Documentation of development and maintenance of the manager of JEVEUX memory

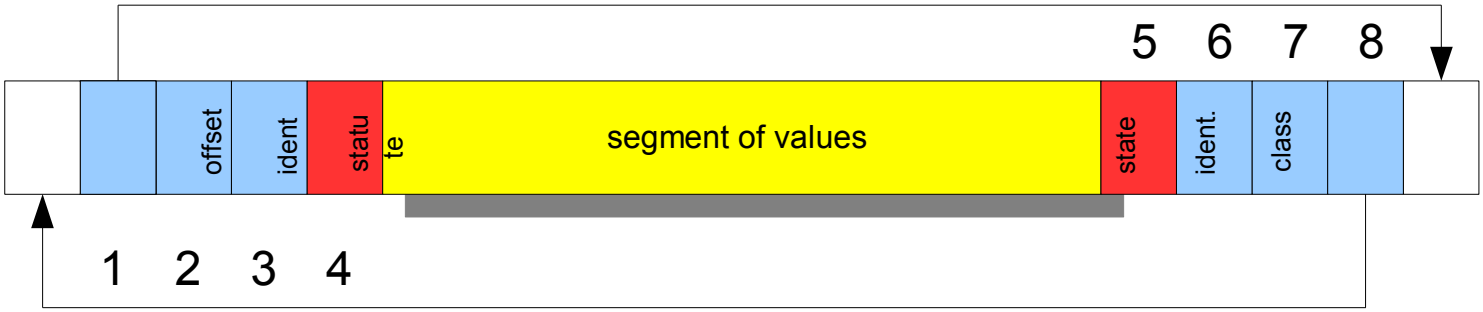
1 Introduction

Code_Aster was developed in FORTRAN 77, this language does not have dynamic management of the memory and does not allow a very strong structuring of the types. The manager of JEVEUX memory allowed to mitigate part of these disadvantages by giving the following opportunities:

dynamic allocation of the zones memories allocated in the course of works,
management of the memory overflows on file, with archivage of the results in the end of works,
structuring of the data of *the Code_Aster*, with access by name to the handled objects and standardization of types FORTRAN used.

This document is intended for documentation and with the maintenance of the routines of the manager of JEVEUX memory prefixed by JJ, I or JX, the routines JX call on nonportable functions in general. Thereafter, we will indicate this set of routines under the term “the software”. A precise description of the operation and the internal organization of the software is detailed there. The reader will be able to which describes refer to documentation [D6.02.01] Management JEVEUX memory the interface of the routines “user” I…. .

2 Organization of the memory



JEVEUX dynamically allocates each memory zone intended to accommodate the values associated with the object or the collection with objects. A control of cumulated space is carried out during each new assignment, and a mechanism of unloading can possibly be implemented to release from spaces of the memory associated with the object that the developer with declared like being used more.

On the platforms 64 bits, the assignment of the memory zone is carried out dynamically using the routine system HPALLOC. This zone is seen in the software through the table of the whole type (INTEGER*8 ISZON) length LISZON and an address of beginning JISZON. It is stored in the /IZONJE/ commun run. One will use in the continuation of the document the term “key” to indicate the unit of addressing. On platform 64 bits the key has as a length 8 bytes and corresponds to length of type INTEGER*8.

The zone is managed key by key in unit of the type INTEGER (unit of addressing), the segments of values associated with the JEVEUX objects are framed by 8 keys containing, in this order, following information:

1) the address according to the last word constituting the segment of values and the 8 identifiers;
2) the value of a shift used to align the segments of the type length higher than the unit of addressing. On platform 64 bits, this value is always null for the segments of values associated with objects of the type INTEGER and REAL*8. It is worth 8 (bytes) sometimes for the segments of the values associated with objects of the type COMPLEX*16 : indeed it happens that the beginning of the segment of value (position 5) cannot coincide with a position in the table of reference ZK16, it is then necessary to move of a key (8 bytes). It is even more frequent with type CHARACTER when it is worth 16,32 or 80!
3) the whole identifier associated with the simple objects or the collections;
4) the statute of the segment of values which can take the value X or U (coded on an integer);
5) the state of the segment of values which can take the value X, A or D (coded on an integer); ;
6) the whole identifier associated with the objects of collection;
7) the code of the class associated with the JEVEUX object ;
8) the address of the first key preceding the segment by values and the 8 identifiers.

The management of the memory zone with type INTEGER does not make it possible to be aligned correctly with the types of length higher than this unit of addressing. Although the order EQUIVALENCE present in the software makes it possible to align the initial address of the various variables (tables) of reference ZI, ZI4, ZR, ZC, ZL, ZK8, ZK16, ZK24, ZK32 and ZK80, the positioning of a segment of values associated with an object of the type ZK32 have little chance to be aligned with a “multiple” of 4 of table ISZON on platform 64 bits, from where need for managing a shift among the descriptors.

The values making it possible to code the statute and the state of the segment of values are obtained so that the representation does not correspond to any type used within the segment of values and this in order to detect possible crushings of these descriptors. This role is reserved for the routine JJLIRS which is called mainly during the requests of setting in memory of the segment of values. The emission of an error message of the type “POSSIBLE CRUSHING UPSTREAM …” indicate that the

identifier (3) or the statute (4) were crushed, the emission of an error message of the type "POSSIBLE CRUSHING DOWNSTREAM ..." indicate that the state (5) or classifies it (7) were crushed.

On platform 64 bits, the values used have the following octal representation:

01000000000000000000	for X,	03000000000000000000	for A,
02000000000000000000	for U,	04000000000000000000	for D.

Usage of the segments of values

the couple state/statute makes it possible to know the use of a segment of values in memory:

XX indicates that the segment of values is free, this position is directly usable,
UA indicates that the segment of values is used in reading (its disk image will not be brought up to date after release),
UD indicates that the segment of values is used in writing (its disk image will be brought up to date after release),
XD indicates that the segment of values was released, but that its contents will have to be discharged on disc,
XA indicates that the segment of values was released, this position is directly usable.

3 Management of the memory

a request of access in reading or writing on the segment of values associated with a JEVEUX object causes, if it does not appear there already, a loading in memory of the contents of the associated segment of values. The memory address of a JEVEUX object corresponds to its relative position in table ISZON. As a preliminary, it is necessary to carry out a dynamic allocation, using routine JJALLS to insert the segment of values. When the request of assignment fails, the system refusing to allocate a memory zone, a mechanism of release is started, it can involve accesses disc when zones associated with segments with values must be written on the file associated with the base. The new segment of values is allocated with a tolerance of 8 integers which correspond to minimum space associated with a segment with values (1 integer by descriptor). When the search for core memory fails, one causes a stop of the application in <S> error (stop by the supervisor with safeguard of the concepts created).

A call to the function system LOC through routine JXLOCS makes it possible by means of to obtain the relative address of the beginning of the segment of values compared to table ISZON the value of the position of reference of the beginning of the memory zone obtained in JXALLM and stored in the /ILOCJE/ commun run. It is the use of the routine JJALTY which makes it possible to switch on the table Z. and to obtain according to the type the address compared to the good reference.

The assignment of a segment of values associated with an object of the type of which the length is higher than the unit of addressing used (for example for type CHARACTER *24) automatically does not make it possible to be aligned compared to table ISZON, it is sometimes necessary to shift few keys. The value of this shift is stored in the second descriptor preceding the segment by values and the effective size of the segment of values is adjusted by taking account of its associated type.

It then remains to bring up to date the descriptors associated at the segment with values, this operation is carried out by routine JJECRS.

Search for core available

the call to routine JEDISP makes it possible to know at the time of the call, the size of the memory zones available, it carries out the search by traversing the group of the segmentation memory and progressively deposits the size of the free zones or déchargeables in a table provided by argument of call.

Checking of the memory

crushings report which affect the descriptors (state or statute) or the sequence front can be detected by means of routine JXVERI. This routine one by one examines the descriptors of the segments of values in memory. An error message fatal is transmitted during the detection of an anomaly, if not the routine remains dumb.

4 Management of the releases

the release is the most complex mechanism implemented in JEVEUX. It is conceived easily that when that one manages a finished memory capacity, it arrives one moment when it is not possible any more to find of core. It is then necessary to cause unloadings on disc or to recover core of the zones become useless. JEVEUX deals with these mechanisms, with condition of course, that the programmer indicated the objects concerned to him; it is necessary to take some care before releasing an object, several units of program which can use a memory address simultaneously. The strategy of release calls on the one hand on an internal mechanism with the manager of memory which we will describe and on the other hand with rules of programming which are the object of the document [D2.06.99] “New strategy of release of the JEVEUX objects ”.

The release of a segment of values materializes by positioning with the value X of the state instead of the value U. It does not have there another immediate effect, it is only during a later search for storage position that one will treat indeed the contents of the segments of values.

The setting in memory of a JEVEUX object is accompanied by the assignment of an attribute system: the mark. This attribute, of whole type, takes the value of a meter incremented with each call to routine JEMARQ and décrémenté with each call to routine JEDEMA. It is possible to obtain the value of the current mark by calling routine JEVEMA.

The current marks have a strictly positive value thus. Values -1,-2 and -3 are used to treat the following exceptions.

Value -1 is used to permanently keep (throughout the execution of an Aster command) certain objects which will be released by a specific call. This mark is used at the time of the call to routine JEVEUT.

Value -2 is used by JEVEUX to bring back in a temporary way certain objects which will be released at once the finished action (put in memory of the system objects of collection,…).

Value -3 is used to permanently keep (throughout the execution of *Code_Aster*) the objects used by the Supervisor.

Mark -3 can come to replace any existing mark, mark -1 can replace a mark (positive) existing. The system object containing the list of the addresses of the segments of values must then be modified. Mark -3 is used at the time of the call to routine JEVEUS.

One thus builds a hierarchy of the segments of values associated with the objects. Each call to routine JEDEMA will cause the release of the segments of values having the current mark. In order to optimize the releases pulled by a call to JEDEMA, the setting in memory of each segment of values is accompanied by the storage of its position (its memory address) in an object system (segment of values of the whole type). Thus all the segments of values associated with an identical mark are easily identifiable and their location requires only one simple sweeping of a vector of integers. The loop on the segments of values is carried out in two times: one first of all treats overall the collections, then the objects simple and the segments of values associated with the contiguous collections are released.

The actualization of the mark of an object is carried out at the time of the call to routine JJECRS, the object system KDESMA is redimensionné (this object is indicated here by the name of variable FORTRAN used to store its address within the units of program).

It is the routine JJLIDE which carries out indeed the release of the segments of values. The first argument of this routine is the name of appealing, it conditions the type of operation to be carried out:

LIBE standard mechanism of release with examination of the state, the statute and the mark,
HEAP mechanism of release with immediate writing used during the retassage of the files or in mode debug JEVEUX,
LIBF mechanism used at the end of the work during the closing of a base JEVEUX.

Concerning the simple objects, this release does not pose a particular problem: routine JJLIDE checks that the mark associated at the segment with values is identical to the current mark stored in the /IADMJE/ commun run, it modifies the descriptors (state and statute) of the segment of values and assigns to 0 the associated mark, possibly it causes an unloading (JXECRO) and modifies the contents of the attributes memory address and addresses disc. The release of an object of dispersed collection follows the same process, the attributes being modified within the system objects of collection. The release of a collection is more delicate, the system objects having to be maintained accessible in memory as long as a segment of values associated with the one with the objects with collection is present in memory (used, déchargeable, and even removable).

5 Management of the files of direct access

the manager of JEVEUX memory manages report unloadings on disc, to release from core in memory during the execution and to file the results at the end of the work. One uses for this purpose of the files of direct access. They are utilities OPENDR, WRITDR, READDR and CLOSDR which are called (bibc/utilitai/iodr.c).

The address disc of the JEVEUX objects is obtained by combination of the number of the record of the file of direct access used to deposit the values, and possibly the position within this record.

The length of the records is fixed, its value is selected at the time of the opening of bases JEVEUX via routine JEINIF .

The number of records which is part of the parameters, is given in Code_Aster *according to* the conditions of operating. Each base is cut out in logical unit length 12 884 901 888octets (notion of “extend”), this value is affected through function ENVIMA LOFIEM and stored in the /FENVJE/ commun run . JEVEUX manages a total index which it then cuts out for each extend, the address disc is measured compared to the total index, then modulo the number of records, one easily obtains the number of the extend and the relative address. The various logical units are accessible by a local name which is composed starting from the first four characters in small letters of the name of the base associated and the number with extend.

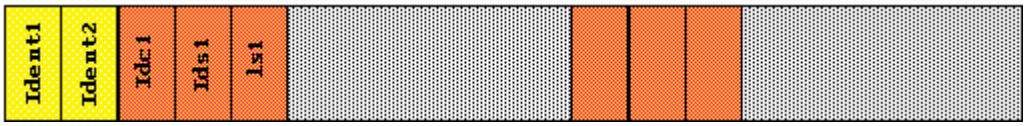
The size of the records defines two classes of JEVEUX objects :

1) the small objects whose size is lower than the length of a record, they are accumulated in a space of the size of a record before transfer on disc,
2) the large objects which require several records to store their contents. During

a request of writing on disc, the contents of the large objects will be directly transferred on disc, whereas a plug of writing is used for the small objects in order to cumulate them and to reach approximately the length of a record before their transfer. During a request of reading, at least it is a record which will be used, in the case of the small objects one uses a plug of reading. The plugs of reading and writing are part of the system objects associated with each base JEVEUX .

