integer;
                       var   devId   : T_systemId) : T_reply;
   external; *)

begin
 .
 .
 reply:=GetDevId(3,devId);
 .
 .
end.
```

This function supplies pointers to the internal descriptors of the file specified by the system identifier.

reply
GetFilePtr
systemId,*fileId*,*pddId*,*volumePddId*

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
fileId	O		longinteger	Unique file identifier.
pddId	O		longinteger	Pointer to permanent descriptor (PDD) of file on disk; within its parent volume.
volumePddId	O		longinteger	Pointer to permanent descriptor of volume containing the file, within its parent volume
reply	O	T-reply	variant record	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly
SYSTEM	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table overflow.
	NOROUTE	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.

Characteristic — The identifiers returned may be used in input to certain utilities; in particular "fileId" uniquely specifies the file, for example in auxiliary tables.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var    systemId    : T_systemId;
         fileId    : longinteger;
    volumePddId    : longinteger;
         reply     : T_reply;


(*    function GetFilePtr (          systemId : T_systemId;
                             var         fileId : longinteger
                             var          pddId : longinteger;
                             var volumePddId : longinteger) : T_reply;

  external; *)

begin
 .
 .
 reply:=GetFilePtr(systemId,fileId,pddId,volumePddId);
 .
 .
end.
```

GETNAME (GET LOCAL NAME OF A FILE)

This function returns the last element of the pathname of the file specified by the system identifier.

```
reply
GetName
systemId,*localName*,startIndex
```

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
localName	O		packed array [min.. max: integer] of char	Last element of pathname (14 characters, padded with blanks if and as necessary).
startIndex	I		integer	Byte offset of first meaningful character in localName.
reply	O	T_reply	variant record	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
FSERROR	DEVICENOTREADY	Device not available.
	NAMENOTFOUND	File name does not exist.

Characteristic

If the input system identifier is for the local root directory ("/"), then the machine name is returned.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'API.d'

var     systemId  : T_systemId;
      localName   : packed array [ 1..20 ] of char;
     startIndex   : integer;
          reply   : T_reply;


(*      function GetName(        systemId : T_systemId;
                         var   localName : packed array [ min..max :
                                             integer] of char;
                               startIndex : integer) : T_reply;
  external; *)


begin
 .
 .
 reply:=GetName(systemId,localName,1);
 .
 .
end.
```

This function returns the pathname of the file identified by the system identifier.

```
reply
GetPathName
systemId,*pathName*,startIndex,*length*
```

Parameter	I/O	System-Defined Type Name	Type	Description
systemId		T_systemId		File identifier.
pathName	O		packed array [min.. max: integer] of char	This parameter begins with "/" and consists of existing names separated by slashes. Each element is limited to 14 characters. The last element may only have 12 characters if the file is keyed or positional.
startIndex	I		integer	Byte offset in pathName array at which you wish pathname to begin.
length	O		integer	Number of characters in pathname.
reply	O	T_reply	variant record	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDOPERATION	Operation not permitted.
	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not available.

Characteristics

1 The pathname returned will be the global pathname if possible; otherwise the local pathname.

2 The global pathname may be specified using the "father directory" convention, in which case it begins "/..", indicating father of local root, and the next element is the machine name.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'API.d'

var     systemId  : T_systemId;
        pathName  : packed array [1..20] of char;
      startIndex  : integer;
          length  : integer;
           reply  : T_reply;

(*    function GetPathName(       systemId  : T_systemId;
                              var  pathName  : packed array [ min..max ] :
                                               of char;
                                 startIndex  : integer;
                              var    length  : integer) : T_reply;
      external; *)

begin
 .
 .
 reply:=GetPathName(systemId,pathName,1,length);
 .
 .
end.
```

GETRECORDVIEW (GET THE VIEW OF THE RECORDS)

This function supplies a complete description of the (user-defined) fields in the records belonging to the file specified by the system identifier.

reply
GetRecordView
systemId,*fieldTable*,*nrFields*

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
fieldTable	O	T_field	packed record	Record containing the following information:
fieldNumber			integer	integer which indicates the index sequence number
fieldCoords		T_field-Coords	packed record	key coordinates:
start			integer	- start of key
length			integer	- key length
fieldKind		T_field-Kind	enumerated	type of index: pm_: primary index su_: secondary index with unique keys

				sd_: secondary index with duplicate keys.
nrFields	0		integer	Number of indices associated with the file.
reply	0	T_reply	variant record	Diagnostics.

Diagnostics — The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.
	INVALIDOPERATION	Invalid operation.
	INVALIDPARAMETERS	Invalid parameters.

Characteristics It is not necessary to open the file before invoking this primitive.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'APC.d'

var   systemId    : T_systemId;
      fieldTable  : packed array [ 1..20 ] of T_field;
      nrFields    : integer;
      reply       : T_reply;

(* type

   T_fieldCoords = packed records
                        start : integer;
                       length : integer
                   end;

   T_fieldKind = ( pn_, su_, sd_ );

   T_field = packed record
                 fieldNumber : integer;
                 fieldCoords : T_fieldCoords;
                   fieldKind : T_fieldKind
             end;

   function GetRecordView(       systemId : T_systemId;
                           var fieldTable : packed array [ min..max :
                                                integer ] of T_field;
                           var   nrFields : integer) : T_reply;
   external; *)


begin
 .
 .
 reply:=GetRecordView(systemId,fieldTable,nrFields);
 .
 .
end.
```

GETROOT (GET IDENTIFIER OF GLOBAL ROOT)

This function returns the system identifier of the global root directory.

rootId
GetRoot

Parameter	I/O	System-Defined Type Name	Type	Description
rootId	O	T_systemId		Root directory identifier.

Note — Users are advised to obtain the global root identifier from the program context, and not via this primitive.

GETTYPE (GET FILE TYPE)

This function returns the file type based on the identifier given in input.

reply
GetType
systemId,*fileType*

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
fileType	O	T_file	enumerated	Specifies the file type.See characteristic 2.
reply	O	T_reply	variant record	Diagnostics.

Diagnostics The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.

Characteristics

1 It is not necessary to perform an Open on the 'systemId' before calling GetType.

2 The possible values returned by the parameter 'fileType' are :

- bst_: byte file
- pos_: positional file
- key_: keyed file
- blkDev_ : blocked device
- seqDev_ : sequential device

The subtypes of byte file are:

- data_: data file

- dir_: directory
- vol_: volume
- iotable_: program context
- alias_: file name alias
- program_: program directory

The subtypes of positional file are:

- posyd_: record deletion permitted
- posnd_: no record deletion
- posydp_: deletion and packed

The subtypes of keyed file are:

- keyin_: internal primary keys
- keyex_: external primary keys

The subtypes of blocked device are:

- flop_: floppy disk
- mflop_: mini-floppy disk
- hdisk_: hard disk
- sc_: streaming cartridge tape
- mt_: magnetic tape

The subtypes of sequential device are:

- ot_: terminal
- op_: printer
- fr_: free-running connection

Example

```
#include "systypes.i"
type dataItem = char;          (* this type is user-defined *)
#include "fstypes.i"
#include "AMS.d"

var   systemId : T_systemId;
      fileType : T_file;
      reply    : T_reply;

(* type

   T_file = ( bst_, pos_, key_, blkDev_, seqDev_ )

   function GetType(      systemId : T_systemId;
                      var   fileType : T_file) : T_reply;
   external; *)

begin
 .
 .
 reply:=GetType(systemId,fileType);
 .
 .
end.
```

This function retrieves previously written information about a file.

```
reply
GetXtn
systemId,*offset*
```

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier (connect-id or session-id).
offset	O		integer	Information associated with file.
reply	O	T_reply	variant	Diagnostics.

Diagnostics — The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation correctly executed.
SYSTEM	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.

Characteristics

1 The information is accessed using a connect-id or session-id, but is linked to the file and not to any identifier temporarily associated with it.

2 The information was written using PutXtn.

3 This primitive is intended for system use when managing multiple file structures.(The information is an index to the multiple file table).

Note

See also:

- PutXtn

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'APS.d'

var         systemId : T_systemId;
              offset : integer;
               reply : T_reply;

(*  function GetXtn(          systemId : T_systemId;
                        var     offset : integer) : T_reply;
    external; *)

begin
 .
 .
 reply:=GetXtn(systemId,offset);
 .
 .
end.
```

This function lists the names in the specified directory, and may list those in any subdirectories if requested.

reply
ListDir
parentDirId,*dirName*,listSubDir,outputFile

Parameter	I/O	System-Defined Type Name	Type	Description
parentDirId	I	T_systemId		Identifier of directory containing dirName.
dirName	I		packed array [min..max: integer]of dataItem	This parameter may be a sequence of existing names separated by slashes. It must end in blank or null. Each component is limited to 14 characters.
listSubDir	I	T_listDir	enumerated	List option: listsubdir_: list contents of subdirectories as well dontlistsubdir_: do not list contents of subdirectories but indicate them listonly_: list first level only.

outputFile	I	T_systemId		Identifier of file to which output is to be directed. (Names are not listed in alphabetical order).
reply	O	T_reply	variant record	Diagnostics.

Diagnostics — The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDOPERATION	Operation not permitted.
	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not available.
	INVALIDPATHNAME	Invalid pathname.
	NAMENOTFOUND	Directory name does not exist.

