Documentation of development and maintenance of the manager of memory JEVEUX

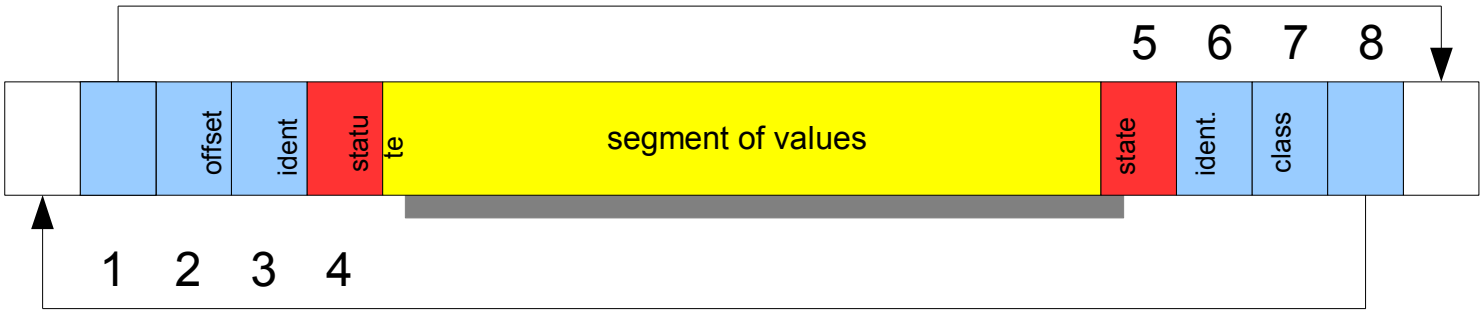
1 Introduction

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 memory JEVEUX allowed to mitigate part of these disadvantages by giving the following opportunities:

- **dynamic allocation** zones memories allocated in the course of work,
- **management of the overflows report** on file, with filing of the results in the end of work,
- **structuring of the data** *Code_Aster*, with access by name to the handled objects and standardisation of types FORTRAN used.

This document is intended for the documentation and the maintenance of the routines of the manager of memory JEVEUX prefixed by JJ, I or JX, routines JX call on nonportable functions in general. Thereafter, we will indicate this whole 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 refer to documentation [D6.02.01] Management Memory JEVEUX who describes the interface of the routines “user” I…. .

2 Organization of the memory



JEVEUX allocates each zone memory intended dynamically to accommodate the values associated with the object or the collection with objects. A control of cumulated space is carried out during each new allowance, and a mechanism of unloading can possibly be put in œuvre 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 allowance of the zone memory 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) of length LISZON and of an address of beginning JISZON. It is stored in the commun run /IZONJE/. One will use in the continuation of the document the term “word” to indicate the unit of addressing. On platform 64 bits the word has as a length 8 bytes and corresponds to length of the type INTEGER*8.

The zone is managed word by word in unit of the type INTEGER (unit of addressing), segments of values associated with the objects JEVEUX are framed by 8 words 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 worthless 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 word (8 bytes). It is even more frequent with the 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 entirety);
5) the state of the segment of values which can take the value X, With or D (coded on an entirety); ;
6) the whole identifier associated with the objects of collection;
7) the code of the class associated with the object JEVEUX ;
8) the address of the first word preceding the segment by values and the 8 identifiers.

The management of the zone memory with the type INTEGER does not allow to align itself 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 little chance has to be aligned with a “multiple” of 4 of the 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 who is called mainly at the time of 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 With,
02000000000000000000	for U,	04000000000000000000	for D.

Use of the segments of values

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

- XX indicate that the segment of values is free, this position is directly usable,
- UA indicate that the segment of values is used in reading (its image disc will not be brought up to date after release),
- UD indicate that the segment of values is used in writing (its image disc will be brought up to date after release),
- XD indicate that the segment of values was released, but that its contents will have to be discharged on disc,
- XA indicate 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 an object JEVEUX cause, if it does not appear there already, a loading in memory of the contents of the associated segment of values. The address memory of an object JEVEUX corresponds to its relative position in the table ISZON. As a preliminary, it is necessary to carry out a dynamic allocation, using the routine JJALLS to insert the segment of values. When the request of allowance fails, the system refusing to allocate a zone memory, 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 entireties which correspond to minimum space associated with a segment with values (1 entirety by descriptor). When the search for place memory fails, one causes a stop of the application in error <S> (stop by the supervisor with safeguard of the concepts created).

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

The allowance 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 the type CHARACTER *24) automatically does not allow to align itself compared to the table ISZON, it is sometimes necessary to shift few words. 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 with the segment with values, this operation is carried out by the routine JJECRS.

Search for place available

The call to the routine JEDISP allows to know at the time of the call, the size of the zones memory available, it carries out research by traversing the whole 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 chaining front can be detected by using the 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 place. It is then necessary to cause unloadings on disc or to recover the place of the zones become useless. JEVEUX deals with these mechanisms, in 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 an address memory 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 objects JEVEUX“.

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

The setting in memory of an object JEVEUX be 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 the routine JEMARQ and décrémenté with each call to the routine JEDEMA. It is possible to obtain the value of the current mark by calling the 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 order Aster) certain objects which will be released by a specific call. This mark is used at the time of the call to the 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 keep permanently (throughout the execution of *Code_Aster*) 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 the routine JEVEUS.