The closing of the files of direct access is essential to bring up to date the index of access, it is the routine JXFERM which call utility CLOSDR . Description

of the records Each



record auto--is described in order to easily be able to identify its contents. As for the memory zone, the records are seen like a succession of keys of the whole type (INTEGER*8). The first two keys give total information about the size of the stored objects: if

Ident 1=Ident2=0 the record contains small objects, three whole keys (descriptors) are placed in front of each segment of values, it contain respectively, when they exist, the identifier of collection (Idc1), the simple identifier of object or the number of object of collection (Ids1) and the length of the segment of value, follow the segment of values, and one starts again for the following until Idcn =Idsn=0; if

Ident 1 or Ident 2 is different from 0, the record contains whole or part of the segment of values associated with a large object. At the time

of the destruction of a large object the identifiers Ident 1 and Ident 2 are positioned with the value opposite (assignment of the sign -). In the same way, at the time of the destruction of a small object, the Idci identifiers and Idsi are affected sign -. Writing

of the objects It

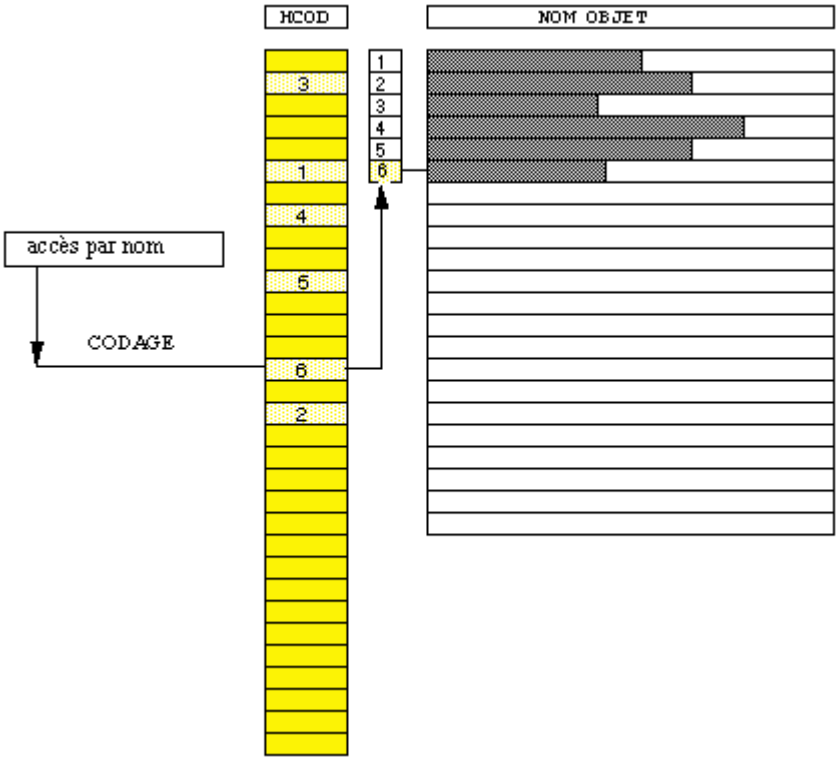
is the routine JXECRO which treats the writing of the JEVEUX objects . It also ensures, when it is necessary, the opening of the logical units associated with the partition in extend of the bases. The routine examines the various records to find a succession of records being able to accommodate the segment of values or the plug following the cases. The records corresponding to large destroyed objects can thus be recovered. The writing of a small object results in a displacement of the contents of the segment of values in the plug of writing by routine JXDEPS with actualization of the descriptors. The plug is transferred on disc only if the segment of values is of a size higher than remaining free space. The segment of values associated with large objects is transferred on disc by routine JXECRB . JXECRB is a hat which calls on the utility WRITDR which brings up to date the descriptors Ident 1 and Ident 2 as well as a meter associated with the record. During later unloadings, the plug of reading can be used to bring up to date the disk image; a logic indicates this kind of use then. Reading

of the objects It

is the routine JXLIRO which treats the reading of the JEVEUX objects . The segments values associated with the small objects is reloaded in memory from one of the plugs of reading or writing. The plug of reading can possibly be discharged on disc before charging a new record. The segment of values associated with the large objects is directly read again using routine JXLIRB . Access

6 by names: hash-coding

the simple JEVEUX objects, objects and objects of collections, are accessible by name. Names handled by the routines I …. comprise 24 characters, the names used in-house by JEVEUX comprise 32 characters to treat the case of the collections. The access by name, if it facilitates legibility and makes it possible to structure the data, cannot be used directly in-house. One thus resorts to an algorithm of hash-coding which, using a function of coding, makes it possible to associate a whole identifier with a name. This system of coding is used to manage the names of the JEVEUX objects for each definite class (partner at each base), but also for the names of the objects of collection. One



uses for this purpose either a couple of objects formed by a vector of integers and a vector of character strings for the management of different the classes, or an object of name directory kind having the characteristic to be of heterogeneous contents (storage of the character strings and the whole identifiers) for the named collections. The use of these directories requires particular functions of access. The design of these objects is carried out in order to and the contain the number of necessary identifiers by minimizing their size collisions on the level of the function of coding. Our choice to dimension the pointer of integers associated with result with the function with addressing stopped on the following condition: where

$nrep = nprem$ **is** $nprem > 1.3 \times nmax$

$nprem$ a prime number and $nmax$ the maximum number of identifiers to be stored. The computation

of the size of the directories is realized by the function JJPREM `in` which is stored in the form of WENT BACK a list to 56 prime numbers up to value 611957 what limits to this value the capacity of the name directories (and that of the pointer of integers to approximately 795.000).

The selected function of coding JXHCOD `calls` on the function system STRMOV `which` makes it possible to transfer byte by byte a character string in a table from integers and to function XOR to cumulate the results in an integer (INTEGER). The identifier is finally obtained by a congruence modulo the length of the directory. $nrep$ Insertion

of a name

the insertion of a new name in the directories associated with the various bases is carried out using routine JJCREN . All the directories of the open bases are examined in order to ensure the unicity of the name of object, routines JEVEUX not accepting as argument the class associated with the base. If

the name directory starts to be saturated, the function of coding, although selected to be dispersive, can give an identical identifier for two distinct character strings, there is then collision. A new identifier must be given by taking account of the value obtained previously. Scan

for noun in a directory This

algorithm requires a certain number of comparisons of character strings and can thus become expensive. A commun run, brought up to date by the last search in the directories of the various bases contains the identifier and the associated character string in order to reduce the cost of search (/IATCJE/). It is the routine JJVERN which carries out a comparison with the contents of the commun run before the call to JJCREN . The return code of routine JJCREN depends on the type of search in the directory: with insertion in name directory (ICRE=1) this return code is obligatorily non-zero (there is possibly stop in error), without insertion it can be worth 0, if not 1 corresponds to a simple object and 2 with a collection. It is the presence of a nonwhite character string between the positions 25 to 32 which indicates that one treats an object of collection.

The characters composing the names of the objects are restricted with the alphanumerics supplemented by the special characters: “

“	the blank,”.	”	the point, “_
”	the underscore, “$	”	the symbol dollar, “&
” it	and commercial.		

The conformity of the character strings is checked after insertion in the directories (during the creation of the name only) by comparison character by character with the contents of the /JCHAJE/ commun run initialized in routine JEDEBU . Does destruction

of a name

the destruction of a name use the same algorithm as insertion, the position in the directory cannot be released because of possible collisions, one thus proceeds while making negative the identifier and while affecting to “?” the character string of the name to be destroyed. Thus it will be always possible to recover this position later on. Redimensioning

of the directories

the redimensioning of the directories is ensured in an automatic way using routine JJAREP . The size of the directories of the bases is doubled at the time of the operation. This operation entirely rebuilds the new existing directory by insertion of the names. The order of insertion being preserved, the system objects do not require particular processing, other that a recopy in a larger receptacle (it follows from there a displacement in memory of the latter) and their actualization on disc. Cases

of the directories of collection

the directories of collection are objects of nonhomogeneous contents: they at the same time store result (of whole type) of the function of coding and the character strings composing the names. They auto--are described and the routines using them contain the following instructions: INTEGER

```
ILOREP           , IDENO, ILNOM, ILMAX, ILUTI, IDEHC PARAMETER
(      ILOREP=1, IDENO=2, ILNOM=3, ILMAX=4, ILUTI=5, IDEHC=6)
```

the value positioned with the address: Represent	**: ILOREP**
	size necessary to all the whole codes, IDENO
	the address from which the names are stored, ILNOM
	the length of the stored character strings, ILMAX
	the maximum dimension of the directory, ILUTI
	the number of actually stored names, IDEHC
	the address from which the whole codes are stored.

The information stored with address ILUTI is brought up to date and used in the internal functions of access to the directories, its value is only accessible in an external way, using utility JELIRA with for name of attribute NOMUTI .

The objects of named collection, during their creation by routine JECROC , are inserted using function JJCODN . Intermediate routine JJCROC allows , according to the value of its second parameter, to insert a new name or to check its existence and to recover its order of insertion.

7 The system objects and the segments of values not referred

the manager of JEVEUX memory uses part of the memory to manage the attributes associated with the objects and for treating certain functions. In order not to multiply the routines of access we chose to use same structures for the JEVEUX objects and the memory used for their management. This is why during the printing of the segmentation memory one sees appearing segments of values associated with illicit names with the meaning user and referring to the various classes open to a given time, but also of the segments of values which are associated with no name. The system objects associated with the global database carry all the following prefix: _____GLOBALE_____ (the name of the base is in position 9 to 24), the suffix (in position 25 to 32) makes it possible to distinguish the various objects. The names of the system objects are built in the same way for the other bases.

The system objects are created during the first call to routine JEINIF . JEVEUX needing permanently to reach the associated segments of values, a specific mark (- 2) their is affected. A particular processing their is reserved during the closing of the bases. List

system objects used by JEVEUX : Suffix

	of the name of the system object Contained	**Standard**	**associated FORTRAN (64 bits) Cuts**	**1**
$$	CARA characteristics	of base associated INTEGER	*8 11	2
$$	IADD addresses	disc with objects INTEGER	*8 2*	NREMAX 3
$$	GENR kind	of objects (E, V, N or X) CHARACTER	*1 NREMAX	4
$$	TYPE standard	of objects (I, R, C, L, K) CHARACTER	*1 NREMAX	5
$$	DOCU documentary	field CHARACTER	*4 NREMAX	6
$$	ORIG documentary	field CHARACTER	*8 NREMAX	7
$$	RNOM lists	names of objects CHARACTER	*32 NREMAX	8

standard	$$ LTYP	of the segments of values INTEGER	LONG *8	NREMAX
9	$$ length	measured in the type segments of values INTEGER	*8 NREMAX	10
$$	LONO effective length	measured in the type of the segments of values INTEGER	*8 NREMAX	11
$$	DATE go back	to first saves INTEGER	*8 NREMAX	12
$$	LUTI length	used of the segments of values INTEGER	*8 NREMAX	13
$$	HCOD counts	of hash-coding INTEGER	*8 NRHCOD	14
$$	USADI description	of the contents of records INTEGER	*8 2*	NBLMAX 15
$$	ACCE number	of access in read/write to records INTEGER	*8 NBLMAX	16
$$	MARQ marks	associated with objects INTEGER	*8 2*	NREMAX 17
$$	INDX index	with the file with direct access associated INTEGER	*8 2*	NBLMAX 18
$$	TLEC plug	with reading INTEGER	*8 LONGBL	19
$$	TECR plug	with writing INTEGER	*8 LONGBL	20
$$	IADM memory addresses	of the objects INTEGER	*8 NREMAX	where

NREMAX

is the maximum number of names associated with a class, NRHCOD is obtained from NREMAX with function JJPREM , NBLMAX
is the maximum number of records, LONGBL
is the length of the records.

The dimension of the majority of the system objects is likely to be readjusted in the course of computation according to the needs, only what milked in keeping with files of direct access and with the length of the records remains fixed. The last 5 objects of the list above do not have a disk image. Segments

of values not referred present in memory Two

segments of values present in memory do not have names to identify them, we indicate them starting from the name of the variables which are used in the subroutines. Standard

	contents	FORTRAN associated Cut	KPOSMA
ISZON	(JISZON+KPOSMA+I) is the position in the segment of values associated to KDESMA of the addresses associated with ième mark INTEGER	*8 LGD	KDESMA
memory addresses	with “marked” objects INTEGER	*8 LGP	

dimensions LGD and LGP are adjusted during the execution, their initial values are respectively the sum lengths of the vectors $$ RNOM of each class and value 50.

8 The collections

the collections of JEVEUX objects are structures which allow the pooling of the attributes and possibly an access named to a group of objects. They can be associated with a single segment with values (contiguous collection) or with as many segments with values with objects (dispersed collection). They are built starting from objects simple JEVEUX , and thus appear in this form among the objects associated with a class. The main object of the collection is the object of kind X, it is a vector of 11 integers containing the identifiers of the various objects composing the collection (inter alia the system objects of the collection which contain a suffix starting with $$). This vector bears the name allotted using routine JECREC (CHARACTER *24). The system objects specific to the collection carry a suffix starting with $$ in position 25, if they are associated with a divided object, they carry a suffix starting with &&. The attributes common to all the objects of collection are deposited among the attributes of the object system $$ DESO (kind , standard, length, etc).