Characteristic	The directory name may have leading blanks, as these are ignored by the system.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'APS.d'

var  parentDirId,outputFile : T_systemId;
                    dirName : packed array [ 1..20 ] of dataItem;
                listSubDir : T_listDir;

(* type

   dataItem = char;

   T_listDir = (listsubdir_, dontlistsubdir_,listonly_);

   function ListDir(      parentDirId : T_systemId;
                     var      dirName : packed array [ 1..20 ] of dataItem;
                           listSubDir : T_listDir;
                           outputFile : T_systemId) : T_reply;
   external; *)

begin
 .
 .
 dirName:='workdir ';
 reply:=ListDir(parentDirId,dirName,listsubdir_,output'file);
 .
 .
end.
```

```
***   LOCKRECORDKEYED EXAMPLE   ***
```

This function locks the record singled out by the specified key. It belongs to the file indicated by the system identifier and by the index sequence number. It is possible to protect records which do not yet exist.

reply
LockRecordKeyed
systemId,fieldNumber,*key*,keyStartIndex,lockLevel

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
fieldNumber	I		integer	Integer which indicates the index sequence number.
key	I		packed array min..max: integer of dataItem	Array which contains the key that identifies the record.'dataItem' is defined by the user and its length must be of 8 bits, e.g. char, byte (-128..127).

keyStartIndex	I		integer	Integer included in the 'key' range It indicates the initial key location.
lockLevel	I	T_lock-Level	enumerated	This parameter indicates the lock level to be set for the record. n : no lock (unlock if previously locked) m_: the requesting process can read or modify the record,the others can only read it. The value e_: exclusive is not available for records.
reply	O	T_reply	variant record	Diagnostics.

Diagnostics

The primitives returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDOPERATION	Invalid operation.
	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
FSERROR	TIMEOUT	Time expired.

Characteristics

1 The key length is that specified when the file is created.

2 If the lock is not available and the timeout (which was declared in OpenKeyed) expires, the operation fails (TIMEOUT error).

3 If this primitive is used on a secondary index with duplicated keys the first instance of the key will be locked.

4 A primary key for a record that does not yet exist may be locked, but this is not the case for a secondary key.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'ftypes.i'
#include 'AM.d'

var   systemId                      : T_systemId;
      fieldNumber,keyStartIndex     : integer;
      key                           : packed array [ 1..15 ] of dataItem;
      lockLevel                     : T_lockLevel;
      reply                         : T_reply

(* type

   dataItem = char;

   T_lockLevel = ( n_, e_, m_ );

   function LockRecordKeyed(          systemId : T_systemId;
                                   fieldNumber : integer;
                           var             key : packed array [ min..max :
                                                   integer ] of dataItem;
                                  keyStartIndex : integer;
                                      lockLevel : T_lockLevel) : T_reply;
   external; *)

begin
 .
 .
 reply:=LockRecordKeyed(systemId,fieldNumber,key,1,m_);
 .
 .
end.
```

This function locks a specified record, belonging to the file indicated by the system identifier. It is possible to protect a record which does not yet exist.

reply
LockRecordPos
systemId,position,lockLevel

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
position	I		longinteger	Numerical value which indicates the record position in the file.
lockLevel	I	T_lockLevel	enumerated	Parameter which indicates the lock level requested. n : no lock (unlock if previously locked). m_ : the requesting process can read or modify the record, the others can read only. The value e_ : exclusive is not available for records.

reply	0	T_reply	variant record	Diagnostics.

Diagnostics — The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	TABLEOVERFLOW	System table full.
	NOROUTING	File inaccessible.
	RESULTSDOUBTFUL	Timeout occurred.
	OPERATIONABORTED	File not accessed due to anomaly.
DEBUG	INVALIDOPERATION	Invalid operation.
	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
FSERROR	TIMEOUT	Time expired

Characteristics — If the indicated lock is not available and the timeout declared by OpenPos has expired, the operation fails (TIMEOUT error).

Example

```
#include "systypes.i"
type dataItem = char;          (* this type is user-defined *)
#include "fstypes.i"
#include "APS.d"

var  systemId  : T_systemId;
     position  : longinteger;
     lockLevel : T_lockLevel;
     reply     : T_reply;

(* type

   T_lockLevel = ( n_, e_, m_ )

   function LockRecordPos(  systemId : T_systemId;
                            position : longinteger;
                           lockLevel : T_lockLevel) : T_reply;
   external; *)

begin
 .
 .
 reply:=LockRecordPos(systemId,2,m_);
 .
 .
end.
```

This function copies a switching procedure ($COMMHD or $COMMFD) and a system bootstrapper ($BOOT) from a specified source to a specified destination.

```
reply
MakeBoot
contextId,*sourceName*,destDirId,*destDeviceName*
```

Parameter	I/O	System-Defined Type Name	Type	Description
contextId	I	T_systemId		Identifier of source context.
sourceName	I		packed array [min1..max1: integer] of char	This parameter may be a sequence of existing names separated by slashes. It must end in blank or null. Each component is limited to 14 characters.
destDirId	I	T_systemId		Identifier of destination device.
destDeviceName	I		packed array [min2..max2: integer] of char	Name of device e.g. FL2, HD1.

reply	T_reply	variant	Diagnostics.

Diagnostics — The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.;
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.
	INVALIDOPERATION	Operation not permitted.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not available.
	INVALIDPATHNAME	Invalid pathname.

Characteristics

1 The switching procedure and bootstrapper must be in the same source directory.

2 The disk onto which they are copied must not be mounted, but must have been prepared to the point of there being a volume on the disk.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var         contextId,destDirId : T_systemId;
      sourceName,destDeviceName : packed array [ 1..20 ] of char;
                          reply : T_reply;

(* function MakeBoot(       contextId : T_systemId;
                        var sourceName : packed array [ min1..max1 :
                                           integer] of char;
                            destDirId : T_systemId;
                        destDeviceName : packed array [ min2..max2 :
                                           integer of char) : T_reply;
   external; *)

begin
 .
 .
 sourceName:='ipl/sys ';
 destDeviceName:='fl2 ';
 reply:=MakeBoot(contextId,sourceName,destDirId,destDeviceName);
 .
 .
 end.
```

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.
	INVALIDOPERATION	Operation not permitted.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not available.
	INVALIDPATHNAME	Invalid pathname.

Note

Standard 24 environments for hard disks are created by the hardware formatter.

See also:

- Make Volume

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'APS.d'

var   parentDirId : T_systemId;
      deviceName : packed array [ 1..20 ] of char;
           reply : T_reply;

(* function MakeStd24(      parentDirId : T_systemId;
                        var   deviceName : packed array [ min..max :
                                             integer ] of char) : T_reply;

begin
 .
 .
 deviceName:='fla '
 reply:=MakeStd24(parentDirId,deviceName)
 .
 .
 end.
```

This function creates a new volume to occupy a whole device or to occupy space within some existing structure.

reply
MakeVolume
dirId,*parentName*,*volumeName*,volInfo .

Parameter	I/O	System-Defined Type Name	Type	Description
dirId	I	T_systemId		Directory containing parentName.
parentName	I		packed array [min1.. max1: integer] of char	Name of device to be occupied by volume or structure within which volume is to be created.This name may have up to 14 characters not including slash, and must end in blank or null.
volumeName	I		packed array [min2.. max2: integer] of char.	Name to be given to new volume, with up to 14 characters not including slash, ending in blank or null.
volumeInfo	I	T_volume-Info	packed record	Record containing the following information:
volSize			longinteger	- length of volume in bytes

releaseNumber			integer	- number of the release
releaseLevel			integer	- release level
nrFiles			integer	- maximum number of files in volume
userCode			integer	- user identifier for volume
reply	0	T_reply	variant record	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	OUTOFDISKSPACE	Not enough space left on disk.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.
	INVALIDOPERATION	Operation not permitted.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation not compatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not available.
	INVALIDPATHNAME	Invalid pathname.
	NAMEALREADYEXISTS	Volume cannot be created.
	NAMENOTFOUND	Directory/Device name does not exist

Characteristic

Before creating a volume to occupy a whole device, the device must have a Standard 24 environment which enables sharing of space between native and emulated systems. If sharing is not envisaged the maximum space on the device should be requested for the volume.

Note

See also:

- MakeStd24

Example

```
#include "systypes.i"
type dataItem = char;          (* this type is user-defined *)
#include "fstypes.i"
#include "AMS.d"

var          dirId : T_systemId;
        parentName : packed array [ 1..20 ] of char;
        volumeName : packed array [ 1..20 ] of char;
           volInfo : T_volumeInfo;

(* type

   T_volumeInfo = packed record
                              volSize : longinteger;
                       releaseNumber : integer;
                        releaseLevel : integer;
                             nrFiles : integer;
                            userCode : integer
                  end;

   function MakeVolume(              dirId : T_systemId;
                       var  parentName : packed array [ min1..max1 :
                                           integer ] of char;
                       var  volumeName : packed array [ min2..max2 :
                                           integer ] of char;
                                volInfo : T_volumeInfo) : T_reply;
   external; *)

begin
  .
  .
  parentName:='fl2 ';
  volumeName:='pippo ';
  volInfo.volSize:=160000;
  volInfo.releaseNumber:=3;
  volInfo.releaseLevel:=0;
  volInfo.nrFiles:=100;
  volInfo.userCode:=2;
  reply:=MakeVolume(dirId,parentName,volumeName,volInfo);
  .
  .
end.
```

MOUNT (MOUNT A REMOVABLE VOLUME)

This function mounts a removable volume so that its name space becomes part of the file system tree structure.

reply
Mount
VolumeContextId,*volumeName*,deviceContextId,*deviceName*

Parameter	I/O	System-Defined Type Name	Type	Description
volumeContextId	I	T_systemId		Directory to which volumeName is to be added.
volumeName	I		packed array [min1.. max1: integer] of char	Pathname to be assigned to volume, which may consist of components separated by slashes, each limited to 14 characters, all of which may exist except the last.
deviceContextId	I	T_systemId		Directory containing deviceName.
deviceName	I		packed array [min2.. max2: integer] of char	Pathname of device, which may consist of components separated by slashes, each limited to 14 characters.
reply	I	T_reply	variant record	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.
	INVALIDOPERATION	Operation not permitted.
	INVALIDPARAMETERS	Invalid parameters.
FSERROR	DEVICENOTREADY	Device not available.
	INVALIDPATHNAME	Invalid pathname.
	NAMEALREADYEXISTS	Directory cannot be created.

Characteristic

The mounted volume can subsequently be accessed by giving volumeContextId and volumeName in input to Connect to get a system identifier of the volume. Files in the volume can then be accessed by giving their pathname relative to the volume and the system identifier of the volume in input to Connect.