One thus builds a hierarchy of the segments of values associated with the objects. Each call to the 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 address memory) in an object system (segment of values of the whole type). Thus the whole of the segments of values associated with an identical mark is easily identifiable and their location requires only one simple sweeping of a vector of entireties. The loop on the segments of values is carried out in two times: one first of all treats overall the collections, then the simple objects 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 the routine JJECRS, if need be, the system object 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 who 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 at the time of the retassage of the files or in debug mode 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: the routine JJLIDE check that the mark associated with the segment with values is identical to the current mark stored in the commun run /IADMJE/, 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 the contents of the attributes modify address memory and address 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 memory JEVEUX manage report unloadings on disc, to release from the place 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 the utilities OPENDR, WRITDR, READDR and CLOSDR who are called (bibc/utilitai/iodr.c).

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

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

The number of recordings 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 (concept of “extend”), this value is affected through the function ENVIMA LOFIEM and stored in the commun run /FENVJE/. JEVEUX manage 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 recordings, 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.

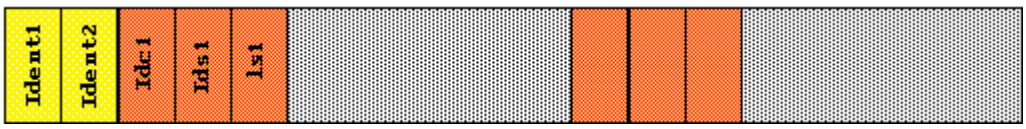
The size of the recordings defines two classes of objects JEVEUX :

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

At the time of 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 recording before their transfer. At the time of a request of reading, at least it is a recording 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 who call the utility CLOSDR.

Description of the recordings



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

- if Ident1=Ident2=0 the recording contains small objects, three words whole (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, follows the segment of values, and one starts again for the following until Idcn=Idsn=0 ;
- if Ident1 or Ident2 is different from 0, the recording contains whole or part of the segment of values associated with a large object.

At the time of the destruction of a large object identifiers Ident1 and Ident2 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 identifiers Idci and Idsi are affected sign -.

Writing of the objects

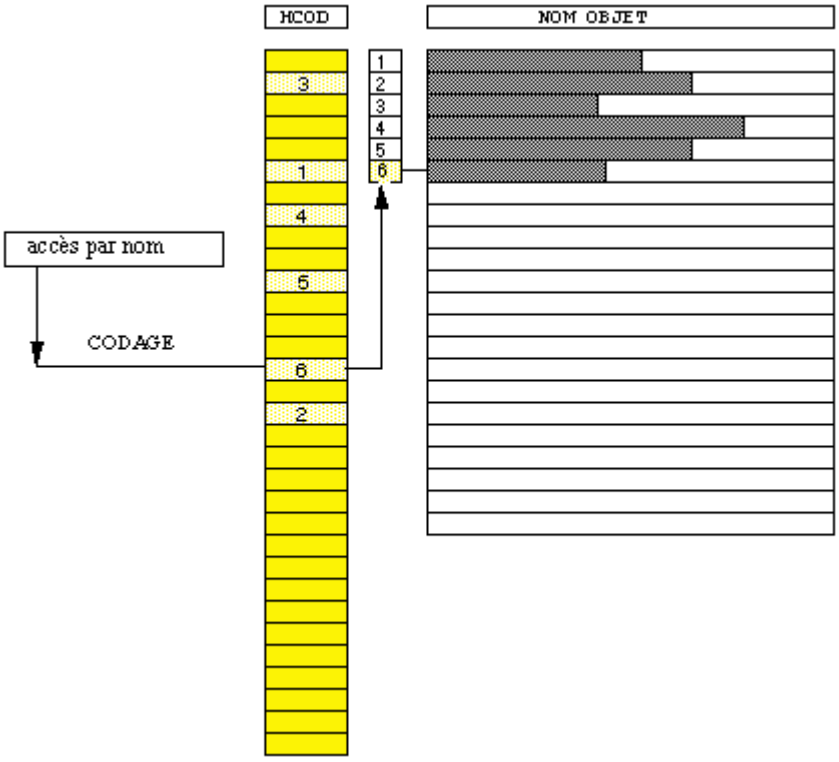
It is the routine JXECRO who treats the writing of the objects JEVEUX. It also ensures, when it is necessary, the opening of the logical units associated with the partition in extend of the bases. The routine examines the various recordings to find a succession of recordings being able to accommodate the segment of values or the plug following the cases. The recordings 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 the 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 the routine JXECRB. JXECRB is a hat which calls on the utility WRITDR who brings up to date the descriptors Ident1 and Ident2 as well as a meter associated with the recording. During later unloadings, the plug of reading can be used to bring up to date the image disc; a logic indicates this kind of use then.

Reading of the objects

It is the routine JXLIRO who treats the reading of the objects JEVEUX. 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 recording. The segment of values associated with the large objects is directly read again using the routine JXLIRB.

6 Access by names: associative addressing

Objects JEVEUX, simple 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 associative addressing 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 objects JEVEUX 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 entireties and a vector of character strings for the management of different the classes, or an object of repertoire kind of names 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 repertoires requires particular functions of access. The dimensioning of these objects is carried out in order to contain the number of necessary identifiers by minimizing their size and the collisions on the level of the function of coding. Our choice to dimension the pointer of entireties associated with the result with the function with addressing stopped on the following condition:

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

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

The calculation of the size of the repertoires is carried out 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 repertoires of names (and that of the pointer of entireties to approximately 795,000).

The selected function of coding JXHCOD fact call to the function system STRMOV who allows to transfer byte by byte a character string in a table from entireties and to the function XOR to cumulate the results in an entirety (INTEGER). The identifier is finally obtained by a congruence modulo the length of the repertoire $nrep$.