The system objects associated with a collection are created in the associated class (identical attribute for all the system objects) and are charged in memory via routine JJCREC . Suffix

	of the name of the system object of collection Contained	Standard	FORTRAN associated Standard	with collection 1
$$	DESO the attributes associated with this object are the common attributes of the collection contiguous	collection: values associated with the various objects with collection INTEGER	*8 dispersed	and contiguous the segment with values exists only for the contiguous collections 2
$$	IADD addresses	disc of the objects of dispersed collections INTEGER	*8 dispersed	3
$$	IADM memory addresses	of the objects of dispersed collections INTEGER	*8 dispersed	4
$$	MARQ marks	associated with the objects with dispersed collections INTEGER	*8 dispersed	5
$$	NOM or object divided list	with the names with objects with collections named according to	the directory associated named	6
$$	LENGTH or object shared length	measured in the type with the segments of values; receives the values associated with attribus LONMAX and variable NOMMAX	INTEGER *8	length 7
$$	LONO or && LONO effective length	measured in the type with the segments with values; is used in-house by software INTEGER	*8	variable length 8
$$	LUTI or && LUTI length	used of the segments of values INTEGER	*8	length variable 9
$$	NUM information	concerning numbered collections INTEGER	*8 numbered	

the routines using the collections, and more precisely the descriptor object of the collection contain the following instructions: INTEGER

```
 IVNMAX        ,     IDDESO, IDIADD     ,     IDIADM,     +
IDMARQ              ,     IDNOM,                IDLONG,     +
IDLONO              ,     IDLUTI, IDNUM     PARAMETER
 (    IVNMAX = 0, IDDESO = 1, IDIADD = 2, IDIADM = 3, +
IDMARQ              = 4, IDNOM =  5,             IDLONG = 7, +
IDLONO              = 8, IDLUTI = 9, IDNUM =  10) What
```

makes it possible to position directly in the memory zone to obtain the identifiers of the system objects (when they exist) in the following order: $$ DESO, $$ IADD, $$ IADM, $$ MARQ, $$ NOM, $$ LENGTH, $$ LONO, $$ LUTI and $$ NUM. The maximum number of objects of collection is stored with address IVNMAX .

The named collections

the objects associated with such a collection are accessible by their name (function JEXNOM) and by their sequence number from insertion (function JEXNUM). It is possible to use routines JENUNO

and JENONU to pass from the number to the name and conversely. The length of the names of the objects is restricted with 8 characters (CHARACTER*8) if the collection leans on a directory of "internal" name, or can be worth 8,16 or 24 if the collection leans on a directory of "external" name, i.e. created beforehand (shared name directory).

The numbered collections

the objects associated with such a collection are only accessible by their sequence number from insertion (function JEXNUM). The object system $$ NUM is a vector of 2 integers respectively containing the maximum number of objects of collection and the number of objects used.

The dispersed collections Each

object is associated with a segment of values, it is thus not necessary to bring back the group of the collection to reach a particular object. In this case it is necessary to manage 3 system objects: one for the memory addresses of the segments of values ($$IADM), the other for the addresses disc ($ $IADD) and the last to manage the releases ($$MARQ).

The contiguous collections There

exists only one segment of values for all the objects of the collection which is created and dimensioned once and for all at the time of the first setting in memory of one of the objects of collection. This segment of value is associated with the object system $$ DESO.

The collections variable length Each

object must be dimensioned: by affecting the attribute length by routine JEECRA or while providing of a vector length (divided object). In this case 3 system objects are necessary: for the lengths ($$LONG or divided object), for the lengths in the type of the associated segments of values ($$LONO or && LONO in the case of a divided object) and finally for the lengths used ($$LUTI or && LUTI in the case of a divided object). In the case of

the contiguous collections, it is possible by means of to reach directly the vector cumulated lengths (LONCUM) function JEXATR combined with the call to JEVEUO to obtain the address of this vector. This access makes it possible to be freed from a call to JEVEUO by object of collection. The dimension of the object system $$ LONO is incremented of 1 compared to the length of the object system $$ LENGTH for this purpose . This
mechanism was extended to the collections dispersed to reach attributes LONMAX and LONUTI for each object of collection, the access to the attribute for each object of the collection using which can function JELIRA appearing expensive.

The collections length fixes Each

object has same dimension, this attribute can be affected various ways: by directly affecting the attribute length of an object of the collection or attribute LONT overall length for a contiguous collection (call to JEECRA).

The mechanism of access to the objects of collection

the requests of access to the objects of collection request the system objects attached to the collection, of the same way that it is necessary to have access to the system objects associated with a class during the requests on the simple objects. It is thus necessary that these objects are present in memory as soon as a request is carried out on one of the objects of collection. Routine JJALLC is charged to put in memory the system objects of collection. They obey special rules concerning the releases because they can be discharged from the memory only when all the objects of collection themselves were discharged (actualization of the addresses disc and memory). The management of

the divided objects is even more delicate because it is necessary to be able to secure itself against an inopportune release of the latter, for this purpose, they receive a particular mark which is worth -1.

The various requests on the objects of collection are carried out starting from the routines I used for the simple objects, but require the use of the functions of synchronization JEXNOM , JEXNUM or JEXATR . These functions of the type CHARACTER *32 update the contents of the respective commun runs /IDATJE/ , /INUMJE/ and /KNOMJE/ , moreover they replace the character string associated in the name of the JEVEUX object in position 25 to 32 by the respective suffixes $$ XNOM, $$ XNUM and $$ XATR. The routines of low level then will search this information within the various commun runs according to the type of access.

9 The poursuites

the system object of suffix $$ CARA (containing the name of the base associated in position 9 to 24), contains information necessary to the reopening of the file of direct access, it contains inter alia, the position of the segment of values associated with the addresses disc with all with the objects contained in the base, as well as dimensions characteristic of the system objects. One thus takes the precaution at the head to store it in the first record. In the event of poursuite on a basis, the first action carried out will be the relecture of the contents of this object. Routine JXLIR 1 opens the file associated with the first “extend” (glob.1) with characteristics which are clean for him (what can lead to an alarm message), reads the first 14 values (3 for the descriptors of the segment of values on disc and the 11 expected values) then closes the file. The length of the index being known, it is then possible to reopen the files properly (the system objects can be deposited on the various files constituting the base). The system objects not having a disk image are created and initialized (memory address, plug of input/output,…).

10 The processing of the Creation JEVEUX objects

of the objects

the creation of the descriptors (name and attribute of class, kind and type) of the JEVEUX objects is carried out using routine JECREO for the simple objects, and by routine JECREC for the collections. The decoding of the character string passed in argument to affect the attributes is carried out by routine JJANAL . In the case of the objects of kind E the attributes length are directly affected.

The assignment of the attributes

the generic attributes are affected during the creation of the simple name of object or of collection, they appear in the following table: for

the objects simple and the collections: CLAS

classifies	fastening of the object to a data base. V	: base Volatile, G : global database, L : base Local, C : E base Compiled catalogs
GENR	kind of the object	: simple variable, V : vector, N : name directory TYPE
type	FORTRAN of the object I,	R, C, L, K8, K16, K24, K32, K80 LTYP
length	of the type managed	automatically for the types I, R, C, L, standardized to 8,16,24,32 and 80 for the characters for

the collections only: Access

Type	of access: NO if named, NU if numbered NO	can be followed by the name of name directory STOCKAGE

CONTIG	or DISPERSES MODELONG	
mode	of definition length of the objects of collection: CONSTANT or VARIABLE VARIABLE	can be followed by the name of the pointer length LONT
overall length	of a contiguous collection NMAXOC	
maximum	number of objects of the collection	

the other attributes are affected using routine JEECRA , these attributes appear in table Ci - below: for

the simple objects or the objects of collection: LONMAX

length	of the object of kind V NOMMAX
length	of the object of kind N LONUTI
length	used of the object of kind V NOMUTI
length	used of the object of documentary kind N
DOCU	field (4 characters)

the reading of the attributes

the values associated with the various attributes can be consulted at any time using routine JELIRA , including in-house managed attributes: accessible

internal attributes: GO BACK

date	to last disc unloading of object ORIG
not	used IADM
memory	IADD
addres s	
addres ses	disc LONO
length use	measured in the type of the segments of values and according to their kind USE (statute and state) of the segment of values in memory: UD , UA , XD , XA or XX .

The statute and the state of a segment of values in memory can be collected by this routine by means of USE for value of the argument of the name of attribute. Value XOUS for this same argument makes it possible to determine if the object is a collection (X) or a simple object (S). Note:

The consultation of the attributes of the objects of collection can require the setting in memory of the attribute objects, and their release at the end of the action. A temporary mark equal to -2 is affected in this case. Request

of access to the objects

all the requests of access to the JEVEUX objects (simple objects, objects of collection or whole collections), which they are direct (JEVEUO, JEVEUS , JEVEUT) or indirect (JEEXIN, JENONU , …) follows the following process: processing

of the name of object passed in argument by JJVERN , possibly
, put in memory or checking of the presence in memory of the system objects in the case of a collection by routine JJALLC , then using routine JJCROC , determination of the identifier of object of named collection or checking of the sequence number of collection numbered, according to
the type, call to JJALTY to obtain the address compared to Z* table of the commun run of reference, possibly
put in memory then assignment of the identifiers of the segment of values and the mark, and determination of the relative address by routine JXVEUO , in
certain cases (for example consultations of attributes), release of the object and/or collection by JJLIDE .

The attributes necessary to the description of the JEVEUX object are read again or determined by the routine JXVEUO , the processing is immediate for the simple objects because one has access directly to the attributes in the system objects associated with the base, some operations are necessary to treat the attributes of collection or object of collection (positioning in the system objects of collection).

Alternative JEVEUS allocates in a permanent way the segment of values in memory. Destruction

of the objects

the destruction of a JEVEUX object (simple object, collection or object of dispersed collection) requires two interventions: destruction of the segment of values and destruction of the attributes. For a simple object, the segment of values can have a disk image, in this case it is also necessary to destroy the latter, the corresponding records will be marked free and could be recovered later. The segment of values in memory will be marked free. The disk image, if it exists is marked free by means of, according to the type of object, the descriptors of the record (object system $$ USADI) or the descriptors within a record (assignment of the sign -). This function is provided by routine JXLIBD . Attributes (name, length, kind, etc) will be released (routine JJMZAT) and their position in the system objects of the associated base will be available for new the creation of descriptor. The system object containing the address of the marked objects must also be reactualized. The processing of an object of collection is identical, the actualization of the attributes is carried out on the system objects of the collection. The segment of values for an object of contiguous collection cannot of course be destroyed. The destruction of a collection is carried out by destroying all the objects of collection and the system objects of the collection provided that they are not divided. Routines JEDETR and JEDETC make it possible to destroy JEVEUX objects, the first work from an identifier, the second, more expensive, first of all carry out a search for descriptor in the directories of the classes open from a character string to a given position. Routine JEDETV is only used V to destroy the objects on the volatile basis associated with the class between the various commands of the Code_Aster . *Explicit*

release of the objects Although

the mechanism of release is implemented with the notion of mark and the compulsory calls to routines JEMARQ and JEDEMA , certain configurations require an explicit call to the following routines of release: JELIBE

releases the object required by respecting the affected mark, JELIBS
releases the object of name passed in argument when the associated mark is worth -3, JELIBZ
releases all the objects associated with a class with which the mark associated is worth -1. Recopy

objects

utility JEDUPO allows a JEVEUX object to duplicate (simple object, or collection supplements) possibly while depositing result on a different class. The new objects are released at the end of the operation. If this action does not raise any difficulty for the simple objects, some care are to be taken concerning the collections leaning on external pointers. The latter can be recreated to become system objects specific to the collection (one does not profit any more a pooling of the attributes concerned) or to be preserved such as they are, but it is then not allowed to deposit result recopy on another class. The receptacle can preexist (the user provides a name or a character string), in this case it is destroyed at the beginning of operation. The recopy does not require obligatorily the presence in memory of the segments of values to be copied, they can be read again directly on disc. It

is possible to use the utility JEDUPC which works from a under-character string of characters but requires on the other hand a preliminary search of the names in the directory (what can prove to be expensive). Printing

of the contents of the segments of values

utility JEIMPO is charged to print in a pleasant way the contents of (of) the segment (S) of value (S) associated (S) with the JEVEUX objects . The objects system (associates with a class or a collection) are treated by routine JEPRAT . A setting in memory being able to be carried out, a particular mark (- 2) is assigned to the segments of values charged. According to the type of object (simple object, object of collection or collection) one recovers the attributes associated with (X) the segment (S) with values to call the routine JJIMPO which carries out the formatting of the data.

11 The processing of the bases Certain

operations treat in their entirety bases JEVEUX , they are essential to initiate the system of management of memory, but are also used at the end of the process. Attention, the contents of the bases is systematically enriched during the execution of the command Aster POURSUITE , and it is essential to finish execution FIN by the command to close the files of direct access properly. Only a stop with a message UTMESS of the <S type > makes it possible to the SUPERVISOR to validate the concepts created and to properly close the files of direct accesses by call to routine JEFINI

the opening of a base

the length of the records of the file of direct access and the initial length of the name directory remain the only adjustable parameters associated with bases JEVEUX . They are specified at the time of the call to routine JEINIF , one indicates also the statute of the base at the beginning of work for if required reopening an existing file, the statute at the end of the work makes it possible to avoid superfluous inputs/outputs if the base is not preserved. The reopening of a base (command POURSUITE in the Code_Aster *or* reading of the catalog of the compiled elements) requires the knowledge length of the records of the file of direct access and of the contents of certain system objects, the first record contains the data essential to the reconstitution of this various information. Routine JXLIR 1 is charged to read again the beginning of the first record: one opens file of direct access (with an index whose size is fixed at 11), one reads information at the beginning of record, then the file is closed. One can then open the file of direct access with a table of suitable index length, and read again the contents of the system objects stored on disc during the preceding execution.

The closing of a base

the operation of closing of a base, carried out by routine JELIBF , consists in releasing all the objects which are attached there, with possibly writing on disc and bringing up to date the system objects. Two loops are necessary to release the objects: the first treats the collections, the second treats the simple objects. The system objects are then discharged, the addresses disc are treated in the last, the plugs of input/output are emptied, finally one brings up to date the characteristics of the base on the first record. The file of direct access is then closed by call to routine JXFERM .