Note

See also:

- Connect

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'APS.d'

var         volumeContextId : T_systemId;
                 volumeName : packed array [ 1..20 ] of char;
            deviceContextId : T_systemId;
                 deviceName : packed array [ 1..20 ] of char;
                      reply : T_reply;

(*  function Mount(         volumeContextId : T_systemId;
                    var          volumeName : packed array [ min1..max1 :
                                                integer ] of char;
                            deviceContextId : T_systemId;
                    var          deviceName : packed array [ min2..max2 :
                                                integer ] of char) : T_reply;
    external; *)

begin
 .
 .
 volumeName:='pippo ';
 deviceName:='fl2 ';
 reply:=Mount(volumeContextId,volumeName,deviceContextId,deviceName);
 .
 .
end.
```

This function opens the file indicated by the system identifier. The primitive causes an I/O session identifier to be initialised with the parameters passed by the user.

reply
OpenByte
systemId,accessMode,lockLevel,timeout

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I/O	T_systemId		File identifier.
accessMode	I	T_access-Rights	enumerated	This parameter indicates the operations which the user wants to be able to perform on the file. It may be nil_ or any combination of the following basic values: r : file only available for reading a_ : data may be written outside the current bounds of the file.

				w_: data may be modified without restrictions. x_: file may be executed.
lockLevel	1	T_lock-Level	enumerated	This parameter specifies the type of lock requested on the file n : no lock e : exclusive lock No other user may open the file. The value m : modify is not available for files.
timeout	1	T_timeout	enumerated	This parameter is a file attribute valid from the moment the file is opened until it is closed. It indicates the time in "quanta" (0..15) that the user is prepared to wait for any requested operation to be executed.
reply	0	T_reply	variant record	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDOPERATION	Operation not permitted.
	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	TIMEOUT	Time expired.

Characteristics

1 The file opened in byte mode must be a byte, positional or keyed file.

2 The system identifier given in input must be a Connect_id. In output, a session-id is returned.

3 The operations declared by the "accessMode" parameter must be a subset of those defined for the file at Connect time.

4 If the file is not available when 'Timeout' expires, the operation fails (TIMEOUT error).

5 If the parameter 'timeout' has the value 15, the user's request is for infinite time. The time quantum is defined at system configuration time.

Note

See also the following primitive:

- Connect

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'API.d'

var   systemId    : T_systemId;
      accessMode  : T_accessRights;
      lockLevel   : T_lockLevel;
      timeout     : T_timeout;
      reply       : T_reply;

(* type

   T_accessRights = ( nill_, r_, a_, ra_, w_, rw_, aw_, raw_,
                      x_, rx_, ax_, rax_, wx_, rwx_, awx_, rawx_)

   T_lockLevel = ( r_, e_, m_ );

   T_timeout = 0..15;

   function OpenByte(var   systemId : T_systemId;
                         accessMode : T_accessRights;
                          lockLevel : T_lockLevel;
                            timeout : T_timeout) : T_reply;
   external; *)

begin
 .
 .
 reply:=OpenByte(systemId,rw_,e_,1);
 .
 .
end.
```

This function opens the keyed file specified by the system identifier. The primitive causes an I/O session identifier to be initialised with the parameters passed by the user.

reply
OpenKeyed
systemId,accessMode,lockLevel,timeout

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I/O	T_systemId		File identifier.
accessMode	I	T_access-Rights	enumerated	This parameter indicates the operations which the user wants to execute on the file.See OpenByte for details.
lockLevel	I	T_lock-Level	enumerated	This parameter specifies the type of lock requested on the file: n : no lock e_: exclusive lock No other user may open the file.

timeout	I	T_timeout	enumerated	This parameter is a file attribute valid from the moment the file is opened until it is closed. It indicates the time in "quanta" (0..15) that the user is prepared to wait for any requested operation to be executed.
reply	O	T_reply	variant record	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSWARNING	BYTESTREAMOPERATION	Only byte operations possible.
	POSITIONALOPERATION	Only positional operations possible.
	NOPRIMARYINDEX	Primary index removed.
FSERROR	TIMEOUT	Time expired.

Characteristics

1 The operations declared by the "accessMode" parameter must be a subset of those defined for the file at Connect time.

2 If the file is not available when timeout expires, the operation fails (TIMEOUT error).

3 If the parameter 'timeout' has the value 15, the user's request is for infinite time. The time quantum is defined at system configuration time.

4 The identifier given in input must be a Connect-id. In output a session-id is returned.

Note

See also the following primitives:

- Connect
- OpenByte

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var   systemId    : T_systemId;
      accessMode  : T_accessRights;
      lockLevel   : T_lockLevel;
      timeout     : T_timeout;
      reply       : T_reply;

(* type

   T_accessRights = ( nill_, r_, a_, ra_, w_, rw_, aw_, raw_,
                      x_, rx_, ax_, rax_, wx_, rwx_, awx_, rawx_);

   T_lockLevel = ( n_, e_, m_ );

   T_timeout = 0..15;

   function OpenKeyed(var    systemId : T_systemId;
                          accessMode : T_accessRights;
                           lockLevel : T_lockLevel;
                             timeout : T_timeout) : T_reply;
   external; *)

begin
 .
 .
 reply:=OpenKeyed(systemId,r_,n_,0);
 .
 .
end.
```

This function opens the positional file specified by the system identifier.The primitive causes an I/O session identifier to be initialised with the parameters passed by the user.

reply
OpenPos
systemId,accessMode,lockLevel,timeout

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I/O	T_systemId		File identifier.
accessMode	I	T_access-Rights	enumerated	This parameter indicates the operations which the user wants to execute on the file.See OpenByte for details.
lockLevel	I	T_lock-Level	enumerated	This parameter specifies the type of lock requested on the file: n : no lock e : exclusive lock. No other user may open the file.

timeout	1	T_timeout	enumerated	This parameter is a file attribute valid from the moment the file is opened until it is closed. It indicates the time in "quanta" (0..15) that the user is prepared to wait for any requested operation to be executed.
reply	0	T_reply	variant record	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSWARNING	BYTESTREAM-OPERATION	Only byte operations possible.
FSERROR	TIMEOUT	Time expired.

Characteristics

1 The operations declared by the "accessMode" parameter must be a subset of those defined for the file at Connect time.

2 If the file is not available when timeout expires, the operation fails (TIMEOUT error).

3 If the parameter 'timeout' has the value 15,the user's request is for infinite time. The time quantum is defined at system configuration time.

4 The identifier given in input must be a Connect-id. In output, an I/O session-id is returned.

Note

See also the following primitive:

- Connect
- OpenByte

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var  systemId    : T_systemId;
     accessMode  : T_accessRights;
     lockLevel   : T_lockLevel;
     timeout     : T_timeout;
     reply       : T_reply;

(* type

   T_accessRights = ( nill_, r_, a_, ra_, w_, rw_, aw_, raw_,
                      x_, rx_, ax_, rax_, wx_, rwx_, awx_, rawx_);

   T_lockLevel = ( n_, e_, m_ );

   T_timeout = 0..15;

   function OpenPos(var   systemId : T_systemId;
                        accessMode : T_accessRights;
                         lockLevel : T_lockLevel;
                           timeout : T_timeout) : T_reply;
   external; *)

begin
 .
 .
 reply:=OpenPos(systemId,rw_,e_,0);
 .
 .
end.
```

This function opens the file indicated by the system identifier. The primitive causes an I/O session identifier to be initialised with the parameters passed by the user.

reply
OpenSeq
systemId,accessMode,lockLevel,timeout

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I/O	T_systemId		File identifier.
accessMode	I	T_access-Rights	enumerated	This parameter indicates the operations which the user wants to execute on the file. See OpenByte for details.
lockLevel	I	T_lock-Level	enumerated	This parameter specifies the locklevel requested on the file: n : no lock e : exclusive lock. No other user may open the file.

timeout	I	T_timeout	enumerated	This parameter is a file attribute valid from the moment the file is opened until it is closed. It indicates the time in "quanta" (0..15) that the user is prepared to wait for any requested operation to be executed.
reply	O	T_reply	variant record	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	TIMEOUT	Time expired.

Characteristics

1 The file opened in sequential mode must be a sequential or byte file.

2 The identifier given in input must be a Connect-id. In output, a session-id is returned.

3 The operations declared by the "accessMode" parameter must be a subset of those defined for the file at Connect time.

4 If the file is not available when timeout expires, the operation is aborted (TIMEOUT error).

5 If the parameter 'timeout' has the value 15, the user's request is for infinite time. The time quantum is defined at system configuration time.

Note See also the primitives:

- Connect
- OpenByte.

Example

```
#include "systypes.i"
type dataItem = char;          (* this type is user-defined *)
#include "fstypes.i"
#include "AMS.d"

var  systemId    : T_systemId;
     accessMode  : T_accessRights;
     lockLevel   : T_lockLevel;
     timeout     : T_timeout;
     reply       : T_reply;

(* type

   T_accessRights = ( nill_, r_, a_, ra_, w_, rw_, aw_, raw_,
                      x_, rx_, ax_, rax_, wx_, rwx_, awx_, rawx_);

   T_lockLevel = ( n_, e_, m_ );

   T_timeout = 0..15;

   function OpenSeq(var    systemId : T_systemId;
                        accessMode : T_accessRights;
                         lockLevel : T_lockLevel;
                           timeout : T_timeout) : T_reply;
   external; *)

begin
 .
 .
 reply:=OpenSeq(systemId,w_,e_,0);
 .
 .
end.
```

PRY (PRY FOR INFORMATION ABOUT A FILE)

This function returns information about the specified file to a specified output file.

```
reply
Pry
parentDirId,*fileName*,outputFile
```

Parameter	I/O	System-Defined Type Name	Type	Description
parentDirId	I	T_systemId		Directory containing fileName.
fileName	I		packed array [min..max: integer] of char	Pathname of file which may consist of components separated by slashes and must end in blank or null. Each component is limited to 14 characters, the last to 12 if the file is keyed or positional.
outputFile	I	T_systemId		Identifier of file to which output is to be directed.
reply	O	T_reply	variant	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDOPERATION	Operation not permitted.
	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not available.
	INVALIDPATHNAME	Invalid pathname.
	NAMENOTFOUND	Directory/File name does not exist.

Characteristics

1 The information returned includes the file type and subtype (see GetType), the allocation unit, create time, last access time, last modify time and file size. The different types of output are described in full in the "SHELL Commands Reference Manual", PRY command.

Note

See also:
- GetType.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var    parentDirId,outputFile : T_systemId;
                    fileName : packed array [ 1..20 ] of char;

(*  function Pry(     parentDirId : T_systemId;
                  var    fileName : packed array [ min..max :
                                     of char;
                       outputFile : T_systemId) : T_reply;
    external; *)

begin
 .
 .
 fileName:='pippo ';
 reply:=Pry(parentDirId,fileName,outputFile);
 .
 .
end.
```

This function associates an integer value with a specified file.