Insertion of a name

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

If the repertoire of names starts to be saturated, the function of coding, although selected to be dispersive, can give an identical identifier for two distinct chains, there is then collision. A new identifier must be given by taking account of the value obtained previously.

Scan for noun in a repertoire

This algorithm requires a certain number of comparisons of character strings and can thus become expensive. A commun run, brought up to date by last research in the repertoires of the various bases contains the identifier and the associated chain in order to reduce the cost of research (/IATCJE/). It is the routine JJVERN who carries out a comparison with the contents of the commun run before the call to JJCREN. The code return of the routine JJCREN depends on the type of research in the repertoire: with insertion in the repertoire of names (ICRE=1) this code return is obligatorily nonnull (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 chain between the positions 25 to 32 which indicates that one treats an object of collection.

The characters composing the names of the objects are limited to the alphanumerics supplemented by the special characters:

' '	the white,	'.'	the point,
'_'	the underlined white,	'$'	the symbol dollar,
'&'	and commercial one.		

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

Destruction of a name

Does the destruction of a name use the same algorithm as insertion, the position in the repertoire 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 repertoires

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

Case of the repertoires of collection

The repertoires of collection are objects of nonhomogeneous contents: they store at the same time the 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 the whole of the whole codes,
IDENO	the address from which the names are stored,
ILNOM	the length of the stored chains,
ILMAX	the maximum dimension of the repertoire,
ILUTI	the number of actually stored names,
IDEHC	the address from which the whole codes are stored.

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

Objects of named collection, during their creation by the routine JECROC, are inserted using the function JJCODN. The 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 segments of values not referred

The manager of memory JEVEUX use part of the memory to manage the attributes associated with the objects and to treat certain functions. In order not to multiply the routines of access we chose to use the same structures for the objects JEVEUX and for the memory used for their management. This is why during the impression of the segmentation memory one sees appearing segments of values associated with illicit names with the direction user and referring to the various classes open to a given moment, but also of the segments of values which are associated with no name. The system objects associated with the base TOTAL all carry the following prefix: _____TOTAL_____(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 at the time of the first call to the routine JEINIF. JEVEUX needing permanently to reach the associated segments of values, a specific mark (- 2) their is affected. A particular treatment their is reserved during the closing of the bases.

List of the system objects used by JEVEUX :

	Suffix of the name system object	Contents	Type FORTRAN associated (64 bits)	Size
1	$$CARA	characteristics of the associated base	INTEGER*8	11
2	$$IADD	addresses disc of the objects	INTEGER*8	2*NREMAX
3	$$GENR	kind of the objects (E, V, NR or X)	CHARACTER*1	NREMAX
4	$$TYPE	type of the objects (I, R, C, L, K)	CHARACTER*1	NREMAX
5	$$DOCU	documentary field	CHARACTER*4	NREMAX
6	$$ORIG	documentary field	CHARACTER*8	NREMAX
7	$$RNOM	list of the names of objects	CHARACTER*32	NREMAX
8	$$LTYP	types of the segments of values	INTEGER*8	NREMAX
9	$$LONG	length measured in the type of the segments of values	INTEGER*8	NREMAX
10	$$LONO	effective length measured in the type of the segments of values	INTEGER*8	NREMAX

11	$$DATE	date of first safeguard	INTEGER*8	NREMAX
12	$$LUTI	length used of the segments values	INTEGER*8	NREMAX
13	$$HCOD	table of associative addressing	INTEGER*8	NRHCOD
14	$$USADI	description of the contents of the recordings	INTEGER*8	2*NBLMAX
15	$$ACCE	number of access in read/write to the recordings	INTEGER*8	NBLMAX
16	$$MARQ	marks associated with the objects	INTEGER*8	2*NREMAX
17	$$INDX	index of the file of associated direct access	INTEGER*8	2*NBLMAX
18	$$TLEC	plug of reading	INTEGER*8	LONGBL
19	$$TECR	plug of writing	INTEGER*8	LONGBL
20	$$IADM	addresses memory of the objects	INTEGER*8	NREMAX

where

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

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

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.

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

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

8 Collections

Collections of objects JEVEUX 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 the simple objects 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 entireties containing the identifiers of the various objects composing the collection (inter alia the system objects of the collection which contain a suffix starting with $$). This vector bears the name allotted using the routine JECREC (CHARACTER*24). The system objects specific to the collection carry a suffix starting with $$ in position 25, if they are associated with a divided object, they carry a suffix starting with &&. The attributes common to the whole of the objects of collection are deposited among the attributes of the system object $$DESO (kind, type, length, etc).

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

	Suffix of the name of the system object of collection	Contents	Associated type FORTRAN	Type of collection
1	$$DESO the attributes associated with this object are the common attributes of the collection	contiguous collection: values associated with the various objects with collection	INTEGER*8	dispersed and contiguous the segment of values exists only for the contiguous collections
2	$$IADD	addresses disc of the objects of dispersed collections	INTEGER*8	dispersed
3	$$IADM	addresses memory of the objects of dispersed collections	INTEGER*8	dispersed
4	$$MARQ	marks associated with the objects with dispersed collections	INTEGER*8	dispersed
5	$$NOM or divided object	list of the names of objects of named collections	according to the associated repertoire	named
6	$$LONG or divided object	length measured in the type of the segments of values; receives the values associated with the attribus LONMAX and NOMMAX	INTEGER*8	of variable length
7	$$LONO or &&LONO	effective length measured in the type of the segments of values; is used in-house by the software	INTEGER*8	of variable length
8	$$LUTI or &&LUTI	length used of the segments of values	INTEGER*8	of variable length
9	$$NUM	information concerning the 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 zone memory to obtain the identifiers of the system objects (when they exist) in the following order: $$DESO, $$IADD, $$IADM, $$MARQ, $$NOM, $$LONG, $$LONO, $$LUTI and $$NUM. The maximum number of objects of collection is stored with the address IVNMAX.