The retassage of a base During

the operations of destruction of JEVEUX object , the associated disk space is marked free but is not systematically recovered. The retassage makes it possible “to fill” the vacuums in “going up” the records. It is thus necessary to modify the attribute addresses disc of the objects contained in the records to move. This operation is immediate for the simple objects, concerning the collections it is necessary to have access to the system object containing the addresses disc (which itself can be located the record to be moved!). There is no reorganization within the records containing the images of small objects. One uses routine JETASS and one calls on alternative “JETASS″ of the routine of release JJLIDE . This utility can be directly called by the command FIN in the Code_Aster .

The recopy of the bases This

operation must be carried out to take into account indeed the retassage, the files of direct access WRITDR which cannot be reduced out of core. Routine JXCOPY works from bases closed and the restores in the same state. This utility can be called by the command FIN in the Code_Aster .

12 The printings

routine JEIMPD makes it possible to print the list of the JEVEUX objects present on one or more bases. The list is made up starting from the associated catalog (object system $$ RNOM) and one prints following information for each object:

the associated identifier in the name of object,
the name of the object,
the kind of the object,
the type of the object,
the length in the type,
the length in byte of the segment of values,
the number of the record containing the segment of values,
the position in the record for the small objects,
the number of access in reading on the record,
the number of access in writing on the record.

```
CONTENU

        OF BASE G NOM

  OF THE BASE:               MAXIMUM GLOBALE
  NB OF RECORDS: 5242 LONGUEUR
  OF RECORD (BYTES): 819200
--------------------------------------------------------------------------------

 ----

NUM ------------- NOM ----------------- G T - L - LOTY- - IADD- --LIADD- NB AC1

        ________GLOBALE ________$$CARA - V    - I 8  11     1      24       0     2
        ________GLOBALE ________$$IADD - V    - I 8  4000   1      136      0     3
        ________GLOBALE ________$$GENR - V    - K 1  2000   1      32160    0     4
        ________GLOBALE ________$$TYPE - V    - K 1  2000   1      34184    0     5
        ________GLOBALE ________$$DOCU - V    - K 4  2000   1      36208    0     6
        ________GLOBALE ________$$ORIG - V    - K 8  2000   1      44232    0     7
        ________GLOBALE ________$$RNOM - V    - K 32 2000   1      60256    0     8
        ________GLOBALE ________$$LTYP - V    - I 8  2000   1      124280   0     9
        ________GLOBALE ________$$LONG - V    - I 8  2000   1      140304   0     10
       ________GLOBALE ________$$LONO - V    - I 8  2000   1      156328   0     11
       ________GLOBALE ________$$DATE - V    - I 8  2000   1      172352   0     12
       ________GLOBALE ________$$LUTI - V    - I 8  2000   1      188376   0     13
       ________GLOBALE ________$$HCOD - N    - I 8  4177   1      204400   0     14
       ________GLOBALE ________$$USADI - V   - I 8  10484  1      237840   0     15
       ________GLOBALE ________$$ACCE - V    - I 8  5242   1      321736   the 0     …
```

printing of the memory

routine JEIMPM makes it possible to print the list of the JEVEUX objects present in memory. Following information is printed:

the class of the object,
the associated identifier of collection in the name of object or 0,
the simple identifier of object or associated object of collection in the name of object,
the value (whole) of the mark associated at the segment with values,
the relative memory address of the segment of values,
the statute of the segment of values (X or U),
the length measured in unit of addressing (whole) of the segment of values,
the state of the segment of values (X, A or D),
the name of the object (possibly supplemented by the number of object of collection).

```
OBJECTS
ALLOCATE DYNAMICALLY ------------------------------------------------------------------------------
CL

 - --  NUM-- - MY ---------  IADY--------- - U - LON UA - - S  - ------------- NOM -------------- |G
```

```
| 0|   1|       -2  | 69888784            |U| 11         | D| ________GLOBALE ________$$CARA |G
| 0|   2|       -2  | 108444896           |U| 4000       | D| ________GLOBALE ________$$IADD |G
| 0|   3|       -2  | 106261840           |U| 251        | D| ________GLOBALE ________$$GENR |G
| 0|   4|       -2  | 106263920           |U| 251        | D| ________GLOBALE ________$$TYPE |G
| 0|   5|       -2  | 105733504           |U| 1001       | D| ________GLOBALE ________$$DOCU |G
| 0|   6|       -2  | 108476976           |U| 2000       | D| ________GLOBALE ________$$ORIG |G
| 0|   7|       -2  | 108493056           |U| 8004       | D| ________GLOBALE ________$$RNOM |G
| 0|   8|       -2  | 108557168           |U| 2000       | D| ________GLOBALE ________$$LTYP |G
| 0|   9|       -2  | 108573248           |U| 2000       | D| ________GLOBALE ________$$LONG |G
| 0|   10      | -2  | 108589328           |U| 2000       | D| ________GLOBALE ________$$LONO |G
| 0|   11      | -2  | 108605408           |U| 2000       | D| ________GLOBALE ________$$DATE |G
| 0|   12      | -2  | 108621488           |U| 2000       | D| ________GLOBALE ________$$LUTI |G
| 0|   13      | -2  | 108637568           |U| 3203       | D| ________GLOBALE ________$$HCOD |G
| 0|   14      | -2  | 46912496140304      |U| 188742     | D| ________GLOBALE ________$$USADI |G
| 0|   15      | -2  | 106270864           |U| 62914      | D| ________GLOBALE ________$$ACCE |G
| 0|   16      | -2  | 105685344           |U| 4000       | D| ________GLOBALE ________$$MARQ |G
| 0|   17      | -2  | 105717424           |U| 2000       | D| ________GLOBALE ________$$INDI |G
| 0|   18      | -2  | 106774256           |U| 102400     | D| ________GLOBALE ________$$TLEC |G
| 0|   19      | -2  | 107593536           |U| 102400     | D| ________GLOBALE ________$$TECR |G
| 0|   20      | -2  | 108412816           |U| 4000       | D| ________GLOBALE ________$$IADM |G
| 0|   21      | 0|   96179360           |X| 21         | D| &FOZERO .PROL           |G
| 0|   22      | 0|   85544864           |X| 2|          D| &FOZERO .VALE           |G
| 0|   23      | 0|   71159344           |X| 1|          D| && NUM_CONCEPT_UNIQUE |G
| 0|   24      | 0|   111103824          |X| 5010       | D| &&SYS.KRESU |G
| 0|   25      | 0|   76660688           |X| 11         | D| &CATASTROPHES.ACOUSTIQUE |G
| 0|   26      | 0|   111189040          |X| 136        | D| &CATASTROPHES.ACOUSTIQUE $$        DESO…
|G
| 0|   37      | 0|   113181328          |X| 379        | D| &CATA.GD.NOMCMP $$NOM |G| 0|          38|
0|
113184432          |X|    149             | D| &CATA.GD.NOMCMP $$LONG          |G| 0| 39| 0| 113185696
|X|
150    | D|       &CATA.GD.NOMCMP $   $LONO           |G| 0| 40        | 0| 113186976|X| 149|          D|
&CATA.GD.NOMCMP $
$LUTI   |G       | 0|   41           | 0| 113188240        |X| 379| D| &CATA.GD.NOMGD |G| 0|          42|
0
| 113191344        |X   | 149           | D| &CATA.GD.TYPEGD |G|        0| 43| 0| 86518560|X
| 11   | D|       &CATASTROPHES   .MECANIQUE           |G|        0| 44| 0| 113192608|X|
15776   |      D| &CATASTROPHES   .           MECANIQUE $$         DESO… |G| 0| 135|
0|   91910000       |   U|           11| D| MA1 .GROUPENO      |G| 135| 1| 0|         78747056
|X|
1| D   | MY     1 .GROUPENO              1 |G| 135         | 2| 0| 106203168     |X|
1| D| MY       1.   GROUPENO            2 |G| 135          | 3| 0|     74182160|              X|
1| D| MY       1.   GROUPENO           3 |G| 135          | 4| 0|     79595680|X             |
1| D| MA1       .GROUPENO              4…          the printing     of the directories
system          routine             JEIMPR           make it possible to print                    the
directories
```

associated with the various opened

bases, she traverses the system object sequentially $$RNOM and prints the values of the attributes when the first character of the name is different from "?". Following information is printed: the simple identifier of object, the name of the simple object, the kind of the object, the type

of the object, the length of the type
used, the length of
the object (attribute LONMAX
or NOMMAX),
the length measured in the type
of the segment of values, the number of the record
containing the disk image, the relative memory address
of the segment of values the address of the dynamically allocated pointer
. ----------

```
------ CATALOG CLASSE G ------ ------------------------------------------------------------------------------------------

--- NUM -------------- NOM ---------------- G T L --LENGTH--- - LOTY- - IADD- --------KADM

-------  --------KDYN------- 1 ________GLOBALE ________$$CARA - VI 8 11 11 1 1090758 69888784 2 ________GLOBALE

        ________$$IADD - VI 8 4000    4000       1 5910272                    108444896           3 __
       _  _GLOBALE ________$$GENR - V-K- 1    2000 2000     1    5637390                 106261840          4 __
```

```
      _  _GLOBALE ________$$TYPE - V-K- 1    2000 2000     1    5637650                 106263920           5 __
      _  _GLOBALE ________$$DOCU - V-K- 4    2000 2000     1    5571348                 105733504           6 __
      _  _GLOBALE ________$$ORIG - V-K- 8    2000 2000     1    5914282                 108476976           7 __
      _  _GLOBALE ________$$RNOM - V-K-32    2000 2000     1    5916292                 108493056           8 __
      _  _GLOBALE ________$$LTYP - VI 8    2000 2000     1    5924306                108557168           9 __
      _  _GLOBALE ________$$LONG - VI 8    2000 2000     1    5926316                108573248           10 _
      _ __GLOBALE ________$$LONO - VI    8 2000     2000    1 5928326                 108589328           11 __
     __ __GLOBALE ________$$DATE - VI    8 2000     2000    1 5930336                 108605408           12 __
     __ __GLOBALE ________$$LUTI - VI    8 2000     2000    1 5932346                 108621488           13 __
     __ __GLOBALE ________$$HCOD - VI    8 3203     3203    1 5934356                 108637568           14 __
          __ __GLOBALE ________$$USADI  -  VI        8  188742          188742        2           5864054372198
46912496140304
      15 ________GLOBALE _____ _$$ACCE   - VI 8  62914      62914       1 5638518      106270864 16
     __ ____GLOBALE ________$$MARQ - VI    - 8 4000    4000   0 5565328                 105685344          17 __
     __ __GLOBALE ________$$INDI - VI    8 2000     2000    0 5569338                 105717424          18 __
     __ __GLOBALE ________$$TLEC - VI    8 102400     102400    0      5701442            106774256          19
     __ ______GLOBALE ________$$TECR -V    - I 8 102400   102400  0      5803852            107593536          20
      __  ______GLOBALE ________$$IADM - V     - I 8 4000    4000  0 5906262                  108412816           21
&FOZERO
       .PROL - V-K-24 6 6 0 4377080    96179360     22    &FOZERO                  .VALE           - V-R- 8
      2 2  0 3047768            85544864          23        &       &_NUM       _             CONCEPT_UNIQUE            -
V
      - I  8 1 1           0 1249578          71159344             24                  &&SYS.KRESU           - V-K-
       80  500 500 0 6242638 111103824                        …             the printing              of            the
attributes
       routine JEIMPA                        prints          all                 the attributes          for
```

a JEVEUX object

. WRITING

OF THE ATTRIBUTES FROM "MY 1 .DIME" JEIMPA PRINTING OF THE ATTRIBUTES OF >MA1 .DIME

```
 < CLAS G GENR V TYPE I LTYP     4            DOCU

X  GOES BACK 0 LONMAX 6 LONUTI 6 LONO 6     IADM

20357178
IADD 0
LADD 0
USE D               WRITING

OF              THE ATTRIBUTES
           TO
"MA1.              CONNEX
              "
JEIMPA         PRINTING
OF
the ATTRIBUTES
OF >MA   1

  .CONNEX < access NU STOCKAGECONTIG     MODELONGVARIABLE

 NMAXOC 204 NUTIOC 0 LONT 1472     CLAS         G

GENR   V TYPE
I LTYP 4 DOCU
U DATES
-1292845870           LONO
1472             IADM
          20270746
IADD 0
LADD 0
USE    D
Note:
: The printing
    of the attributes
           of the objects
       of collection
or
their
contents   can
```

require

the setting in memory of the attribute objects, and their release at the end of the action. A temporary mark equal to - 2 is affected in this case. The printing of the contents of a segment of values routine JEIMPO makes it possible to print

the segments of values associated with a JEVEUX object

. PRINTING SEGMENT OF VALUES >MA1 .DIME < >>>>> 1 - 361 0 204 0 0 6 - 3 PRINTING

```
 SEGMENT OF VALUES >MA1 .COORDO     .                   VALE
 < >
      >>>>          1 -            0.00000                     D+00
      5.00000

         D-01 1.00000D+00 5.00000D-01 5.00000     D-01    6 - 1.00000
         D+00
              1.00000  D+00 5.00000  D-01 1.00000  D+00 0.00000  D+00 11 -  7.50000D-01
              1.00000  D+00 5.00000  D-01 7.50000  D-01 1.00000  D+00 16  - 1.00000D+
             00 7.50000  D-01 1.00000  D+00 0.00000  D+00 5.00000  D-01 21  - 2.00000D+
             00 5.00000  D-01 5.00000  D-01 2.00000  D+00 1.00000  D+00 26  - 5.00000D-
             01 2.00000  D+00 0.00000  D+00 7.50000  D-01 2.00000  D+00 31  - 5.00000D-
             01 7.50000  D-01 2.00000  D+00 1.00000  D+00 7.50000  D-01 36  - 2.00000D+
             00 5.00000  D-01 1.00000  D+00 1.00000  D+00 1.00000  D+00 41  - 1.00000D+
             00 1.00000  D+00 5.00000  D-01 1.00000  D+00 2.00000  D+00 46  - 1.00000D+
             00 1.00000  D+00 2.00000  D+00 0.00000  D+00 1.00000  D+00 51  - 1.00000D+
             00 0.00000  D+00 1.00000  D+00 2.00000  D+00 0.00000  D+00 56  - 5.00000D-
             01 3.00000  D+00 5.00000  D-01 5.00000  D-01 3.00000  D+00 61  - 1.00000D+
             00 5.00000  D-01 3.00000  D+00 0.00000  D+00 7.50000  D-01 66  - 3.00000D+
             00 5.00000  D-01 7.50000  D-01 3.00000  D+00 1.00000  D+00 71  - 7.50000D-
             01 3.00000  D+00 5.00000  D-01 1.00000  D+00 3.00000  D+00 76  - 1.00000D+
             00 1.00000  D+00 3.00000  D+00 0.00000  D+00 1.00000  D+00 81  - 3.00000D+
             00 0.00000  D+00 0.00000  D+00 1.00000  D+00 5.00000  D-01…  APPENDIX 1:
             Description  of the commun runs  used  in the manager  of  JEVEUX memory
```