```
reply
PutXtn
systemId,offset
```

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		. File identifier.
offset	I		integer	Information associated with the file.
reply	O	T_reply	variant record	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation correctly executed.
SYSTEM	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.

Characteristics

1 The information can subsequently be accessed using GetXtn.

2 This primitive is intended for system use when managing multiple file structures. (The information is an index to the multiple file table.)

3 The information is not associated at Connect or I/O session level, but with the file itself.

Note

See also:

- GetXtn

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var      system_Id : T_systemId;
            offset : integer;
             reply : T_reply;

(* function PutXtn(       systemId : T_systemId;
                            offset : integer) : T_reply;

begin
 .
 .
 reply:=PutXtn(systemId,3);
 .
 .
end.
```

This function reads the number of bytes indicated by the 'dataSize' parameter from the file identified by the system identifier.

reply
ReadByte
systemId,position,*data*,startIndex,dataSize,*lengthRead*

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
position	I		longinteger	Byte offset within the file at which to start the read operation.
data	O		packed array [min..max: integer] of data-Item	Array which will contain the data. 'dataItem' is defined by the user and its length must be of 8 bits, e.g. char, byte (-128..127).
startIndex	I		integer	Integer within the 'data' range. It indicates the initial location in the data array where the transferred data should start.

dataSize	I		integer	Integer which indicates the number of bytes to be transferred into 'data'.
lengthRead	O		integer	Number of bytes effectively read.
reply	O	T_reply	variant record	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDOPERATION	Invalid operation.
	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not available.
	ENDOFFILE	End of file encountered.
	OUTOFBOUNDS	Specified 'position' is not within the file bounds.

Characteristics

1 The file must be open in byte mode and may be a byte, keyed or positional file.

2 The ENDOFFILE error can only occur when:
'lengthRead' 'dataSize', i.e. 'position' valid but
closer then 'dataSize' to the end of the file.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var   systemId                              : T_systemId;
      position                              : longinteger;
      data                                  : packed array [ 1..16 ] of
                                                dataItem;
      startIndex,dataSize,lengthRead        : integer;
      reply                                 : T_reply;

(* type

   dataItem = char;

   function ReadByte(          systemIc : T_systemId;
                               position : longinteger;
                     var           data : packed array [ min..max :
                                            integer ] of dataItem;
                           startIndex : integer;
                             dataSize : integer;
                     var   lengthRead : integer) : T_reply;
   external; *)

begin
 .
 .
 reply:=ReadByte(systemId,40,data,2,10,lengthRead);
 .
 .
end.
```

READINFOBYTE (READ INFORMATION ABOUT A BYTE FILE)

This function reads the attributes of the byte file identified by the system identifier.

reply
ReadInfoByte
systemId,*fileInfo*

Parameters	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
fileInfo	O	T_readInfo	packed record	Record containing the following information:
allocation-Unit			longinteger	- memory allocation unit
createTime			longinteger	- file creation date
lastAccess			longinteger	- date of the last access to the file
lastModify			longinteger	- date of the last file modification
userType			integer	- user-defined file identifier.
reply	O	T_reply	variant record	Diagnostics.

Diagnostics — The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.

Characteristic — It is not necessary to open the file before invoking this primitive.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var  systemId : T_systemId;
     fileInfo : T_readInfo;
     reply    : T_reply;

(* type

   T_readInfo = packed record
                    allocationUnit : longinteger;
                        createTime : longinteger;
                        lastAccess : longinteger;
                        lastModify : longinteger;
                          userType : integer
                end;

   function ReadInfoByte(     systemId : T_sytemId;
                          var fileInfo : T_readInfo) : T_reply;

   external; *)

begin
 .
 .
 reply:=ReadInfobyte(systemId,fileInfo);
 .
 .
end.
```

READINFOKEYED (READ INFORMATION ABOUT KEYED FILE)

This function reads the attributes of the keyed file indicated by the system identifier and by an index sequence number.

```
reply
ReadInfoKeyed
systemId,fieldNumber,*fileInfo*
```

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
fieldNumber	I		integer	Integer which indicates the index sequence number.
fileInfo	O	T_readIn-foK	packed record	Record containing the following information (see also Type Definitions earlier in this chapter).
allocationUnit			longinteger	- memory allocation unit
createTime			longinteger	- date of file creation
lastaccess			longinteger	- date of the last access to the file

lastModify			longinteger	- date of the last file modification
userType			integer	- user-defined file identifier
recordLength			integer	- record length
splitStrategy		T_split-Strategy	enumerated	- split strategy.
reply	0	T_reply	variant record	Diagnostics.

Diagnostics — The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDOPERATION	Invalid operation.
	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.

Characteristics It is not necessary to open the file before invoking the primitive.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'XMS.d'

var   systemId    : T_systemId;
      fieldNumber : integer;
      fileInfo    : T_readInfoK;
      reply       : T_reply;

(* type

   T_readInfo = packed record
                      allocationUnit : longinteger;
                          createTime : longinteger;
                          lastAccess : longinteger;
                          lastModify : longinteger;
                            userType : integer
                end;

   T_readInfoP = packed record
                          readInfo : T_readInfo;
                      recordLength : integer
                 end;

   T_splitStrategy = ( s50_, s?0_, s100_ );

   T_readInfoK = packed record
                            readInfo : T_readInfoP;
                       splitStrategy : T_splitStrategy
                 end;

   function ?eadInfoKeyed(        systemId : T_systemId;
                              fieldNumber : integer;
                          var    fileInfo : T_readInfoK) : T_reply;
   external; *)

begin
 .
 .
 reply:=?eacInfoKeyed(systemId,C,fileInfo);
 .
 .
end;
```

READINFOPOS (READ INFORMATION ABOUT A POSITIONAL FILE)

This function reads the attributes of the positional file identified by the system identifier.

reply
ReadInfoPos
systemId,*fileInfo*

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
fileInfo	O	T_readInfoP	packed record	Record containing the following information (see also Type Definitions earlier in this chapter):
allocationUnit			longinteger	- memory allocation unit.
createTime			longinteger	- date of file creation.
lastaccess			longinteger	- date when the file was last accessed.
lastModify			longinteger	- date of the last file modification.
userType			integer	- user-defined identifier.

recordLength			integer	- record length.
reply	0	T_reply	variant record	Diagnostics

Diagnostics — The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDOPERATION	Invalid operation.
	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.

Characteristic — It is not necessary to open the file before invoking this primitive.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var  systemId : T_systemId;
     fileInfo : T_readInfoP;
     reply    : T_reply;

(* type

   T_readInfo = packed record
                    allocationUnit : longinteger;
                        createTime : longinteger;
                        lastAccess : longinteger;
                        lastModify : longinteger;
                          userType : integer
                end;

   T_readInfoP = packed record
                           readInfo : T_readInfo;
                      recordlength : integer
                 end;

   function ReadInfoPos(     systemId : T_systemId;
                         var fileInfo : T_readInfoP) : T_reply;
   external; *)

begin
 .
 .
 reply:=ReadInfoPos(systemId,fileInfo);
 .
 .
end.
```

READKEYED (READ A RECORD FROM A KEYED FILE)

This function reads the (first) record identified by the key in the file identified by the system identifier, using the index specified by the index sequence number.

reply
ReadKeyed
systemId,fieldNumber,*key*,keyStartIndex,*data*,dataStartIndex

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
fieldNumber	I		integer	Integer which indicates the index sequence number.
key	I		packed array [min1..max1: integer of data-ltem	Array containing the key. 'dataltem' is defined by the user and its length must be of 8 bits, e.g. char, byte (-128..127)
keyStart-Index	I		integer	Integer within the 'key' range. It indicates the initial location of the key in the array.

data	O		packed array [min2.. max2: integer] of dataItem	Array which will contain the data. For dataItem see 'key'.
dataStart-Index	I		integer	Integer within the 'data' range. It indicates the initial location in the 'data' array where the transferred data will start.
reply	O	T_reply	variant record	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDOPERATION	Invalid operation.
	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not available.
	KEYNOTFOUND	The specified key does not exist.

Characteristics

1 The keyed file must be open in keyed mode.

2 The length of the key is taken to be as specified at file creation time.

3 The primitive will not work when only part of the key is given.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'APS.d'

var systemIc                                          : T_systemId;
    fieldNumber,keyStartIndex,dataStartIndex          : integer;
    key                                               : packed array [ 1..16 ]
                                                          of dataItem;
    data                                              : packed array [ 1..100 ]
                                                          of dataItem;
    reply                                             : T_reply;

(* type

   dataItem = char;

   function ReadKeyed(          systemId : T_systemId;
                          fieldNumber : integer;
                     var            key : packed array [ min1..max1 :
                                            integer ] of dataItem;
                         keyStartIndex : integer;
                     var           data : packed array [ min2..max2 :
                                            integer ] of dataItem;
                        dataStartIndex : integer) : T_reply;
   external; *)

begin
 .
 .
 reply:=ReadKeyed(systemId,C,key,1,data,1);
 .
 .
end.
```

READPOS (READ A RECORD BY ITS POSITION)

This function reads the record in the specified position in the file indicated by the system identifier.

```
reply
ReadPos
systemId,position,*data*,startIndex
```

Parameters	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
position	I		long-integer	Numerical value which indicates the relative position of the record to be read.
data	O		packed array [min..max: integer] of dataItem	Array which will contain the data. 'dataItem' is defined by the user and its length must be of 8 bits, e.g. char, byte (-128..127)

startIndex	1		integer	Integer within the 'data' range. It indicates the initial location in the data array, where transferred data will start.
reply	0	T_reply	variant record	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation execution correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDOPERATION	Invalid operation.
	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not ready.
	OUTOFBOUNDS	The record position is outside the bounds of the file.
	RECORDNOTFOUND	No valid record at the specified position.

Characteristics

1 The file must be open in positional mode and must be a keyed or positional file.

2 The RECORDNOTFOUND error implies that the specified position was within the bounds of the file, as otherwise the OUTOFBOUNDS error would have occurred.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var   systemId    : T_systemId;
      position    : longinteger;
      data        : packed array [ 1..16 ] of dataItem;
      startIndex  : integer;
      reply       : T_reply;

(* type

   dataItem = char;

   function ReadPos(       systemId : T_systemId;
                           position : longinteger;
                     var        data : packed array [ min..max :
                                         integer ] of dataItem;
                          startIndex : integer) : T_reply;
   external; *)

begin
 .
 .
 reply:=ReadPos(systemId,1),data,1);
 .
 .
end.
```

READSEQ (READ A RECORD IN SEQUENTIAL MODE)

This function reads the next data in the file identified by the system identifier.

reply
ReadSeq
systemId,*data*,startIndex,dataSize,*lengthRead*

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
data	O		packed array [min..max: integer] of dataItem	Array which will contain the data. 'dataItem' is defined by the user and its length must be of 8 bits, e.g. char, byte (-128..127)
startIndex	I		integer	Integer within the 'data' range. It indicates the location in 'data' at which the transferred data will start.

dataSize	1		integer	Integer which indicates the maximum number of bytes to be transferred into 'data'.
lengthRead	0		integer	Integer which indicates the effective number of bytes transferred.
reply	0	T_reply	variant record	Diagnostics.