Named collections

The objects associated with such a collection are accessible by their name (function JEXNOM) and by their sequence number of insertion (function JEXNUM). It is possible to use the routines JENUNO and JENONU to pass from the number to the name and conversely. The length of the names of the objects is limited to 8 characters (CHARACTER*8) if the collection is based on an “internal” repertoire of name, or can to be worth 8.16 or 24 if the collection is based on an “external” repertoire of name, i.e. created beforehand (repertoire of names divided).

Numbered collections

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

Dispersed collections

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

Contiguous collections

There exists only one segment of values for the whole of 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 system object $$DESO.

Collections variable length

Each object must be dimensioned: by affecting the attribute length by the 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 an object divided) 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 to reach directly the vector cumulated lengths (LONCUM) by using the 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 system object $$LONO is incremented of 1 compared to the length of the system object $$LONG for this purpose.
This mechanism was extended to the collections dispersed to reach the 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.

Collections fixed length

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 the attribute LONT of 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, in the same manner that it is necessary to have access to the system objects associated with a class at the time of 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. The 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 associated character string in the name of the object JEVEUX 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 Continuations

The system object of suffix $$CARA (containing the name of the base associated in position 9 to 24), contains the necessary information with 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 the unit 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 recording. In the event of continuation on a basis, the first action carried out will be the second reading of the contents of this object. The routine JXLIR1 open the file associated with the first “extend” (glob.1) with characteristics which is clean for him (what can lead to a message of alarm), 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 an image disc are created and initialized (addresses memory, plug of input/output,…).

10 Treatment of the objects JEVEUX

Creation of the objects

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

CLAS	class of fastening of the object to a database.	V : base Volatile, G : base Total, L : base Local, C : base Catalogues compiled
GENR	kind of the object	E : simple variable, V : vector, NR : repertoire of names
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, NAKED NO can be followed by the name of

	if numbered	the repertoire of names
STORAGE	CONTIG or DISPERSE	
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 the 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 NR
LONUTI	length used of the object of kind V
NOMUTI	length used of the object of kind NR
DOCU	documentary field (4 characters)

Reading of the attributes

The values associated with the various attributes can be consulted constantly using the routine JELIRA, including in-house managed attributes:

- accessible internal attributes:

DATE	date of last disc unloading of the object
ORIG	not used
IADM	address memory
IADD	address disc
LONO	length measured in the type of the segments of values and according to their kind
USE	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 while using USE for value of the argument of the name of attribute. The value XOUS for this same argument allows 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

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

- treatment 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 the routine JJALLC, then using the routine JJCROC, determination of the identifier of object of named collection or checking of the sequence number of numbered collection,
- according to the type, call to JJALTY to obtain the address compared to the table Z* 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 the routine JXVEUO,
- in certain cases (for example consultations of attributes), release of the object and/or collection by JJLIDE.

Attributes necessary to the description of the object JEVEUX are read again or determined by the routine JXVEUO, the treatment 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).

The alternative JEVEUS allocate in a permanent way the segment of values in memory.

Destruction of the objects

Destruction of an object JEVEUX (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 an image disc, in this case it is also necessary to destroy the latter, the corresponding recordings will be marked free and could be recovered later. The segment of values in memory will be marked free. The image disc, if it exists will be marked free while using, following the type of object, descriptors of the recording (system object $$USADI) or descriptors within a recording (assignment of the sign -). This function is provided by the 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 treatment 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 the whole of the objects of collection and the system objects of the collection provided that they are not divided. Routines JEDETR and JEDETC allow to destroy objects JEVEUX, the first works starting from an identifier, the second, more expensive, first of all carries out a search for descriptor in the repertoires of the classes open starting from a character string to a given position. The routine JEDETV is only used to destroy the objects on the volatile basis associated with the class V between the various orders of *Code_Aster*.

Release clarifies objects

Although the mechanism of release is implemented with the concept of mark and the calls obligatory to the routines JEMARQ and JEDEMA, certain configurations require an explicit call to the following routines of release:

JELIBE release the object requested by respecting the affected mark,
JELIBS release the object of name passed in argument when the associated mark is worth -3,
JELIBZ release the whole of the objects associated with a class with which the mark associated is worth -1.

Recopy of the objects

The utility JEDUPO allows to duplicate an object JEVEUX (simple object, or collection supplements) possibly by depositing the 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 being pressed 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 the result of the recopy on another class. The receptacle can preexist (the user provides a name or a chain), 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 who works starting from a under-chain of characters but requires on the other hand a preliminary research of the names in the repertoire (what can prove to be expensive).

Impression of the contents of the segments of values

The utility JEIMPO are charged to print in a pleasant way the contents of (of) the segment (S) of value (S) associated (S) with the objects JEVEUX. The objects system (associates with a class or a collection) are treated by the 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 who carries out the formatting of the data.