13 /FENVJE/ INTEGER LFIC, MFIC COMMON /FENVJE/ LFIC, MFIC LFIC length maximun in bytes

of a extend
, MFIC cuts maximum
in bytes of disk space

	usable. /IACCED//JIACCE/ PARAMETER
(N	= 5) COMMON /IACCED/ IACCE (1) COMMON /JIACCE/ JIACCE

(N) variable of

reference and position
of the segment of values
associated with the system object

	with suffix: IACCE, JIACCE $$ACCE /IADMJE/ INTEGER IPGC, KDESMA, LGD, LGDUTI, KPOSMA, LGP, LGPUTI
COMMON /IADMJE/	IPGC

, KDESMA

, LGD, LGDUTI , KPOSMA, LGP, LGPUTI IPGC Value of the current
mark (varies between -3 and N), KDESMA addresses segments of

values	containing the addresses of the marked objects, LGD
length	of the segment of value associated to KDESMA, LGDUTI length used of the segment
	of value associated to KDESMA, KPOSMA addresses
segments	of values containing the positions associated with each mark
, LGP	length of the segment of value associated to KPOSMA, LGPUTI length used of the segment
	of value associated to KPOSMA. /IATCJE/ INTEGER
ICLAS	, ICLAOS, ICLACO, IDATOS, IDATCO, IDATOC COMMON /IATCJE/

ICLAS, ICLAOS

, ICLACO , IDATOS, IDATCO, IDATOC ICLAS classifies current
, ICLAOS classifies simple object, ICLACO classifies collection

, identifying	IDATOS
of	the simple object, identifying IDATCO
	of the collection, identifying
IDATOC	of the object of collection
. /IATRJE//JIATJE/	PARAMETER (N = 5)
INTEGER	LTYP, LENGTH, DATE, IADD, IADM

, + LONO, HCOD

, CARA, LUTI, IMARQ
COMMON /IATRJE/ LTYP (1), LENGTH (1) , DATE (1)
, IADD (1), IADM (1), + LONO (1) , HCOD (
1), CARA (1), LUTI (1), IMARQ (1) COMMON /JIATJE/ JLTYP (N), JLONG (
N), JDATE (N), JIADD (N), JIADM (N), + JLONO (N), JHCOD

(N), JCARA (N), JLUTI (N), JMARQ (N) variable of reference and position
of the segment of values associated with the system object

with suffix: LTYPE, JLTYP $$LTYP LONG, JLONG $$LONG DATE, JDATE $$DATE IADD, JIADM $$IADD IADM

, JIADD $IADM	$	LONO
, JLONO $LONO	$	HCOD
, JHCOD $HCOD	$	CARA
, JCARA $CARA	$	LUTI
, JLUTI $LUTI	$	IMARQ
, JMARQ $MARQ	$	/ICODJE/
INTEGER		NUMATR
COMMON identifying		/IDATJE/
NUMATR NUMATR of the system		object

of

collection /IDATJE/ PARAMETER ($$LONO

N = 5) INTEGER NRHCOD, NREMAX, NREUTI COMMON /ICODJE/ NRHCOD (N),

NREMAX (N) , NREUTI (N) NRHCOD cuts (in integer) system object $$HCOD NREMAX cuts (in integer)

system object	$$RNOM NREUTI length used of
the system object	$$RNOM /IENVJE/ INTEGER LBIS, MODELS,
LOLS,	RENTED, LOR8, LOC8 COMMON /IENVJE/ LBIS , MODELS

, LOLS

```
                , RENTED,          LOR8, LOC8 LBIS length out of bits of the
integer
                , MODELS length  in bytes of the integer, LOLS length
```

in bytes	of logic, RENTED length
in	bytes of the unit of addressing,
LOR8	length in bytes of reality, LOC
8 length	in bytes of the complex. /IEXTJE/ PARAMETER
	(N = 5) INTEGER IDN, IEXT
,	NBENRG COMMON /IEXTJE/ IDN (N)

```
              , IEXT (N)

                    , NBENRG (N  ) IDN
                    is not used any more             since    the use
                    of the named accesses  . IEXT number of extends open
```

	NBENRG maximum number of records of a extend /IFICJE/
	PARAMETER (N = 5) INTEGER
NBLMAX	, NBLUTI, LONGBL, + KITLEC, KITECR

```
              , KINDEF

                    , KIADM      , + IITLEC
                    ,          IITECR    , NITECR    , KMARQ COMMON
                                    /IFICJE/    NBLMAX (N    ), NBLUTI    (N),
LONGBL
                   (N                 ), + KITLEC    (N),    KITECR (N)    ,
KINDEF
                    (N), KIADM (N),  + IITLEC (N), IITECR (N), NITECR (N
                   ),                 KMARQ (N) For each base associated with the
index
                   I                 ranging between 1 and N NBLMAX number
maximun
```

d'enregistrements, NBLUTI many records used, LONGBL

length	in bytes of the records
, KITLEC	addresses in K1ZON of the segment
of values	associated with the plug with reading, KITECR
addresses	in K1ZON of the segment of values associated with the plug with writing, KINDEF
addresses	in ISZON of the segment of values associated with the index used for
the files	with direct access , KIADM addresses in ISZON of the segment of values associated with the memory addresses, IITLEC
addresses	disc of the plug of reading (record number), IITECR addresses
disc	of the plug of writing (record number), NITECR cuts
in bytes	of the portion of the plug of writing used, KMARQ address
in ISZON	of the segment of values associated with marks. /ILOCJE/ INTEGER
ILOC	COMMON pointer /ILOCJE/ ILOC ILOC on the address of the beginning of

the memory zone

allocated
by JXALLM. /INUMJE/

INTEGER NUMEC COMMON /INUMJE/ NUMEC NUMEC number of the object of collection

or number

of insertion
in a name directory

. /ISTAJE/ INTEGER ISTAT COMMON /ISTAJE/ ISTAT (4) ISTAT codes associated with the state and

to the statute

of the segments of
values ISTAT (1) corresponds

to X ISTAT (2) corresponds to U ISTAT (3) corresponds to A ISTAT (4) corresponds
to D /IXADJE/ INTEGER
IDINIT, IDXAXD COMMON
/IXADJE/ IDINIT, IDXAXD
IDINIT is worth 5, beginning of

the managed

memory zone, IDXAXD
initial position in ISZON

for	the search. /IZONJE/ INTEGER LK1ZON, JK
1ZON,	LISZON, JISZON, ISZON (1) COMMON /IZONJE/ LK1ZON

, JK1ZON

, LISZON , JISZON EQUIVALENCE (ISZON (1), K1ZON (1)
) LK1ZON length in character (CHARACTER*1) of
the managed zone , JK1ZON addresses in

K1ZON of	the beginning of the zone, LISZON length in integer (INTEGER*8
in addressing	64 bits , INTEGER *4 in addressing 32
bits)	of the managed zone, JISZON addresses in ISZON of the beginning of the zone, ISZON memory zone in integer (INTEGER
*8 in	addressing 64 bits, INTEGER*4 in addressing
32 bits	) managed dynamically; this table of the whole type is not part of the variables deposited in the commun run, but it is put in equivalence with the table of type character. The order EQUIVALENCE makes it possible to align the two tables of the whole type and nature

in order to be able to use one indifferently or the other following the needs. /KUSADI//JUSADI/ PARAMETER (N = 5) COMMON /KUSADI/ IUSADI (1) COMMON /JUSADI/ JUSADI

(N) variable of

reference and position
of the segment of values
associated with the system object

with suffix: IACCE, JIACCE $$USADI /JCHAJE/ INTEGER ILLICI, JCLASS (0:255) COMMON /JCHAJE/ ILLICI

, JCLASS ILLICI	is worth

-1, JCLASS

is affected with result of
function ICHAR on the licit

characters is worth	, if not ILLICI. /JENVJE/ INTEGER MSLOIS COMMON /JENVJE/ MSLOIS MSLOIS masks being worth the sum of the weights

of the MODELS

- the first 1 integers
, intended to replace

the operation modulo (MODELS) by function AND /JCONJE/ INTEGER MSSTAT, LSSTAT COMMON /JCONJE/ MSSTAT, LSSTAT MSSTAT is not used more

, mask

being worth the sum of
the weights of the LSSTAT first integers

. LSSTAT	(LBISEM - 4) where LBISEM is the length out of bit of the integer, is used in
JELIRA	to obtain the equivalent in the form of character of the statute or the state associated with a segment with values. /KATRJE/, /JKATJE PARAMETER

(N = 5) CHARACTER*1 GENR, TYPE CHARACTER*4 DOCU CHARACTER

```
*8 ORIG CHARACTER

        *32  RNOM COMMON
        /KATRJE/      GENR (8    ), TYPE
        (8), DOCU (2)       , ORIG
        (1), RNOM       (1) COMMON
        /JKATJE/     JGENR
        (N), JTYPE (N),  JDOCU (N), JORIG (N), JRNOM (N) variable of
reference
        and position  of the segment of values associated with the
system object
```

with suffix: GENR $$GENR GENR $$TYPE DOCU $$DOCU ORIG $$ORIG RNOM $$RNOM /KBASJE/ PARAMETER (N =

5)	CHARACTER
*8	NOMBAS
COMMON	/KBASJE/
	NOMBAS
(N) NOMBA	name

of the base

(used for
the messages d'error)
/KFICJE/ PARAMETER (N =

5) *2 DN2 CHARACTER*5 CLASSE CHARACTER*8 NOMFIC
CHARACTE
R

, KSTOUT

, KSTINI COMMON /KFICJE/
CLASSE
, NOMFIC (N), KSTOUT
(N), KSTINI (N), + DN2 (N) DN2
unutilised
! CLASSE makes it possible to store all the names of open
classes
(first

	letter of
the basic name	) NOMFIC name of open bases KSTOUT statute in output of bases (“SAVES” or “DESTROYS
”) KSTINI	statute in poursuite
of the bases	(“DUMMY”, “debut” or “CONTINUE”) /KNOMJE/ CHARACTER
*24	NOMEC COMMON /KNOMJE/ NOMEC NOMEC name of object of collection

or name

to be inserted in a directory
/KZONJE/ CHARACTER

*1 K1ZON COMMON /KZONJE/ K1ZON (8) K1ZON memory zone in character

(CHARACTER

*1) managed dynamically
. /NFICJE/ INTEGER

NBCLA COMMON /NFICJE/ NBCLA NBCLA many classes opened simultaneously

/NOMCJE/ CHARACTER
*24 NOMCO CHARACTER

*32 NOMUTI, NOMOS, NOMOC, BL32 COMMON /NOMCJE/

NOMUTI

, NOMOS, NOMCO
, NOMOC, BL32 NOMUTI name used (in

the calls to the routine I….), NOMOS name of the simple object

, NOMCO	name of the collection, NOMOC name of the object of collection
	, BL32 chain white
length	32. /UNDFJE INTEGER
	LUNDEF, IDEBUG COMMON /UNDFJE/
LUNDEF	, affected IDEBUG LUNDEF in JEDEBU