Diagnostics The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSWARNING	INCOMPLETERECORD	Incomplete record read.
FSERROR	DEVICENOTREADY	Device not ready.
	ENDOFFILE	End of file encountered.

Characteristics

1 The file must be open in sequential mode and must be a sequential or byte file.

2 This primitive is used to read data from sequential and byte files in sequence and transfer them to a user area starting from the location indicated. N.B. min + startIndex + dataSize _= max .

3 When reading from a byte file, if the number of bytes read is less than the number requested, the data is still read and lengthRead indicates the number of bytes actually read.

4 If the sequential file is a terminal, the record is a line on the screen. The number of characters read includes the linefeed character.

5 At the end of the read operation the current position will be increased by lengthRead. The current position can be set using the SeekP primitive and its value can be retrieved using the SenseP primitive. It gets changed by the WriteSeq primitive as well as by the ReadSeq primitive.

Note

See also:

- SeekP
- SenseP
- WriteSeq.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var  systemId                          : T_systemId;
     data                              : packed array [ 1..16 ]
                                                 of dataItem;
     startIndex,dataSize,lengthRead    : integer;
     reply                             : T_reply;

(* type

   dataItem = char;

   function ReadSeq(       systemId : T_systemId;
                    var        data : packed array [ min..max :
                                        integer ] of dataItem;
                          startIndex : integer;
                            dataSize : integer;
                    var lengthRead   : integer) : T_reply;
   external; *)

begin
 .
 .
 reply:=ReadSeq(systemId,data,1,8,lengthRead);
 .
 .
end.
```

REMOVE (REMOVE A FILE OR DIRECTORY)

This function removes the file or empty directory indicated by 'fileName' and releases all the memory which had been allocated.

reply
Remove
parentDirId,*fileName*

Parameter	I/O	System-Defined Type Name	Type	Description
parentDirId	I	T_systemId		System identifier of directory containing "fileName".
fileName	I		packed array [min..max: integer] of char	This parameter may be a sequence of existing names, separated by a "/" It must not begin with a "/" and must end with a ' '. Each intermediate component may have a maximum of 14 characters and the whole name (max - min + 1) is limited to 60 characters.

reply	0	T_reply	variant record	Diagnostics.

Diagnostics The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not available.
	INVALIDPATHNAME	Invalid pathname.
	NAMENOTFOUND	File name does not exist.
	NOTEMPTYDIRECTORY	Directory has not been removed because it is not empty.

Characteristics

1 If the file is in use, it is only marked for removal and removed later when it is no longer being used.

2 After the file has been removed, subsequent attempts to Connect that file give a 'NAMENOTFOUND' error.

3 If the file is an alias file, the aliased file is unaffected (only the alias link disappears).

Note

See also:
- Connect

Example

```
#include "systypes.i"
type dataItem = char;          (* this type is user-defined *)
#include "fstypes.i"
#include "AMS.d"

var parentDirId : T_systemId;
    fileName    : packed array [ 1..5 ] of char;
    reply       : T_reply;

(* function Remove(     parentDirId : T_systemId;
                    var     fileName : packed array [ min..max :
                                       integer ] of char) : T_reply;
   external; *)

begin
 .
 .
 fileName:="anna ";
 .
 reply:=Remove(parentDirId,fileName);
 .
 .
end.
```

RENAME (RENAME A FILE OR DIRECTORY)

This function changes the name of the specified file or directory in accordance with a stated option and predefined rules.

reply
Rename
oldDirId,*oldFileName*,newDirId,*newFileName*,existAction

Parameter	I/O	System-Defined Type Name	Type	Description
oldDirId	I	T_systemId		Identifier of directory containing oldFileName.
oldFileName	I		packed array [min1..max1: integer] of char	This may be a sequence of existing names separated by slashes and ending in blank or null. Each component is limited 14 characters, the last to 12 if the file is positional or keyed.
newDirId	I	T_systemId		Identifier of directory containing newFileName.
newFileName	I		packed array [min2..max2: integer] of char	see oldFileName

existAction			enumerated	Option selected in the event that newFileName already exists, possible values: - rename_ - abort_ see Characteristic 2.
reply	0	T_reply	variant record	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	OUTOFDISKSPACE	Not enough space left on disk.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDOPERATION	Operation not permitted.
	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not available.
	INVALIDPATHNAME	Invalid pathname.
	NAMEALREADYEXISTS	Directory/File cannot be created.
	NAMENOTFOUND	Directory/File name does not exist.

Characteristics

1 If newFileName does not exist, the renaming is carried out regardless of the types of object involved.

2 The action(s) taken if newFileName does exist depend on the types of the objects involved as shown in the table below.

Object having old name	Object having new name	Option ('existAction')	Action(s)
disk file (byte, pos or key) or device (seq)	disk file	rename_	File renamed
		abort_	Operation aborted
	device (dev) directory (dir) or volume(vol)	any	Operation aborted
directory or volume	disk file	any	File renamed
	dev	any	Operation aborted
	dir or vol	any	newFileName removed; object renamed.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var     oldDirId,newDirId : T_systemId;
        oldFileName : packed array [ 1..20 ] of char;
        newFileName : packed array [ 1..20 ] of char;
        existAction : T_existAction;

(* type

   T_existAction = (rename_, abort_ );

   function Rename(        oldDirId : T_systemId;
                   var oldFileName : packed array [ min1..max1 :
                                       integer ] of char;
                       newDirId : T_systemId;
                   var newFileName : packed array [ min2..max2 :
                                       integer ] of char;
                       existAction : T_existAction) : T_reply;
   external; *)

begin
 .
 .
 oldFileName:='pippo ';
 newFileName:='fred ';
 reply:=Rename(oldDirId,oldFileName,newDirIo,newFileName,abort_);
 .
 .
end.
```

SEEKP (SET CURRENT POSITION OF A BYTE FILE)

This function sets the current position of the byte file indicated by the system identifier to a specified value.

currentPos
SeekP
systemId,offset

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
offset	I		longinteger	Byte position to which current position is to be set.
currentPos	O		longinteger	Actual current position after the operation.

Characteristics

1 The primitive does not return an error code but any error will result in an invalid current position (currentPos negative).

2 The requested current position may be outside the bounds of the file.

3 The current position is for use in input to ReadSeq or WriteSeq, to work on the file sequentially from a predefined position.

Note

See also:

- ReadSeq
- WriteSeq.

Example

```
#include "systypes.i"
type dataItem = char;          (* this type is user-defined *)
#include "fstypes.i"
#include "AMS.d"

var          systemId : T_systemId;
               offset : longinteger;
           currentPos : longinteger;

(* function SeekP(   systemId : T_systemId;
                       offset : longinteger) : currentPos;
   external; *)

begin
 .
 .
 currentPos:=SeekP(systemId,10);
 .
 .
end;
```

SENSEP (FIND THE CURRENT POSITION OF A BYTE FILE)

This function returns the current position of the byte file specified by the system identifier.

currentPos
SenseP
systemId

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
currentPos	O		longinteger	Current position.

Characteristics

1 The primitive does not return an error code but any error will result in an invalid current position.

2 The current position is for use in input to ReadSeq or WriteSeq, to work on the file sequentially from a predefined position.

Note

See also:
- ReadSeq
- WriteSeq.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var      systemId : T_systemId;
       currentPos : longinteger;

(* function SenseP(      systemId : T_systemId) : currentPos;
   external; *)

begin
 .
 .
 currentPos:=SenseP(systemId);
 .
 .
end.
```

This function reduces the size of the file specified by the system identifier. (It may not be used to increase the size.)

```
reply
SetBoundsByte
systemId,address,bounds
```

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
address	I	T_address		Address returned by ComputeByte. It indicates the new upper or lower limit of the file.

bounds	I	T_bounds	enumerated	Parameter which can contain one of the following values: bof_: beginning of file. This fixes a new lower limit of the file. All the data preceding 'address' are removed eof_: end of file. This fixes a new upper limit of the file. All the data coming after 'address' are deleted.
reply	O	T_reply	variant record	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDOPERATON	Invalid operation.
	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not available.

Characteristics

1 To delete all the data from a file, it is possible to calculate (via ComputeByte) an address for input to SetBoundsByte which is outside the present limits of the file. This address must either be smaller than the beginning of the file (in which case the input value for SetBoundsByte should be eof) or bigger than the address of the end of the file (in which case the input value for SetBoundsByte should be bof).

Note See also the following primitives:

- ComputeByte
- EndByte
- BeginByte

Example

```
#include "systypes.i"
type dataItem = char;          (* this type is user-defined *)
#include "fstypes.i"
#include "AMS.d"

var  systemId : T_systemId;
     address  : T_address;
     bounds   : T_bounds;
     reply    : T_reply;

(* type

   dummyRec = record
              end;

   T_address = ^dummyRec;

   T_bounds = ( bof_, eof_ );

   function SetBoundsByte( systemId : T_systemId;
                           address : T_address;
                           bounds : T_bounds) : T_reply;
   external; *)

begin
 .
 .
 reply:=SetBoundsByte(systemId,address,eof_);
 .
 .
end.
```

This function completely empties the file specified by the system identifier as accessed via the index specified by the index sequence number. (No other use of this function is currently implemented.)