11 Treatment of the bases

Certain operations treat in their entirety the bases JEVEUX, they are essential to initiate the management system 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 order Aster CONTINUATION, and it is essential to finish the execution by the order END to close the files of direct access properly. Only a stop with a message UTMESS type <S> allows the SUPERVISOR to validate the concepts created and to properly close the files of direct accesses by call to the routine JEFINI

The opening of a base

The length of the recordings of the file of direct access and the initial length of the repertoire of names remain the only adjustable parameters associated with the bases JEVEUX. They are specified at the time of the call to the 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 allows to avoid superfluous inputs/outputs if the base is not preserved. The reopening of a base (order CONTINUATION in *Code_Aster* or reading of the catalogue of the compiled elements) requires the knowledge length of the recordings of the file of direct access and of the contents of certain system objects, the first recording contains the data essential to the reconstitution of this various information. The routine JXLIR1 is charged to read again the beginning of the first recording: one opens file of direct access (with an index whose size is fixed at 11), one reads information at the beginning of recording, 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 the routine JELIBF, consists in releasing the whole of 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 recording. The file of direct access is then closed by call to the routine JXFERM.

The retassage of a base

At the time of the operations of destruction of object JEVEUX, 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 recordings. It is thus necessary to modify the attribute addresses disc of the objects contained in the recordings 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 recording to be moved!). There is no reorganization within the recordings containing the images of small objects. The routine is used JETASS and one calls on the alternative “JETASS“routine of release JJLIDE. This utility can be directly called by the order END in *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 not being able to be reduced in place. The routine JXCOPY work starting from closed bases and restores them in the same state. This utility can be called by the order END in *Code_Aster*.

12 Impressions

The routine JEIMPD allows to print the list of the objects JEVEUX present on one or more bases. The list is made up starting from the associated catalogue (system object $$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 recording containing the segment of values,
- the position in the recording for the small objects,
- the number of access in reading on the recording,
- the number of access in writing on the recording.

```
--------------------------------------------------------------------------------

       CONTAINED BASE G

  NAME OF THE BASE               : TOTAL
  MAXIMUM NB OF RECORDINGS: 5242
  RECORD LENGTH (BYTES): 819200

 --------------------------------------------------------------------------------

---- NUM ------------- NAME ---------------- G T - L - LOTY- - IADD- --LIADD- NB AC

       1 ________GLOBALE ________$$CARA    - VI  8     11      1       24     0
       2 ________GLOBALE ________$$IADD    - VI  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    - VI  8   2000      1   124280     0
       9 ________GLOBALE ________$$LONG    - VI  8   2000      1   140304     0
      10 ________GLOBALE ________$$LONO    - VI  8   2000      1   156328     0
      11 ________GLOBALE ________$$DATE    - VI  8   2000      1   172352     0
      12 ________GLOBALE ________$$LUTI    - VI  8   2000      1   188376     0
      13 ________GLOBALE ________$$HCOD    - Ni  8   4177      1   204400     0
      14 ________GLOBALE ________$$USADI   - VI  8  10484      1   237840     0
      15 ________GLOBALE ________$$ACCE    - VI  8   5242      1   321736     0
…
```

Impression of the memory

The routine JEIMPM allows to print the list of the objects JEVEUX 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 with the segment with values,
- the address relative memory 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- ------------- NAME --------------

|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| MA1     .GROUPENO              1
|G|  135|        2|    0|            106203168|X|           1| D| MA1     .GROUPENO              2
|G|  135|        3|    0|             74182160|X|           1| D| MA1     .GROUPENO              3
|G|  135|        4|    0|             79595680|X|           1| D| MA1     .GROUPENO              4
…
```

Impression of the repertoires system

The routine JEIMPR allows to print the repertoires associated with the various opened bases, it 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 recording containing the image disc,
- the address relative memory of the segment of values
- the address of the dynamically allocated pointer.

```
--------------------------------------------------------------------------------
------     CATALOGUE CLASS G
------
--------------------------------------------------------------------------------

--- NUM  -------------- NAME ---------------- G H T --LENGTH--- - LOTY- - IADD- --------KADM------- --------KDYN-------

       1 ________GLOBALE ________$$CARA    -V-I- 8       11      11      1            1090758           69888784
       2 ________GLOBALE ________$$IADD    -V-I- 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    -V-I- 8     2000    2000      1            5924306          108557168
       9 ________GLOBALE ________$$LONG    -V-I- 8     2000    2000      1            5926316          108573248
      10 ________GLOBALE ________$$LONO    -V-I- 8     2000    2000      1            5928326          108589328
```

```
     11  ________GLOBALE ________$$DATE    -V-I- 8     2000    2000      1            5930336           108605408
     12  ________GLOBALE ________$$LUTI    -V-I- 8     2000    2000      1            5932346           108621488
     13  ________GLOBALE ________$$HCOD    -V-I- 8     3203    3203      1            5934356           108637568
     14  ________GLOBALE ________$$USADI   -V-I- 8   188742  188742      2       5864054372198       46912496140304
     15  ________GLOBALE ________$$ACCE    -V-I- 8    62914   62914      1            5638518           106270864
     16  ________GLOBALE ________$$MARQ    -V-I- 8     4000    4000      0            5565328           105685344
     17  ________GLOBALE ________$$INDI    -V-I- 8     2000    2000      0            5569338           105717424
     18  ________GLOBALE ________$$TLEC    -V-I- 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
```

...

Impression of the attributes

The routine JEIMPA print the whole of the attributes for an object JEVEUX.