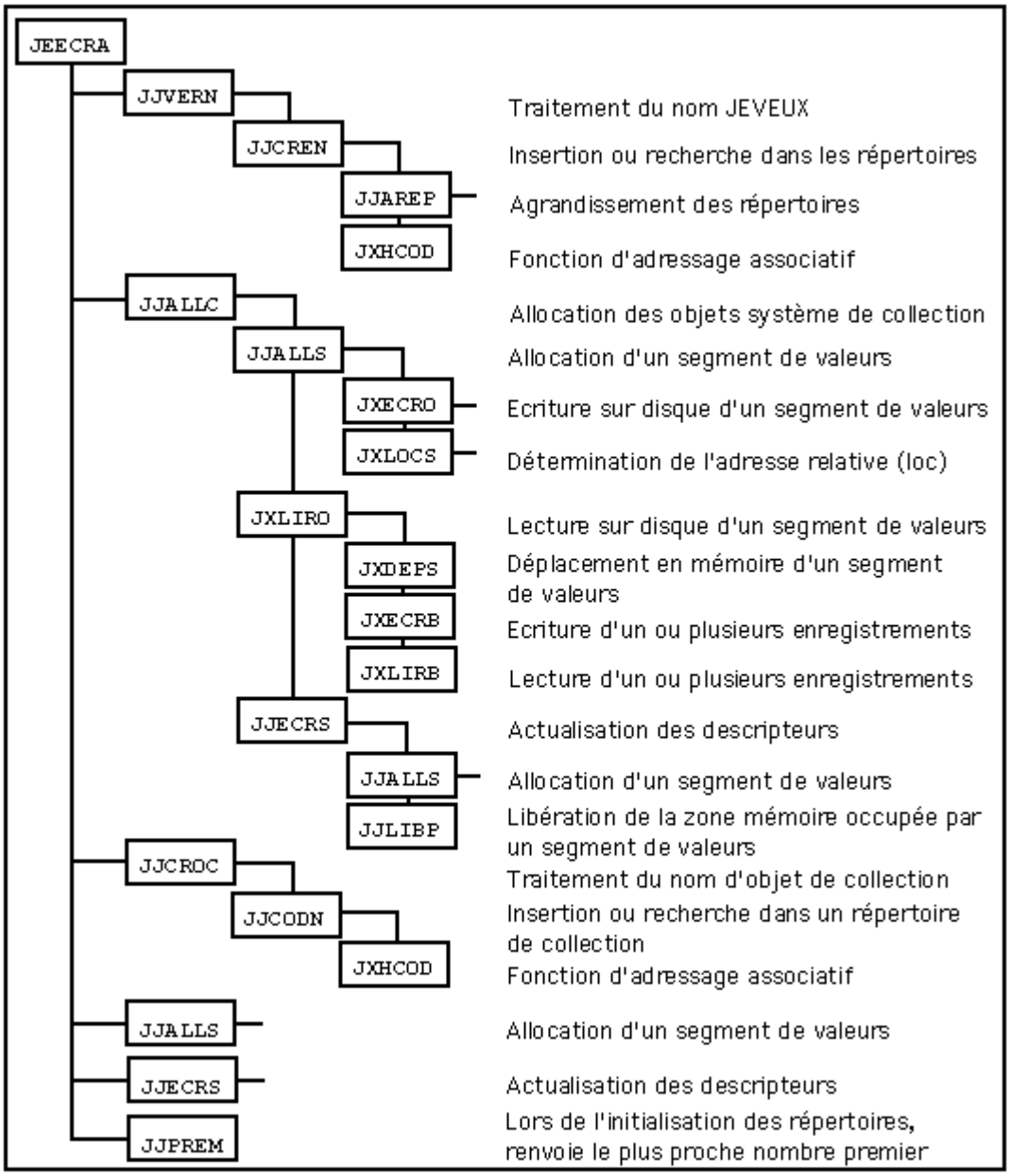
by

the function envima ISNNEM (not has number), IDEBUG is worth

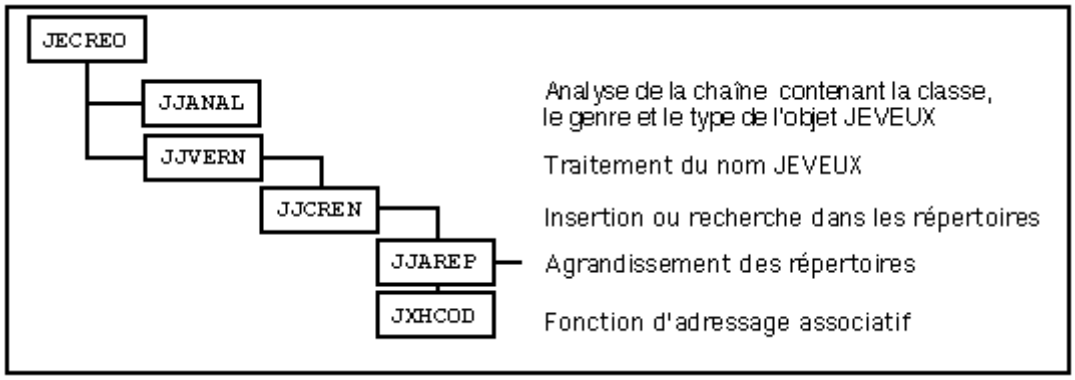
1 in	debugg mode, 0 if not. APPENDIX 2: Shafts of call simplified of some
under	- programs One presents below

14 the shafts of principal subroutines JEVEUX, one has voluntarily

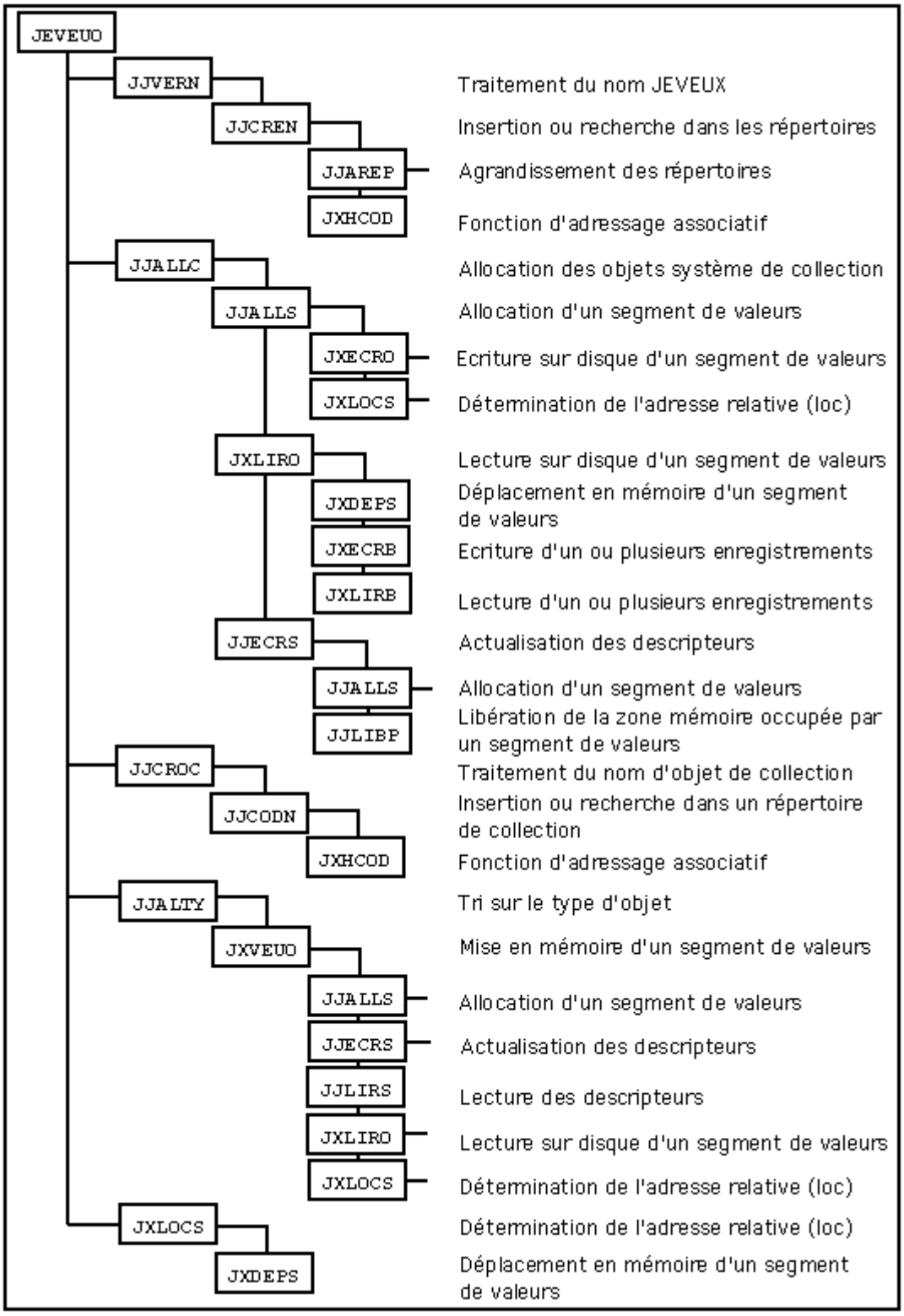
restricted on three levels of subroutines to facilitate comprehension. The truncated branches indicate that there exist other calls JEVEUX in the subroutine. Shaft of call of routine JEECRA Shaft of call of routine JECREO Shaft of call