```
reply
SetBoundsKeyed
systemId,fieldNumber,address,bounds
```

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
fieldNumber	I		integer	Integer which indicates the sequence number of the index.
address	I	T_address		Address (within index) returned by ComputeKeyed. or TransformKeyed. It indicates the new upper or lower limit of the file.

bounds	I	T_bounds	enumerated	Parameter which can contain one of the following values: bof_: beginning of file. This fixes a new beginning of the file. All the data preceding 'address' are removed. eof_: end of file. All the data coming after 'address' are deleted.
reply	O	T_reply	variant record	Diagnostics.

Diagnostics The primitive returns one of the following replies (see Error Types earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDOPERATION	Invalid operation.
	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
	NOTYETIMPLEMENTED	Operation imcompatible with the current implementation.
FSERROR	DEVICENOTREADY	Device not available.

Characteristics

1 To delete all the data from the file, the address input to SetBoundsKeyed must be outside the present limits of the file. This address must either be smaller than the beginning of the file (in which case the input value of "bounds" is eof) or bigger than the address of the end of the file (in which case the input value of bounds should be bof).

Note

See also the following primitives:

- ComputeKeyed
- EndKeyed
- BeginKeyed
- TransformKeyed.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var  systemId    : T_systemId;
     fieldNumber : integer;
     address     : T_address;
     bounds      : T_bounds;
     reply       : T_reply;

(* type

   dummyRec = record
              end;

   T_address = ^dummyRec;

   T_bounds = ( bof_, eof_ );

   function SetBoundsKeyed(    systemId : T_systemId;
                           fieldNumber : integer;
                               address : T_address;
                                bounds : T_bounds) : T_reply;
   external; *)

begin
 .
 .
 reply:=SetBoundsKeyed(systemId,0,address,eof_);
 .
 .
end.
```

This function completely empties the file specified by the system identifier. (No other use of this function is currently implemented.)

reply
SetBoundsPos
systemId,address,bounds

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier
address	I	T_address		Address returned by ComputePos or TransformPos. It indicates the new upper or lower limit of the file.

bounds	1	T_bounds	enumerated	Parameter which can contain one contain one of the following values: bof_: beginning of file. This fixes a new lower limit of the file. All the data preceding 'address' are removed. eof_: end of file. This fixes a new upper limit of the file. All the data coming after 'address' are deleted.
reply	0	T_reply	variant record	Diagnostics.

Diagnostics The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDOPERATION	Invalid operation.
	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
	NOTYETIMPLEMENTED	Operation incompatible with the current implementation.
FSERROR	DEVICENOTREADY	Device not available.

Characteristic

To delete all the data from a file, it is possible to calculate (via ComputePos or TransformPos) an address (which will be the input value for SetBounds) outside the present limits of the file. This address must either be smaller than the beginning of the file (in which case the input value of "bounds" should be eof) or bigger than the address of the end of the file (in which case the input value of "bounds" should be bof).

Note

See also the following primitives:

- ComputePos
- TransformPos
- EndPos
- BeginPos

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var  systemId : T_systemId;
     address  : T_address;
     bounds   : T_bounds;
     reply    : T_reply;

(* type

   dummyRec = record
              end;

   T_address = ^dummyRec;

   T_bounds = ( bof_, eof_ );

   function SetBoundsPos( systemId : T_systemId;
                          address  : T_address;
                          bounds   : T_bounds) : T_reply;
   external; *)

begin
 .
 .
 reply:=SetBoundsPos(systemId,address,bof_);
 .
 .
end.
```

This function requests a number of buffers of a specified size to be exclusively associated with a specified file.

```
reply
SetBufferOptions
systemId,nrPrivate,sizePrivate
```

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
nrPrivate	I		integer	Number of buffers requested.
sizePrivate	I		integer	Length in bytes of buffers requested.
reply	O	T_reply	variant record	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	OUTOFDISKSPACE	Not enough space left on disk.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDOPERATION	Operation not permitted.
	INVALIDID	Invalid identifier.

Characteristics

1 The use of private buffers is an alternative to the normal means of carrying out I/O, which is by means of the system buffer pool.

2 The system identifier given in input must be a Connect-id.

3 The file must not already be open or the operation will fail.

4 The use of this primitive more than once on the same connected file, to change to a different private buffer pool, is not permitted.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMC.d'

var       systemId : T_systemId;
      . nrPrivate : integer;
     sizePrivate : integer;
           reply : T_reply;

(* function SetBufferOptions    systemId : T_systemId;
                                nrPrivate : integer;
                               sizePrivate : integer) : T_reply;
   external; *)

begin
 .
 .
 nrPrivate:=10;
 sizePrivate:=256;
 reply:=SetBufferOptions(systemId,nrPrivate,sizePrivate);
 .
 .
end.
```

This function sets the I/O mode for a specified file.

reply
SetIoOptions
systemId,iotype,seqHint

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier (Connect_id or Open_id).
iotype	I	T_iotype	enumerated	I/O mode (see Characteristics 1 to 5), possible values: - async_: asynchronous writes - sync_: synchronous writes - directwv_: direct writes with verify - directw_: direct writes without verify - directv_: direct verify - unchanged_: confirms existing mode (see Characteristic 6)
seqHint	I		boolean	If TRUE, indicates the file will be handled sequentially (see Characteristic 6).

reply	0	T_reply	variant record	Diagnostics.

Diagnostics The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.

Characteristics

1 The default I/O mode is asynchronous, in which the user process does not wait for write operations to be completed. When changes to file bounds are caused by write operations (i.e. when data is appended), these will also be done asynchronously.

2 Synchronous mode, on the other hand, ensures that date updates and the setting of bounds will be done in the appropriate order.

3 The asynchronous and synchronous modes both make use of buffers for temporary storage of data; the direct modes do not unless absolutely necessary (see 4 below).

4 The direct modes cause I/O to operate directly on user space (no buffering), and the system disables memory compaction during I/O operations. These modes are recommended only for large data transfers, and I/O requests should be lined up on disk sector boundaries otherwise some buffering will be necessary. Not all of these options are available for all types of device.

5 The following types of I/O are always synchronous:

- updating volumes, directories and their associated support files
- writing to user files via a WriteInfo primitive
- changing bounds via a SetBounds primitive
- updating last access and modify times when a file is closed.

6 The "sequential hint" can be specified for byte or positional (no deletion) files only. The reason for notifying the system that a file will be accessed sequentially is so that it can optimise throughput by the pre-fetching of blocks. When SetIoOptions is used solely for this purpose, the I/O mode can be left as it was using the unchanged_ value for iotype.

Note

See also:

- SetBounds... primitives
- WriteInfo... primitives
- Close... primitives

Example

```
#include "systypes.i"
type dataItem = char;          (* this type is user-defined *)
#include "fstypes.i"
#include "AFS.d"

var          systemId : T_systemId;
               iotype : T_iotype;
             seqHint : boolean;
               reply : T_reply;

(* type

   T_iotype = ( async_, sync_, directuv_, directu_, directv_, unchanged_ );

   function SetIOOptions(        systemId : T_systemId;
                                   iotype : T_iotype;
                                  seqHint : boolean) : T_reply;
   external; *)

begin
 .
 .
reply:=SetIOOptions(systemId,directu_,false);
 .
 .
end.
```

TRANSFORMKEYED
(TRANSFORM A KEY INTO A POSITION AND AN ADDRESS)

This function returns the position and address of a valid record within a file given a key and a specified option. It also returns the record's key which may or may not be the input key.The input key relates to the file defined by the file identifier, accessed via the index defined by the index sequence number.

```
posRec
TransformKeyed
systemId,fieldNumber,*searchKey*,searchKeySI,searchKeySize,
searchPos,exactness,*address*,*foundKey*,foundKeySI
```

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier
fieldNumber	I		integer	Integer indicating the sequence number of the index containing the key.
searchKey	I		packed array [min..max: integer] of dataltem	Array which contains the key. 'dataltem' is defined by the user and its length must be 8 bits, e.g. char, byte (-128..127)

searchKeySI	I		integer	Integer in the same range as for 'searchKey'. It indicates the initial location of the key in the array.
searchKeySize	I		integer	Integer which indicates the length of the key.
searchPos	I		longinteger	Numeric value indicating the presumed position of the record containing the key. (Hint only).
exactness	I	T_exactness	enumerated	This parameter is used together with the given key, to return the position of a record. It contains one of the following values: ge_: asks for the position of the record containing the first key found whose value is greater than or equal to the one supplied.

			eq_: asks for the position of the record containing the key supplied. If the key does not exist, the primitive returns an invalid position (a negative number). BeginKeyed and EndKeyed must be used to determine the current limits of the file. gt_: asks for the position of the record containing the first key found which is greater than the one supplied. next_: asks for the position of the record containing the next valid key. previous_: asks for the position of the record containing the previous valid key
address	I/O	T_address	Address of the record whose position is returned. This value can be supplied in input to ComputeKeyed or to SetBoundsKeyed.

foundKey	O	packed array [min2 ..max2: integer] of dataltem	Array containing the key associated with the record whose address and position are returned. For dataltem see 'searchKey'.
foundKeySl	I	integer	Integer in the same range as for 'foundKey'. It indicates the start location within the array of the key.
posRec	O	longinteger	Position of the record containing foundKey. This value can only be used in input to ReadPos.

Characteristics

1 This primitive can have key prefixes as input. If the key length in input is less than the real key length, then, when searching, the real key length is assumed to be truncated to the input length.

2 If the key may be a duplicate (as it could be if fieldNumber represents a secondary index), 'searchPos' may be used to indicate a particular instance of the 'searchKey'. If the index indicated by fieldNumber does not allow duplicated keys or if searchPos is a negative value, it is ignored. If the searchKey and searchPos combination does not exist, processing will start from the first instance of 'searchKey'.

3 TransformKeyed can be used alternately with ReadPos to read a file sequentially.

4 The position of the record should not be used as input to DeletePos or WritePos as these primitives do not update the indexes.

5 It is not necessary to open the file before invoking TransformKeyed.

6 The address parameter may also be used in input to this function as a hint to the possible location of the specified key. If the key is not found in that location, then the normal tree-traversing algorithm will be used. Thus if TransformKeyed is being repeatedly used in a loop so as to sequentially handle the file, then the user may help the system to optimise its use. This is because the address returned by the previous Transform may be used in input to the next Transform, thus minimising tree-traversing and hence disc accesses.