```
 WRITING OF THE ATTRIBUTES OF "MA1     .DIME           "

JEIMPA  IMPRESSION OF THE ATTRIBUTES OF >MA1     .DIME           <

CLAS    G
GENR    V
TYPE    I
LTYP              4
DOCU
DATE              0
LONMAX            6
LONUTI            6
LONO              6
IADM        20357178
IADD              0
LADD              0
USE   X D

  WRITING OF THE ATTRIBUTES OF "MA1     .CONNEX         "

JEIMPA  IMPRESSION OF THE ATTRIBUTES OF >MA1     .CONNEX         <

ACCESS   NAKED
STOCKAGECONTIG
MODELONGVARIABLE
NMAXOC          204
NUTIOC            0
LONT           1472
CLAS    G
GENR    V
TYPE    I
LTYP              4
DOCU
DATE     -1292845870
LONO           1472
IADM        20270746
IADD              0
LADD              0
USE   U D
```

Note:

The impression 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.

Impression of the contents of a segment of values

The routine JEIMPO allows to print the segments of values associated with an object JEVEUX.

```
 SEGMENT IMPRESSION OF VALUES >MA1     .DIME                  <

      1 -          361            0          204            0            0
      6 -            3
```

```
 SEGMENT IMPRESSION OF VALUES >MA1     .COORDO    .VALE        <

      1 -  0.00000D+00  5.00000D-01  1.00000D+00  5.00000D-01  5.00000D-01
      6 -  1.00000D+00  1.00000D+00  5.00000D-01  1.00000D+00  0.00000D+00
     11 -  7.50000D-01  1.00000D+00  5.00000D-01  7.50000D-01  1.00000D+00
     16 -  1.00000D+00  7.50000D-01  1.00000D+00  0.00000D+00  5.00000D-01
     21 -  2.00000D+00  5.00000D-01  5.00000D-01  2.00000D+00  1.00000D+00
     26 -  5.00000D-01  2.00000D+00  0.00000D+00  7.50000D-01  2.00000D+00
     31 -  5.00000D-01  7.50000D-01  2.00000D+00  1.00000D+00  7.50000D-01
     36 -  2.00000D+00  5.00000D-01  1.00000D+00  1.00000D+00  1.00000D+00
     41 -  1.00000D+00  1.00000D+00  5.00000D-01  1.00000D+00  2.00000D+00
     46 -  1.00000D+00  1.00000D+00  2.00000D+00  0.00000D+00  1.00000D+00
     51 -  1.00000D+00  0.00000D+00  1.00000D+00  2.00000D+00  0.00000D+00
     56 -  5.00000D-01  3.00000D+00  5.00000D-01  5.00000D-01  3.00000D+00
     61 -  1.00000D+00  5.00000D-01  3.00000D+00  0.00000D+00  7.50000D-01
     66 -  3.00000D+00  5.00000D-01  7.50000D-01  3.00000D+00  1.00000D+00
     71 -  7.50000D-01  3.00000D+00  5.00000D-01  1.00000D+00  3.00000D+00
     76 -  1.00000D+00  1.00000D+00  3.00000D+00  0.00000D+00  1.00000D+00
     81 -  3.00000D+00  0.00000D+00  0.00000D+00  1.00000D+00  5.00000D-01
```

…

13 APPENDIX 1: Description of the commun runs used in the manager of memory JEVEUX

- /FENVJE/

```
INTEGER          LFIC, MFIC
COMMON /FENVJE/  LFIC, MFIC
```

LFIC	length maximun in bytes of a extend,
MFIC	maximum size in bytes of disk space usable.

- /IACCED//JIACCE/

```
 PARAMETER  (NR = 5)
COMMON /IACCED/  IACCE (1)
COMMON /JIACCE/  JIACCE (NR)
```

IACCE, JIACCE	variable of reference and position of the segment of values associated with the system object with suffix: $$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	address of the segments of values containing the addresses of the marked objects,
LGD	length of the segment of value associated with KDESMA,
LGDUTI	length used of the segment of value associated with KDESMA,
KPOSMA	address of the segments of values containing the positions associated with each mark,
LGP	length of the segment of value associated with KPOSMA,
LGPUTI	length used of the segment of value associated with KPOSMA.

- /IATCJE/

```
INTEGER          ICLAS, ICLAOS, ICLACO, IDATOS, IDATCO,
IDATOC
COMMON /IATCJE/  ICLAS, ICLAOS, ICLACO, IDATOS, IDATCO,
IDATOC
```

ICLAS	current class,
ICLAOS	class of the simple object,
ICLACO	class of the collection,
IDATOS	identifier of the simple object,
IDATCO	identifier of the collection,
IDATOC	identifier of the object of collection.

- /IATRJE//JIATJE/

```
PARAMETER  (NR = 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 (NR), JLONG (NR), JDATE (NR), JIADD
(NR), JIADM (NR),
             +                 JLONO (NR), JHCOD (NR), JCARA (NR), JLUTI
(NR), JMARQ (NR)
```

variable of reference and position of the segment of values associated with the system object with suffix:

LTYPE, JLTYP	$$LTYP
LENGTH, JLONG	$$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 /IDATJE/  NUMATR
```

NUMATR identifier of the system object of collection $$LONO

- /IDATJE/

```
       PARAMETER  (NR = 5)
       INTEGER          NRHCOD    , NREMAX    , NREUTI
       COMMON /ICODJE/  NRHCOD (NR), NREMAX (NR), NREUTI (NR)
```

NRHCOD	size (in entirety) of the system object $$HCOD
NREMAX	size (in entirety) of the system object $$RNOM
NREUTI	length used of the system object $$RNOM

- /IENVJE/

```
       INTEGER          LBIS, LAWS, LOLS, RENTED, LOR8, LOC8
       COMMON /IENVJE/  LBIS, LAWS, LOLS, RENTED, LOR8, LOC8
```

LBIS	length out of bits of the entirety,
LAWS	length in bytes of the entirety,
LOLS	length in bytes of logic,
RENTED	length in bytes of the unit of addressing,
LOR8	length in bytes of reality,
LOC8	length in bytes of the complex.