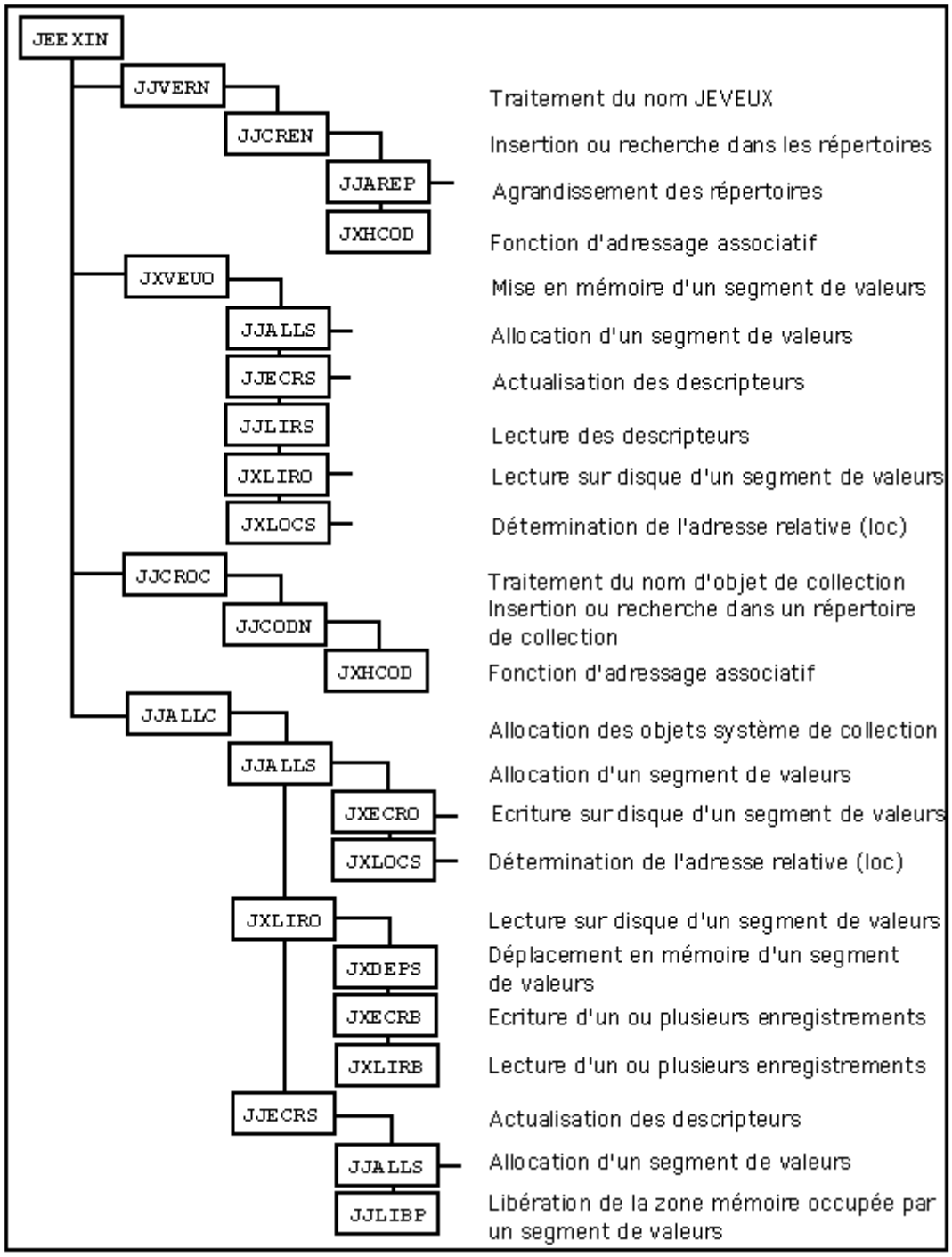
of routine JEVEUO Shaft `of call`



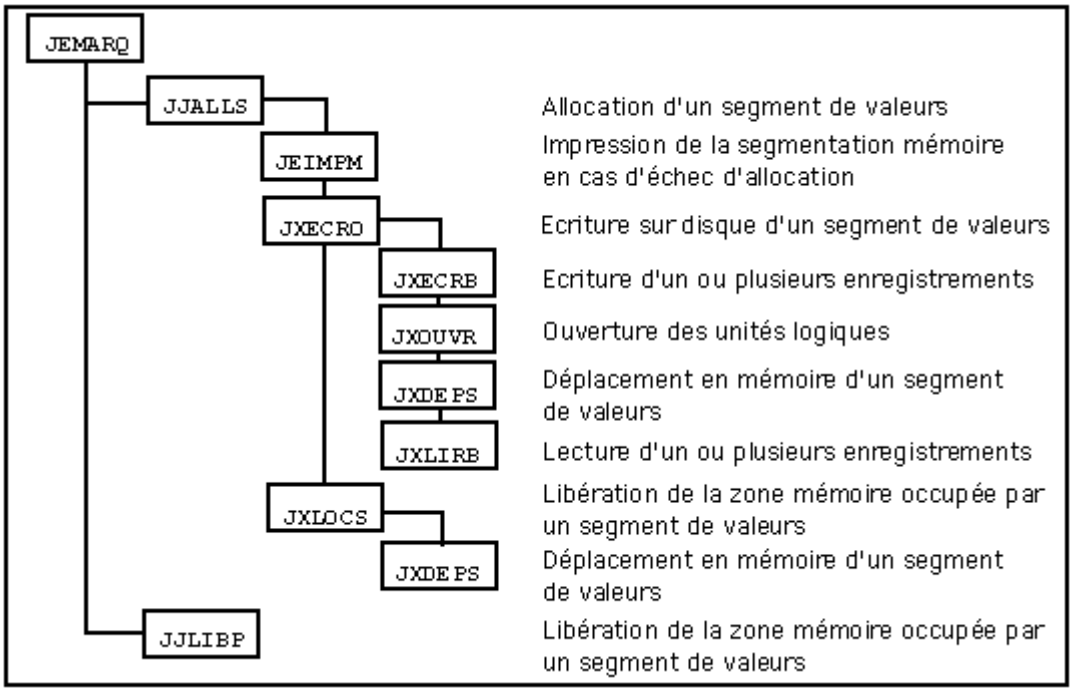
of routine JEEXIN Shaft `of call`



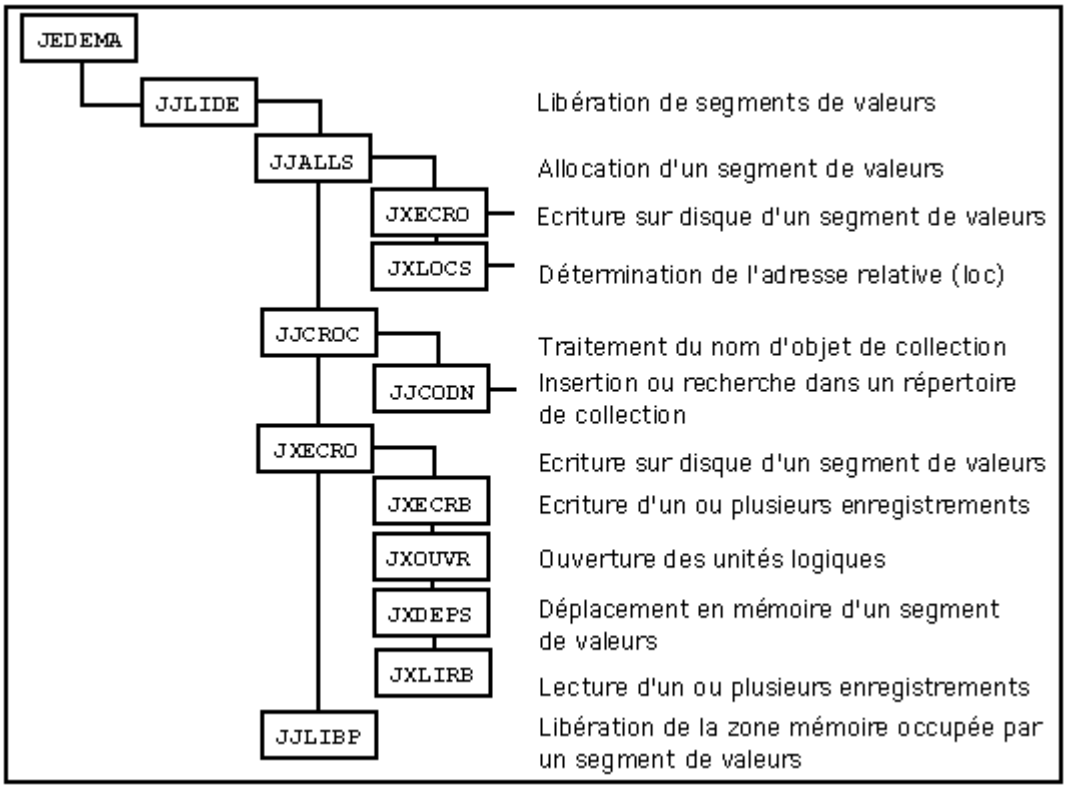
of routine JEMARQ Shaft `of call`



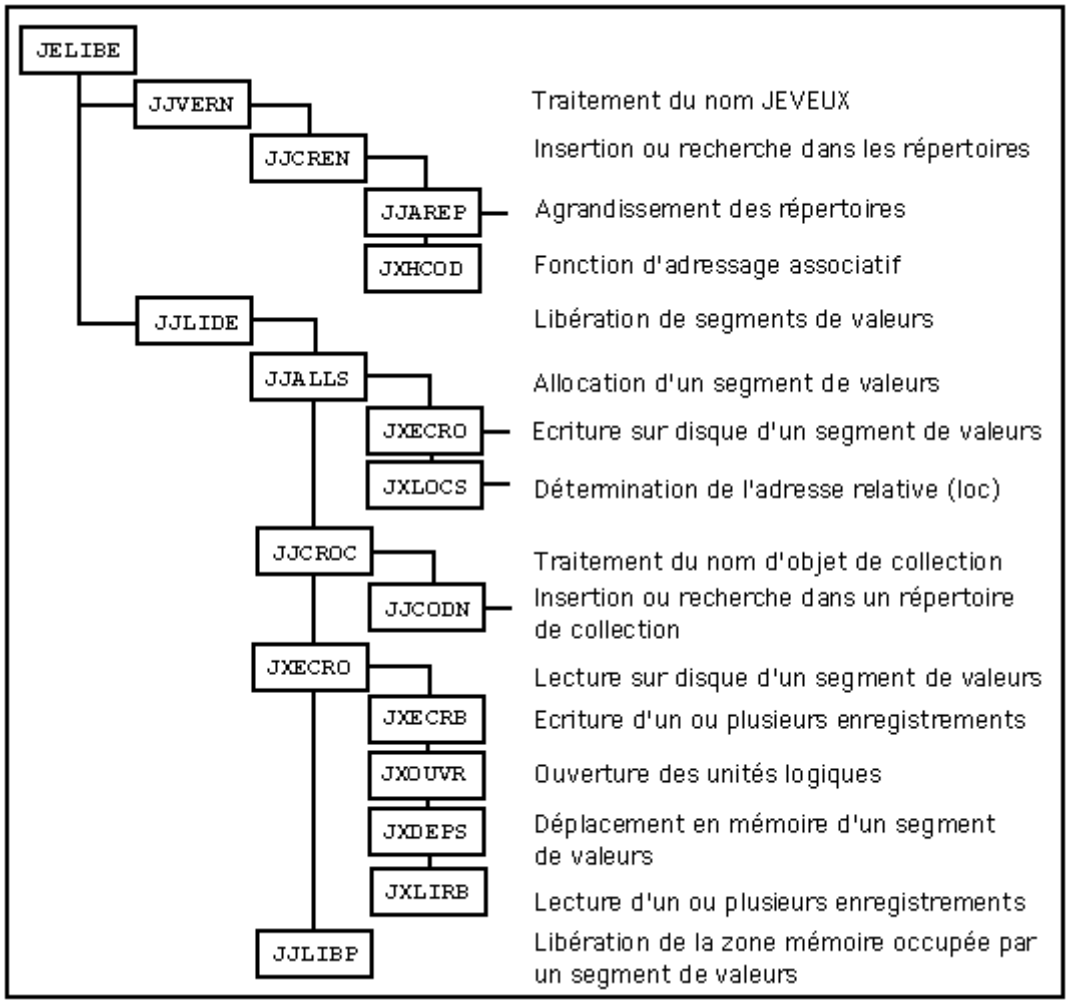
of routine JEDEMA Shaft `of call`



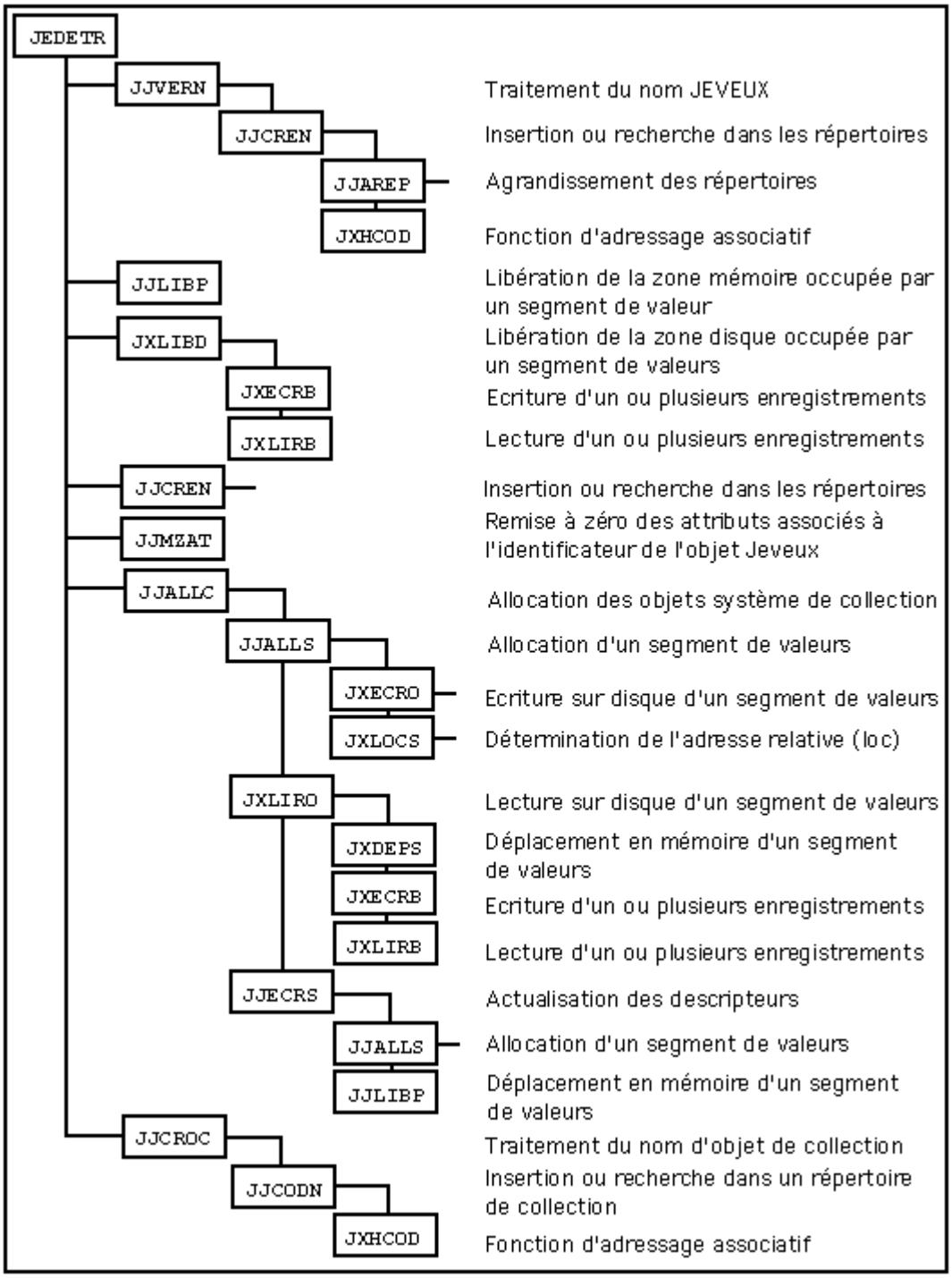
of routine JELIBE Shaft `of call`



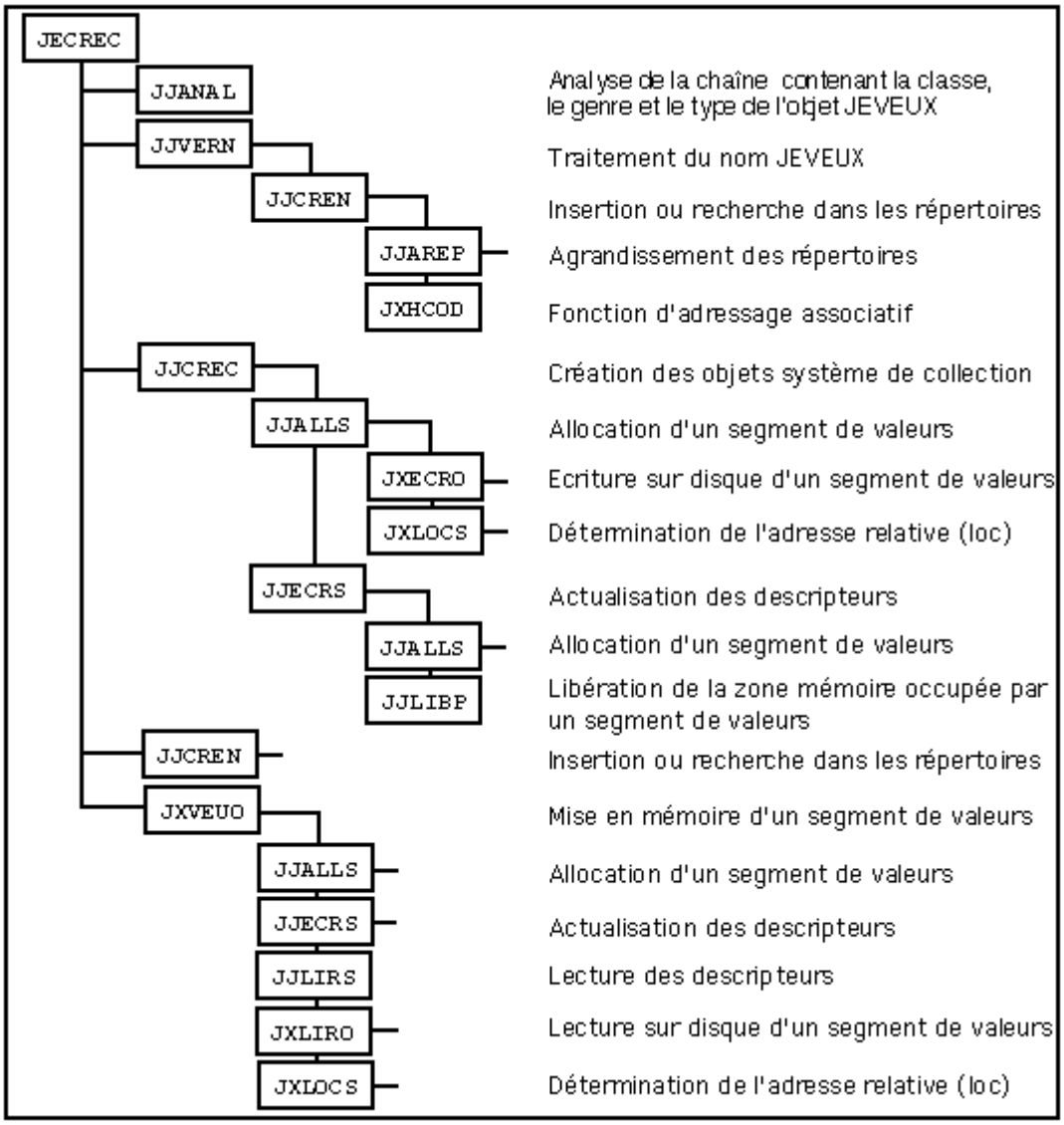
of routine JEDETR Shaft `of call`



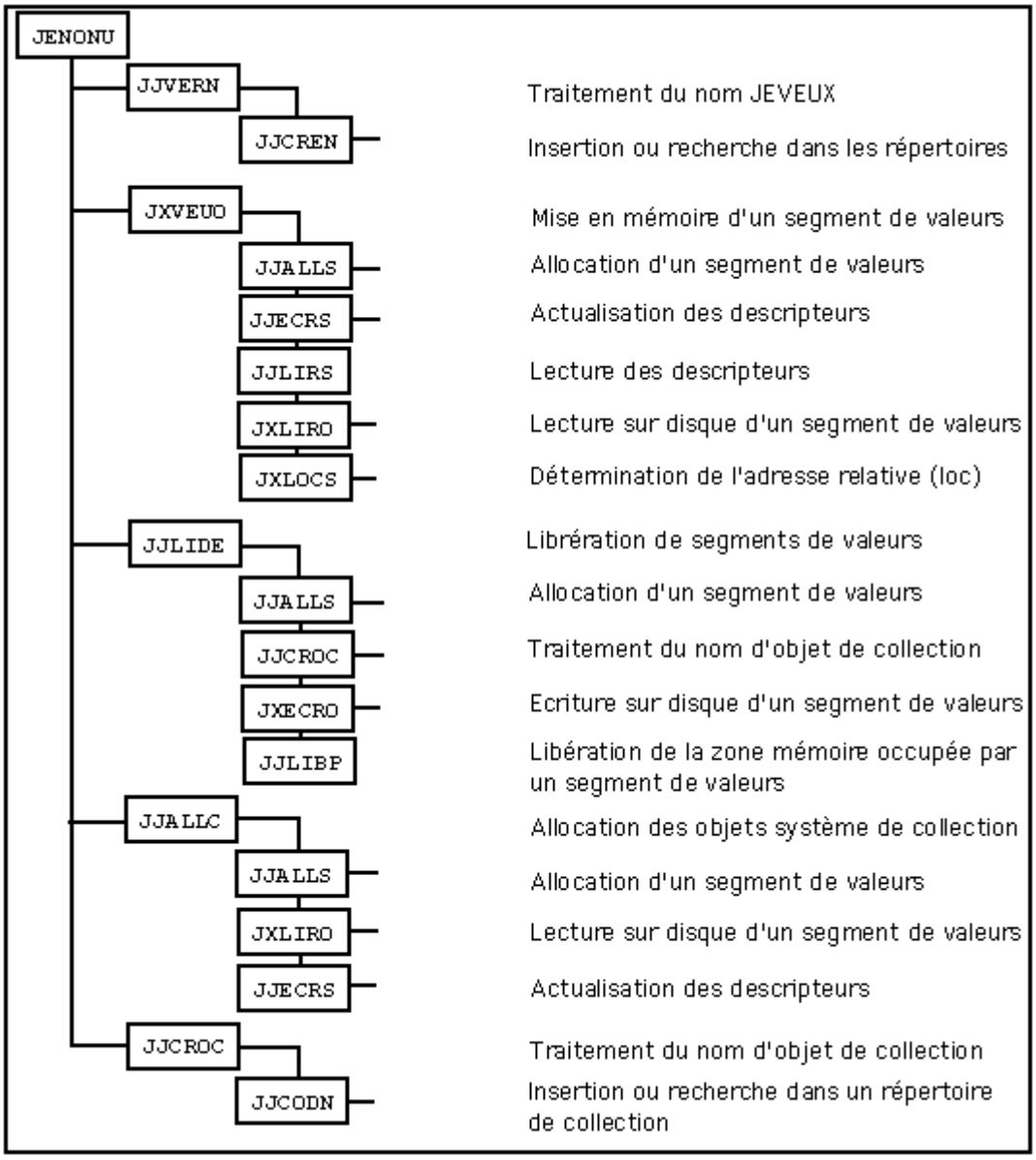
of routine JECREC Shaft `of call`



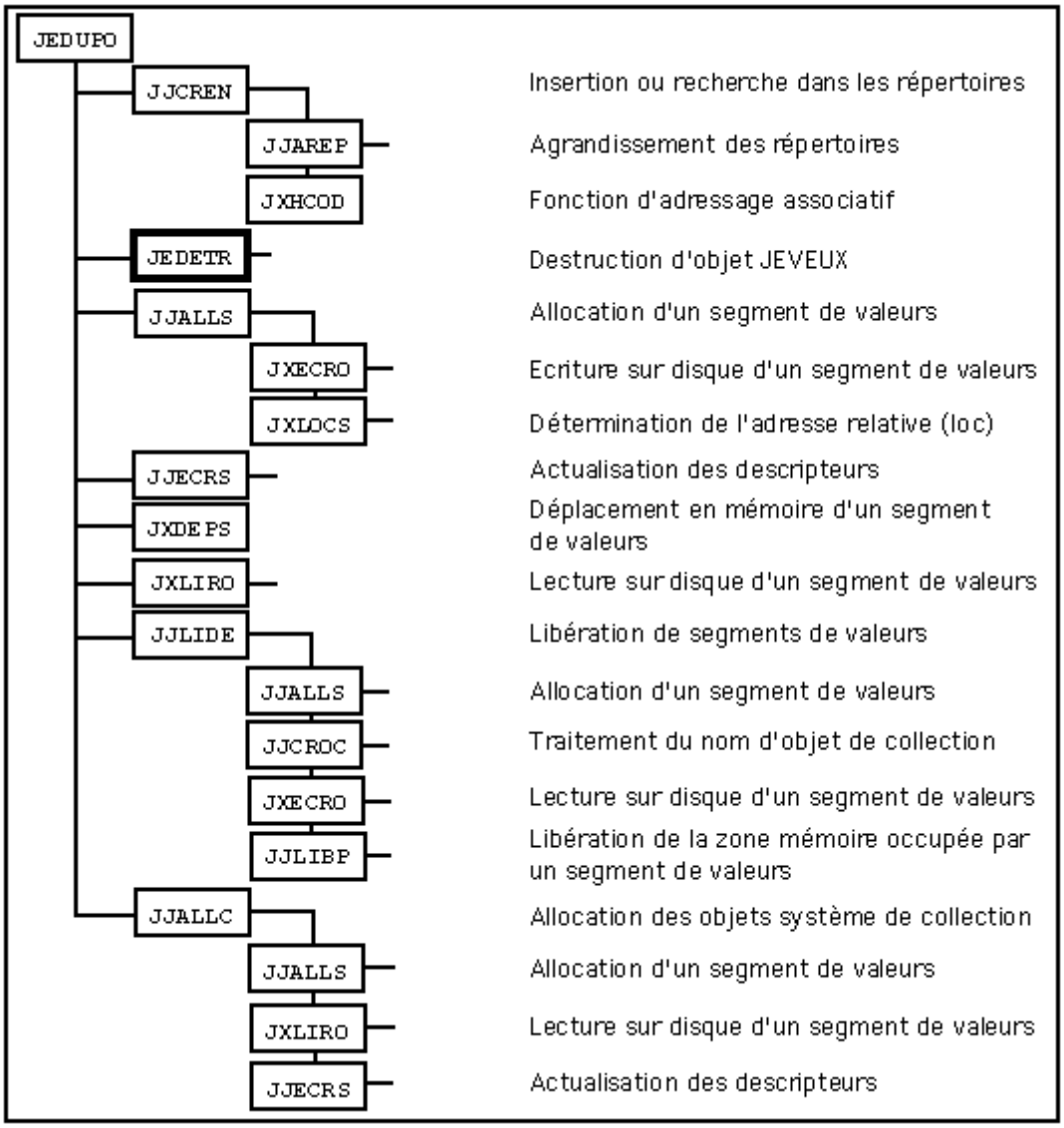
of routine JENONU Shaft `of call`



of routine JEDUPO APPENDIX `3: List`



subroutines and `their`



functions principal JECREC `Creation`

15 of a collection JECREO Creation of an object simple JECROC Creation

of an object	of collection JEDEBU Initialization
	of the parameters of software
	JEDEMA Décrémente the mark and releases
	objects JEDETC Destroyed a set of objects
JEDETR	Destroys an object JEDETV Destroyed the objects
of	Volatile base JEDISP Determines
	more big spaces
available	in memory JEDUPC Duplicates a set of
	objects JEDUPO Duplicates an object JEECRA Affects the attributes
	of an object JEEXIN Tests the existence
of	a descriptor of object
JEFINI	Finishes the execution of software
JEIMPA	Prints the attributes of an object JEIMPD
Prints	the list of the objects present
on	a basis JEIMPM Prints the contents of
the segmentation	memory JEIMPO Prints the contents of the segment
of	value of an object JEIMPR Print the contents of
a directory	JEINIF Initializes the parameters associated
with	a base JELIBE Releases an object JELIBF
Releases	all the objects associated with a base JELIBZ
Releases	all
the objects	associated with the mark -1 JELIRA Reading with the value
with	an attribute with an object JELSTC Turns over the list of the objects
containing	a character string in their identifier
	JEMARQ Increments current mark JENONU Determines the number of object of collection
according to	name JENUNO Determines
the name	of object of collection according to number JEPRAT Prints
the contents	of the objects system JERAZO Gives to 0 the contents of an object
JETASS	Moves the records within
the file	direct access in order to recover
	open space JEVEMA Delivers the value of current mark JEVEUO Returns the position in the table
Z	. segment of values associated with an object
JEVEUS	Returns the position in the table Z. segment of values associated with an object
and	positions the mark with -3 JEVEUT Returns the position in the table Z. segment of values associated with an object
and	positions the mark with -1 JEXATR Function of synchronization giving access the vector cumulated lengths
	of a contiguous collection variable length JEXNOM Function of synchronization making it possible to reach by name an object of collection

JEXNUM	Function of synchronization making it possible to reach by number an object of collection
	JJALLC Assignment of the system objects of collection JJALLS Assignment of a segment of
Tri	values JJALTY on the type of object (before setting
in memory	of a segment of values) JJANAL
Analyzes	character string containing the class, the kind and the type of object JJAREP