7 Any error condition will result in an invalid record position being returned.

Note

See also the following primitives:

- ComputeKeyed
- SetBoundsKeyed
- ReadPos
- BeginKeyed
- EndKeyed

Example

```
#include "systypes.i"
type dataItem = char;          (* this type is user-defined *)
#include "fstypes.i"
#include "AMS.d"

var   systemId                              : T_systemId;
      fieldNumber,searchKeySI,              : integer;
      searchKeySize,foundKeySI              : integer;
      searchKey                             : packed array [ 1..7 ]
                                                  of dataItem;
      searchPos                             : longinteger;
      exactness                             : T_exactness;
      address                               : T_address;
      foundKey                              : packed array [ 1..10 ]
                                                  of dataItem;
      posRec                                : longinteger;

(* type

   dataItem = char;

   dummyRec = record
              end;

   T_address = ^dummyRec;

   T_exactness = ( eq_, ge_, gt_, next_, previous_ );

   function TransformKeyed(     systemId : T_systemId;
                            fieldNumber : integer;
                        var  searchKey : packed array [ min1..max1 :
                                             integer ] of dataItem;
                          searchKeySI : integer;
                       searchKeySize : integer;
                           searchPos : longinteger;
                           exactness : T_exactness;
                        var    address : T_address;
                        var   foundKey : packed array [ min2..max2 :
                                             integer ] of dataItem;
                          foundKeySI : integer) : longinteger;
   external; *)

begin
 .
 .
 posRec:=TransformKeyed(systemId,fieldNumber,searchKey,1,3,2,next_,
                                                address,foundKey,1);
 .
 .
end.
```

TRANSFORMPOS
(TRANSFORM A POSITION INTO A POSITION AND AN ADDRESS)

This function gives the position and the address of a valid record starting from a possible position within a file.

position1
TransformPos
systemId,position,exactness,*address*

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
position	I		longinteger	Numerical value indicating a possible position within the file.
exactness	I	T_exactness	enumerated	This parameter is used together with the value given in 'position', to find the required record. It contains one of the following values:

ge_: asks for the position of the first record whose position is greater than or equal to the one supplied.

eq_: asks for the exact position of the record corresponding to the "position" supplied. If the "position" does not exist, the primitive returns an invalid position (a negative number). BeginPos and EndPos must be used to determine the current limits of the file.

gt_: asks for the position of the first valid record whose position is greater than the one supplied.

next_: the primitive first applies 'ge_' and then returns the position of the next valid record.

				previous_: as in 'next', except that it returns the position of the preceding valid record.
address	0	T_address		Address of the required record. This value can then be used as input to ComputePos or SetBoundsPos.
position1	0		longinteger	Required record position. This value can only be passed to ReadPos.

Characteristics

1 This primitive calculates the position and the address of a record given a possible position in the file and the option indicated by the parameter 'exactness'.

2 It is not necessary to open the file before invoking TransformPos.

3 Any error condition will result in an invalid record position being returned.

Note

See also the following primitives:

- ReadPos
- ComputePos
- SetBoundPos
- BeginPos
- EndPos

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include '4GS.d'

var systemId              : T_systemId;
    position,position1    : longinteger;
    exactness             : T_exactness;
    address               : T_address;

(* type

   T_exactness = ( eq_, ge_, gt_, next_, previous );

   dummyRec = record
              end;

   T_address = ^dummyRec;


   function TransformPos(      systemId : T_systemId;
                              position : longinteger;
                             exactness : T_exactness;
                         var    address : T_address) : longinteger;
   external; *)

begin
 .
 .
 position1:=TransformPos(systemId,4,eq_,address);
 .
 .
end.
```

UNLOCKALLRECORDS

This function removes any locks on the records in the keyed or positional file identified by the system identifier.

reply
UnlockAllRecords
systemId

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
reply	O	T_reply	variant record	Diagnostics.

Diagnostics The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation correctly executed.
SYSTEM	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.
	INVALIDOPERATION	Invalid operation.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var  systemId : T_systemId;
     reply    : T_reply;

(* function UnLockAllRecords(systemId : T_systemId) : T_reply;

   external; *)

begin
 .
 .
 reply:=UnLockAllRecords(systemId);
 .
 .
end.
```

This function cancels the effect of a previous Mount operation for a removable volume.

reply
UnMount
volumeContextId,*volumeName*

Parameter	I/O	System-Defined Type Name	Type	Description
volumeContextId	I	T_systemId		Identifier of volume containing removable volume.
volumeName	I		packed array [min..max: integer] of char	This parameter may be a sequence of existing names separated by slashes and ending in blank or null. Each component is limited to 14 characters.
reply	O	T_reply	variant record	Diagnostics.

Diagnostics — The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation correctly executed.
SYSTEM	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDID	Invalid identifier.
	INVALIDOPERATION	Operation not permitted.
	INVALIDPARAMETERS	Invalid parameters.
FSERROR	DEVICENOTREADY	Device not available.
	INVALIDPATHNAME	Invalid pathname.
	NAMENOTFOUND	Parent volume name does not exist.

Characteristic All files in the volume that were connected are automatically disconnected by the system when the volume is unmounted.

Note See also:
- Mount

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'APS.d'

var     volumeContextId : T_systemId;
            volumeName : packed array [ 1..20 ] of char;
                 reply : T_reply;

(* function Unmount(    volumeContextId : T_systemId;
                    var      volumeName : packed array [ min..max :
                                          integer of char) : T_reply;
   external; *)

begin
 .
 .
 reply:=Unmount(volumeContextId,'pippo ');
 .
 .
end.
```

WRITEBYTE (WRITE BYTES INTO A BYTE FILE)

This function writes bytes into the file identified by the system identifier.

reply
WriteByte
systemId,position,*data*,startIndex,dataSize,writeMode

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
position	I		long-integer	Numerical value which indicates the byte offset within the file where the writing of the data is to begin.
data	I		packed array [min..max: integer] of dataItem	Array containing the data to be written. 'dataItem' is defined by the user and its length must be of 8 bits, e.g. char, byte (-128..127)

startIndex	I		integer	Integer in the same range as for 'data'. It indicates the location at which the data from 'data' area is to start.
dataSize	I		integer	Number of bytes to be written to the file.
writeMode	I	T_writeMode	enumerated	Defines the mode of writing: rewrite_: to update existing data. It is not possible to write outside the bounds of the file. newwrite_: to add new data to the file. It is not possible to modify existing data. The file is extended as necessary. alwayswrite_: to write data without any restrictions.
reply	O	T_reply	variant record	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation correctly executed.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	OUTOFDISKSPACE	No more space on disk.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDOPERATION	Invalid operation.
	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not available.

Characteristics

1 This primitive transfers the data from the user area into the file starting from the position indicated by the user.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AFS.d'

var   systemId                : T_systemId;
      position                : longinteger;
      data                    : packed array [ 1..1o ] of dataItem;
      startIndex,dataSize     : integer;
      writeMode               : T_writeMode;
      reply                   : T_reply;

(* type

   dataItem = char;

   T_writeMode = ( rswrite_, newwrite_, alwayswrite_ );

   function writeByte(      systemId : T_systemId
                            position : longinteger;
                        var      data : packed array [ min..max :
                                          integer ] of dataItem;
                          startIndex : integer;
                            dataSize : integer;
                           writeMode : T_writeMode) : T_reply;
   external; *)

begin
 .
 .
 reply:=WriteByte(systemId,1,data,1,16,alwayswrite_);
 .
 .
end.
```

This function writes information about the file specified by the system identifier.

reply
WriteInfoByte
systemId,fileInfo

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
fileInfo	I	T_writeInfo	packed record	Record which specifies the attributes to be written.
writeAllocationUnit			boolean	- this boolean has the value 'true' if 'allocationUnit' is to be written; it otherwise has the value 'false'
writeUserType			boolean	- this boolean has the value 'true' if 'userType' is to be written; it otherwise has the value 'false'
allocationUnit			longinteger	- memory allocation unit

userType			integer	- personal identifier which the user gives to the file.
reply	0	T_reply	variant record	Diagnostics.

Diagnostics — The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation correctly executed.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDOPERATION	Invalid operation.
	INVALID	Invalid identifier.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not available.

Characteristics

1 It is not necessary to open the file before invoking WriteInfoByte.

2 The information is written immediately to disk (in synchronous mode).

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var   systemId : T_systemId;
      fileInfo : T_writeInfo;
      reply    : T_reply;

(* type

   T-writeInfo = packed record
                     writeAllocationUnit : boolean;
                          writeUserType : boolean;
                         allocationUnit : longinteger;
                               userType : integer
                 end;

   function WriteInfoByte(systemId : T_systemId;
                          fileInfo : T_writeInfo) : T_reply;
   external; *)

begin
 .
 fileInfo.writeAllocationUnit:=true;
 fileInfo.writeUserType:=true;
 fileInfo.allocationUnit:=512;
 fileInfo.userType:=7;
 reply:=writeInfoByte(systemId,fileInfo);
 .
 .
end.
```

WRITEINFOKEYED (WRITE INFORMATION ABOUT A KEYED FILE)

This function writes information about the file indicated by the system identifier.

reply
WriteInfoKeyed
systemId,fieldNumber,fileInfo

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
fieldNumber	I		integer	Integer indicating the sequence number of the index.
fileInfo	I	T_write-InfoK	packed record	Record which contains the following information:
writeInfo		T_writeInfo	packed record	
writeAllo-cationUnit			boolean	- this boolean has the value 'true' if 'allocation-Unit' is to be written; it otherwise has the value 'false'
writeUserType			boolean	- see 'writeAllo-cationUnit'
allocationUnit			longinteger	- memory allo-cationUnit

userType			integer	- personal identifier which the user gives to the file.
writeSplitStrategy			boolean	- this boolean has the value 'true' if splitStrategy is to be written; it otherwise has the value 'false'.
splitStrategy			enumerated	contains one of the following values: s50_,s90_,s100_ See CreateKeyed.
reply	0	T_reply	variant record	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation correctly executed.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDOPERATION	Invalid operation.
	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not available.

Characteristics

1 It is not necessary to open the file before invoking WriteInfoKeyed.

2 The information is written immediately to disk (in synchronous mode).

Note

See also:
- CreateKeyed

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'APS.d'

var  systemId    : T_systemId;
     fieldNumber : integer;
     fileInfo    : T_writeInfoA;
     reply       : T_reply;

(* type

   T_writeInfo = packed record
                    writeAllocationUnit : boolean;
                          writeUserType : boolean;
                         allocationUnit : longinteger;
                               userType : integer
                 end;

   T_splitStrategy = ( s50_, s90_, s100_ );

   T_writeInfoA = packed record
                                writeInfo : T_writeInfo;
                       writeSplitStrategy : boolean;
                            splitStrategy : T_splitStrategy
                  end;

   function WriteInfoKeyed(    systemId : T_systemId;
                           fieldNumber : integer;
                              fileInfo : T_writeInfoA) : T_reply;
   external; *)

begin
 .
 .
 fileInfo.writeInfo.writeAllocationUnit:=true;
 fileInfo.writeInfo.writeUserType:=true;
 fileInfo.writeInfo.allocationUnit:=512;
 fileInfo.writeInfo.userType:=2;
 fileInfo.writeSplitStrategy:=false;
 fileInfo.splitStrategy:=s50_;
 reply:=WriteInfoKeyed(systemId,0,fileInfo);
 .
 .
end.
```

This function writes information about the file indicated by the system identifier.

reply
WriteInfoPos
systemId,fileInfo

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
fileInfo	I	T_writeInfo	packed record	Record which contains the following information:
writeAllocationUnit			boolean	-this boolean has the value 'true' if 'allocationUnit' is to be written; it otherwise has the value 'false'.
writeUserType			boolean	-see 'writeAllocationUnit'.
allocationUnit			longinteger	-memory allocation unit.
userType			integer	-personal identifier which the user gives to the file.

reply	0	T_reply	variant record	Diagnostics.