- /IEXTJE/

```
PARAMETER  (NR = 5)
INTEGER          IDN    , IEXT    , NBENRG
COMMON /IEXTJE/  IDN (NR), IEXT (NR), NBENRG (NR)
```

IDN	is not used any more since the use of the named accesses.
IEXT	number of extends open
NBENRG	maximum number of recordings of a extend

- /IFICJE/

```
PARAMETER      (NR = 5)
INTEGER          NBLMAX    , NBLUTI    , LONGBL    ,
+                KITLEC    , KITECR    , KINDEF    , KIADM
,
+                IITLEC    , IITECR    , NITECR    , KMARQ
 COMMON /IFICJE/  NBLMAX (NR), NBLUTI (NR), LONGBL (NR),
+                KITLEC (NR), KITECR (NR), KINDEF (NR), KIADM
(NR),
+                IITLEC (NR), IITECR (NR), NITECR (NR), KMARQ
(NR)
```

For each base associated with index I ranging between 1 and NR

NBLMAX	maximun number of recordings,
NBLUTI	many recordings used,
LONGBL	length in bytes of the recordings,
KITLEC	address in K1ZON segment of values associated with the plug with reading,
KITECR	address in K1ZON segment of values associated with the plug with writing,
KINDEF	address in ISZON segment of values associated with the index used for the files with direct access,
KIADM	address in ISZON segment of values associated with the addresses memory,
IITLEC	address disc of the plug of reading (record number),
IITECR	address disc of the plug of writing (record number),
NITECR	size in bytes of the portion of the plug of writing used,
KMARQ	address in ISZON segment of values associated with the marks.

- /ILOCJE/

```
INTEGER           ILOC
COMMON /ILOCJE/   ILOC
```

ILOC pointer on the address of the beginning of the zone memory allocated by JXALLM.

- /INUMJE/

```
INTEGER           NUMEC
COMMON /INUMJE/   NUMEC
```

NUMEC number of the object of collection or number of insertion in a repertoire of names.

- /ISTAJE/

```
INTEGER           ISTAT
COMMON /ISTAJE/   ISTAT (4)
```

ISTAT code associated with the state and the statute with the segments with 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 zone managed memory,
IDXAXD	initial position in ISZON for research.

- /IZONJE/

```
INTEGER           LK1ZON, JK1ZON, LISZON, JISZON, ISZON (1)
COMMON /IZONJE/   LK1ZON, JK1ZON, LISZON, JISZON
EQUIVALENCE     (ISZON (1), K1ZON (1))
```

LK1ZON	length in character (CHARACTER*1) managed zone,
JK1ZON	address in K1ZON beginning of the zone,

LISZON	length in entirety (INTEGER*8 in addressing 64 bits, INTEGER*4 in addressing 32 bits) of the managed zone,
JISZON	address in ISZON beginning of the zone,
ISZON	zone memory in entirety (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 allows 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  (NR = 5)
COMMON /KUSADI/  IUSADI (1)
COMMON /JUSADI/  JUSADI (NR)
```

	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	-1 is worth,
JCLASS	is affected with the result of the function ICHAR on the licit characters, if not is worth ILLICI.

- /JENVJE/

```
INTEGER          MSLOIS
COMMON /JENVJE/  MSLOIS
```

MSLOIS	mask being worth the sum of the weights of LOIS-1 first entireties, intended to replace the operation modulo (LAWS) by function AND

- /JCONJE/

```
INTEGER          MSSTAT, LSSTAT
COMMON /JCONJE/  MSSTAT, LSSTAT
```

MSSTAT	is not used any more, mask being worth the sum of the weights of LSSTAT first entireties.
LSSTAT	(LBISEM - 4) where LBISEM is the length out of bit of the entirety, 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  (NR = 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 (NR), JTYPE (NR), JDOCU (NR), JORIG
(NR), JRNOM (NR)
```

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  (NR = 5)
       CHARACTER*8      NOMBAS
       COMMON /KBASJE/  NOMBAS (NR)
```

NOMBA name of the base (used for the error messages)

- /KFICJE/

```
       PARAMETER  (NR = 5)
       CHARACTER*2      DN2
       CHARACTER*5      CLASS
       CHARACTER*8                  NOMFIC    , KSTOUT    , KSTINI
       COMMON /KFICJE/  CLASS    , NOMFIC (NR), KSTOUT (NR), KSTINI
(NR),
      +                 DN2 (NR)
```

DN2	unutilised!
CLASS	allows to store the whole of the names of the open classes (first letter of the basic name)
NOMFIC	name of the open bases
KSTOUT	statute at exit of the bases ('SAVES' or 'DESTROYED')
KSTINI	statute in continuation of the bases ('DUMMY','BEGINNING' or 'CONTINUES')

- /KNOMJE/

```
       CHARACTER*24     NOMEC
       COMMON /KNOMJE/  NOMEC
```

NOMEC name of object of collection or name to be inserted in a repertoire

- /KZONJE/