Enlarging	of the directories JJCODN Insertion or search in a directory
of	collection JJCREC Creation of
the system	objects of collection JJCREN Insertion or search in
the directories	of bases JJCROC Processing of the name
of object	of collection JJECRS Actualization of descriptors
JJIMPO	Printing of the contents of a segment of values
JJLIDE	Release of the segments
of values	JJLIRS Reading of descriptors JJMZAT Given
to	zero of the attributes associated with the identifier
	with a segment with values
JJPREM	At the time of the initialization of a directory, returns the nearest prime number
(appearing	in one) JJVERN Processing dated from the name Jeveux JXALLM Dynamic allocation of the memory zone managed
by	software JXCOPY Recopies
file	of direct access associated with a base with elimination with
record	marked unutilised JXDEPS Displacement in memory with a segment with values JXECRB Writing with one or more
records	JXECRO Writing on disc with a segment
with	values JXFERM Closing with a file with direct access
	JXHCOD Function with hash-coding
JXLIBD	Release with the zone disc occupied
by	a segment with values JXLIBM final
Release	with memory zone dynamically allocated JXLIR1 Reading with
the first	record associated with a base in order to recover its
characteristics	JXLIRB Reading of one or more records JXLIRO Reading on disc of a segment
of values	JXLOCS Determination of the address
relative	of a segment of values JXOUVR Opening
of the logical	units associated with bases JXVERI Examines the integrity
of	the segmentation memory JXVEUO Put in memory of
a segment	of values APPENDIX 4: GLOSSARY Bases files
of	direct access Together: the global database

16 is made up by the files

glob	.1, glob.2, glob.3,... Class Named by the first letter of the

	base, the class makes it possible to associate a JEVEUX object
	with a file. Identifier of JEVEUX object For a simple object or a collection it is the sequence number
of insertion in the directory	of the base, for an object of collection it is the identifier of collection and the sequence number of insertion in the collection. Descriptor of a segment of values One of the 8 integers framing a segment of values
in memory or one of the 3 integers preceding	a segment by values on a record (disc or plug) File of direct access File from which the records are accessible directly
by name or number JEVEUX object	Indicates at the same time the simple objects, the collections and the objects of
collection Object	simple Object from which the attributes are directly accessible among the system objects
associated	with the various classes Objects with collection Objects whose attributes are shared and managed in
simple objects created	with the collection Segment of values Together of the values associated with a JEVEUX object and positioned
in a contiguous way	in memory or on disc Object system simple Objects managed by the software and intended to collect
the values of the various	attributes. Debug Jeveux Option of use of Jeveux allowing during to cause the immediate
unloading of	the segments of values releases and the assignment with value UNDEF or not of the released segment.