Diagnostics — The primitive returns one of the following replies: (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDOPERATION	Invalid operation.
	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not ready.

Characteristics

1. It is not necessary to open the file before invoking WriteInfoPos.
2. The information is written immediately to disk (in synchronous mode).

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var  systemId : T_systemId;
     fileInfo : T_writeInfo;
     reply    : T_reply;

(* type

   T_writeInfo = packed record
                    writeAllocationUnit : boolean;
                          writeUserType : boolean;
                         allocationUnit : longinteger;
                               userType : integer
                 end;

   function WriteInfoPos( systemId : T_systemId;
                          fileInfo : T_writeInfo) : T_reply;
   external; *)

begin
 .
 .
 fileInfo.writeAllocationUnit:=true;
 fileInfo.writeUserType:=true;
 fileInfo.allocationUnit:=512;
 fileInfo.userType:=3;
 reply:=WriteInfoPos(systemId,fileInfo);
 .
 .
end.
```

This function writes a record into the file indicated by the system identifier.

```
reply
WriteKeyed
systemId,*primaryKey*,keyStartIndex,*data*,dataStartIndex,
writeMode
```

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
primaryKey	I		packed array [min1... max1: integer] of dataItem	Array containing the primary key associated with the record. 'dataItem' is defined by the user and its length must be of 8 bits, e.g. char, byte (-128..127).
keyStartIndex	I		integer	Integer within the range of 'data' area . It indicates the initial location of the key.

data	I		packed array (min2.. max2: integer] of dataItem	Array containing the data to be written. See 'primaryKey' for dataItem.
dataStartIndex	I		integer	Integer within the range of 'primaryKey'. It indicates the initial location from which to start transferring the data.
writeMode	I	T_writeMode	enumerated	It indicates the mode of writing: rewrite_: the contents of a record with an already existing primary key are updated. If the record does not exist an error is returned. newwrite_: a new record with a primary key is written. If the key already exists, an error is returned. This option could cause an address previously calculated with ComputeKeyed to change.

				alwayswrite_ : the record indicated is written regardless of the existence of the key. This option could cause an address which has been previously calculated with ComputeKeyed. to change.
reply	0	T_reply	variant record	Diagnostics.

Diagnostics

The primitive returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	OUTOFDISKSPACE	No more space on disk.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDOPERATION	Invalid operation.
	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSWARNING	DUPLICATEDKEY	New secondary key is a duplicate.
	INCONSISTENTDATA-BASE	Index inconsistent with contents of the file.
FSERROR	KEYNOTFOUND	The specified primary key for a record update does not exist.
	DEVICENOTREADY	Device not available.
	KEYALREADYEXISTS	Supposedly new primary key exists already.
	RECORDLOCKED	Record inaccessible because it is already locked by some other process.

Characteristics

1 The primitive transfers the data in the user area to the keyed file starting from the location indicated which must be in the range (min 1 .. max 1) of 'primaryKey'.

2 The size of the record and of the primary key are those set at the time of CreateKeyed.

3 Before writing the record, the primitive checks that no other process has 'locked' the record.

4 GetRecordView can be used to get the current view of the fields of the records in the file.

5 If the primary key is internal, then the appropriate field in data must be equal to 'primaryKey'.

6 When using this function to update a record in a file with secondary indexes, the file must already be open with read (r_) access as well as write (w_) access, so that the record can be read to update secondary indexes correctly as necessary.

Note

See also the primitive:

- GetRecordView.

Example

```
#include 'systypes.i'
type dataItem = char;          (* this type is user-defined *)
#include 'fstypes.i'
#include 'AMS.d'

var   systemId                          : T_systemId;
      primaryKey                        : packed array [ 1..16 ] of
                                                dataItem;
      keyStartIndex,dataStartIndex      : integer;
      data                              : packed array [ 1..100 ] of
                                                dataItem;
      writeMode                         : T_writeMode;
      reply                             : T_reply;

(* type

   dataItem = char;

   T_writeMode = ( rewrite_, newwrite_, alwayswrite_ );

   function WriteKeyed(          systemId : T_systemId;
                       var    primaryKey : packed array [ min1..max1 :
                                               integer ] of dataItem;
                          keyStartIndex : integer;
                       var          data : packed array [ min2..max2 :
                                               integer ] of dataItem;
                         dataStartIndex : integer;
                              writeMode : T_writeMode) : T_reply;
   external; *)


begin
 .
 .
 reply:=WriteKeyed(systemId,primaryKey,1,data,1,newwrite_);
 .
 .
end.
```

WRITEPOS (WRITE A RECORD INTO A POSITIONAL FILE)

This function writes a record in a particular position in the file specified by the system identifier.

reply
WritePos
systemId,position,*data*,startIndex,writeMode

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
position	I		long-integer	Numerical value which indicates the position of the record to be written.
data	I		packed array [min... max: integer] of dataltem	Array containing the data to be written. 'dataltem' is defined by the user and its length must be 8 bits, e.g. char, byte (-128 .. 127).

startIndex	I		integer	Integer in the same range as for 'data'. It indicates the location from which the transfer of data from 'data' area is to start.
writeMode	I	T_writeMode	enumerated	It defines the mode of writing: rewrite_: to update existing data. It is not possible to write outside the bounds of the file. newwrite_: to write new data. If the record is found to already exist then an error is returned. The file is extended accordingly. alwayswrite_: to write data without any restrictions.
reply	O	T_reply	variant record	Diagnostics.

Diagnostics The primitives returns one of the following replies (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation executed correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	OUTOFDISKSPACE	No more disk space.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDOPERATION	Invalid operation.
	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not available.
	RECORDALREADY-EXISTS	The record to be inserted already exists.
	RECORDLOCKED	Record inaccessible because it is already locked by some other process.
	RECORDNOTFOUND	No valid record exists at the specified position.

Characteristics

1. WritePos transfers the data from the user area to the position indicated. This may be in the file or outside it.

2. The size of the record is the same as that fixed at file create time.

3. Before writing the record, the primitive checks that there is no lock on it. If there is a lock, it returns an error.

Example

```
#include "systypes.i"
type dataItem = char;          (* this type is user-defined *)
#include "fstypes.i"
#include "APS.d

var   systemId    : T_systemId;
      position    : longinteger;
      data        : packed array [ 1..16 ] of dataItem;
      startIndex  : integer;
      writeMode   : T_writeMode;
      reply       : T_reply;

(* type

   dataItem = char;

   T_writeMode = ( rewrite_, newwrite_, alwayswrite_ );

   function WritePos(       systemId : T_systemId;
                            position : longinteger;
                     var         data : packed array [ min..max :
                                          integer ] of dataItem;
                         startIndex : integer;
                          writeMode : T_writeMode) : T_reply;
   external; *)

begin
 .
 .
 reply:=WritePos(systemId,?,data,1,alwayswrite_);
 .
 .
end.
```

WRITESEQ (WRITE A RECORD INTO A SEQUENTIAL FILE)

This function writes records in sequence into the file indicated by the system identifier.

```
reply
WriteSeq
systemId,*data*,startIndex,dataSize,writeMode
```

Parameter	I/O	System-Defined Type Name	Type	Description
systemId	I	T_systemId		File identifier.
data	I		packed array [min... max: integer] of dataItem	Array containing the data to be transferred to file. 'dataItem' is defined by the user and its length must be 8 bits, e.g. char, byte (-128..127).
startIndex	I		integer	Integer within the range of 'data'. It indicates the location from which the transfer of the data from 'data' area is to start.

dataSize	1		integer	Integer indicating how many bytes must be transferred to the file.
writeMode	1	T_writeMode	enumerated	This parameter must always contain one of the following values: rewrite_ : to update existing data. It is not possible to write outside the bounds of the file. newwrite_ : this indicates that the data to be written are always added on at the end of file. alwayswrite_ : to write data without any restriction.
reply	0	T_reply	variant record	Diagnostics.

Diagnostics The primitive returns one of the following replies, (see Error Types, earlier in this chapter).

Reply Class	Reply	Meaning
CORRECT	OKAY	Operation execution correctly.
SYSTEM	HARDWAREERROR	HW malfunction.
	SYSTEMERROR	System error.
	TABLEOVERFLOW	System table full.
	OUTOFDISKSPACE	No more space on disk.
	NOROUTING	No route available to access file.
	RESULTSDOUBTFUL	Timeout occurred during remote access.
	OPERATIONABORTED	Anomaly during remote access.
DEBUG	INVALIDOPERATION	Invalid operation.
	INVALIDID	Invalid identifier.
	INVALIDPARAMETERS	Invalid parameters.
	SECURITYVIOLATION	Operation incompatible with the protection status of the file.
FSERROR	DEVICENOTREADY	Device not available.

Characteristics

1 If 'data' contains a linefeed character, it will cause a new line to be started on sequential peripherals (terminal, printer).Thus data terminated by a linefeed is considered to be a record.

2 If no linefeed character is included in data; the system will cause a new line to be started only if the hardware limits of the particular device are reached.

3 If the file is a byte file on disk (opened in sequential mode by OpenSeq), the data is not interpreted. The linefeed character is considered to be as any other character.The data is written starting at the current position which is then increased by dataSize bytes. The current position can be set using the SeekP primitive and its value can be retrieved using the SenseP primitive. It gets changed by the ReadSeq primitive as well as by the WriteSeq primitive.

Note

See also:
- SeekP
- SenseP
- ReadSeq.

Example

```
#include "systypes.i"
type dataItem = char;          (* this type is user-defined *)
#include "fstypes.i"
#include "ANS.d"

var   systemId                : T_systemId;
      data                    : packed array [ 1..50 ] of dataItem;
      startIndex,dataSize     : integer;
      writeMode               : T_writeMode;
      reply                   : T_reply;

(* type

   dataItem = char;

   T_writeMode = ( rewrite_, newwrite_, alwayswrite_ );

   function WriteSeq(          systemId : T_systemId;
                       var          data : packed array [ min..max :
                                             integer ] of dataItem;
                              startIndex : integer;
                                dataSize : integer;
                               writeMode : T_writeMode) : T_reply;
   external; *)

begin
 .
 .
 reply:=WriteSeq(systemId,data,1,20,newwrite_);
 .
 .
end.
```

Code 4004700 N (0)

Printed in Italy