```
       CHARACTER*1      K1ZON
       COMMON /KZONJE/  K1ZON (8)
```

K1ZON zone memory 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	white chain length 32.

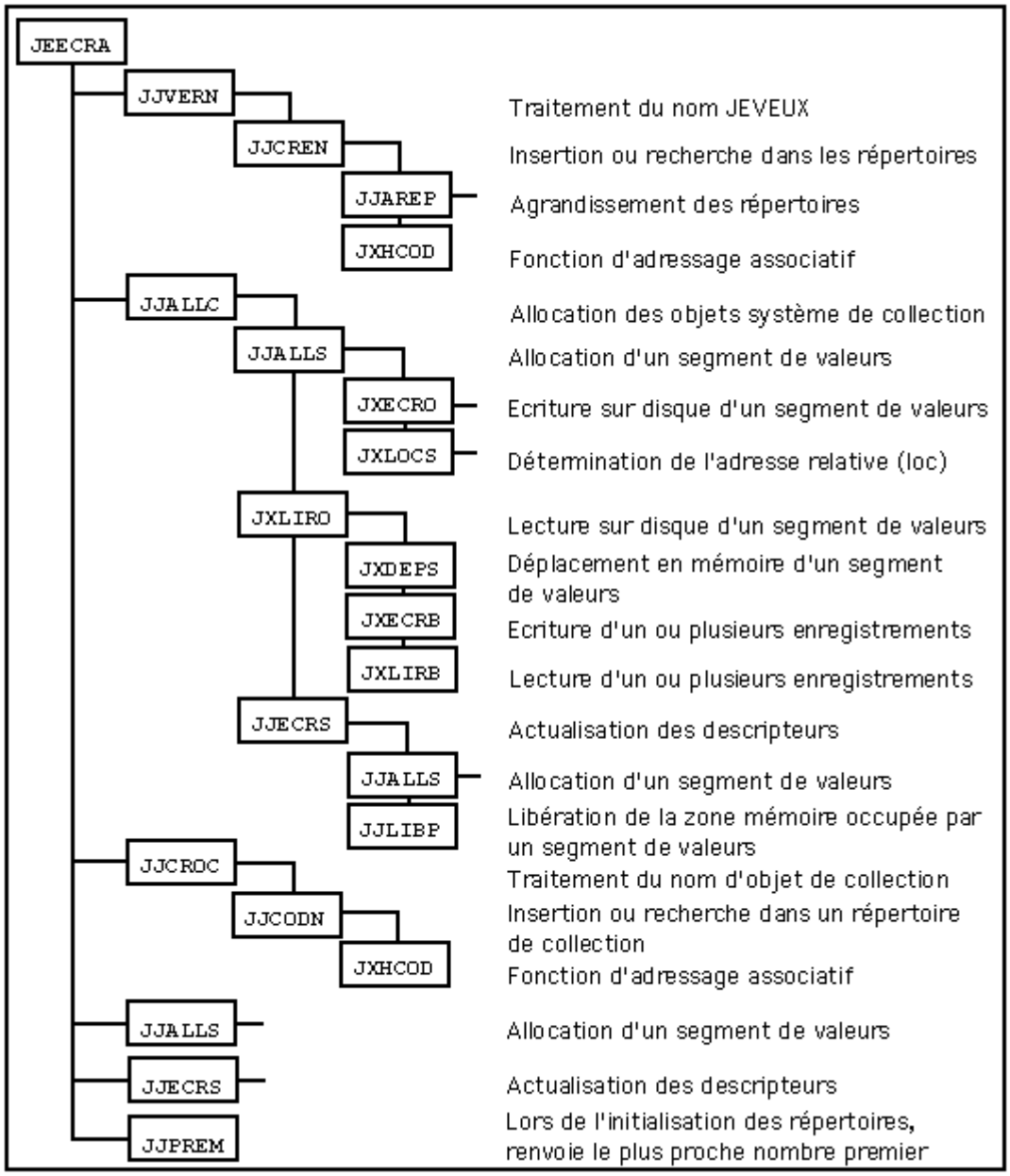
- /UNDFJE

```
INTEGER          LUNDEF, IDEBUG
COMMON /UNDFJE/  LUNDEF, IDEBUG
```

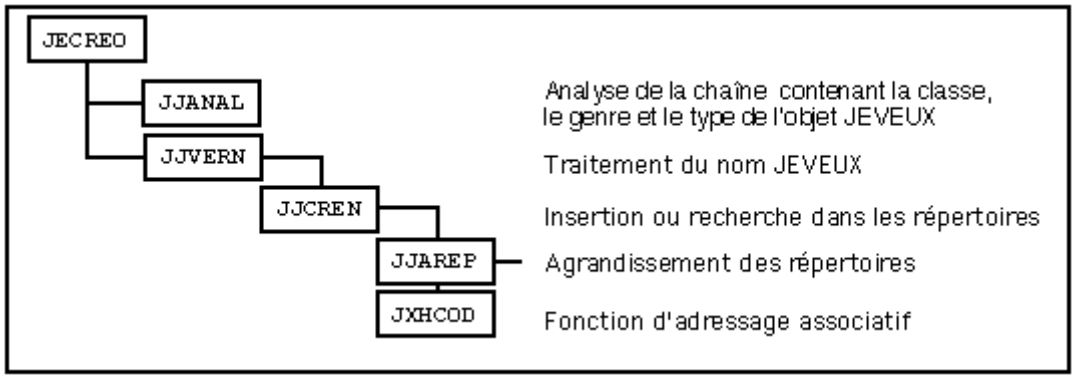
LUNDEF	affected in JEDEBU by the function envima ISNNEM (not has number),
IDEBUG	1 in debugg mode, 0 is worth if not.

14 APPENDIX 2: Trees of call simplifiés of some under - programs

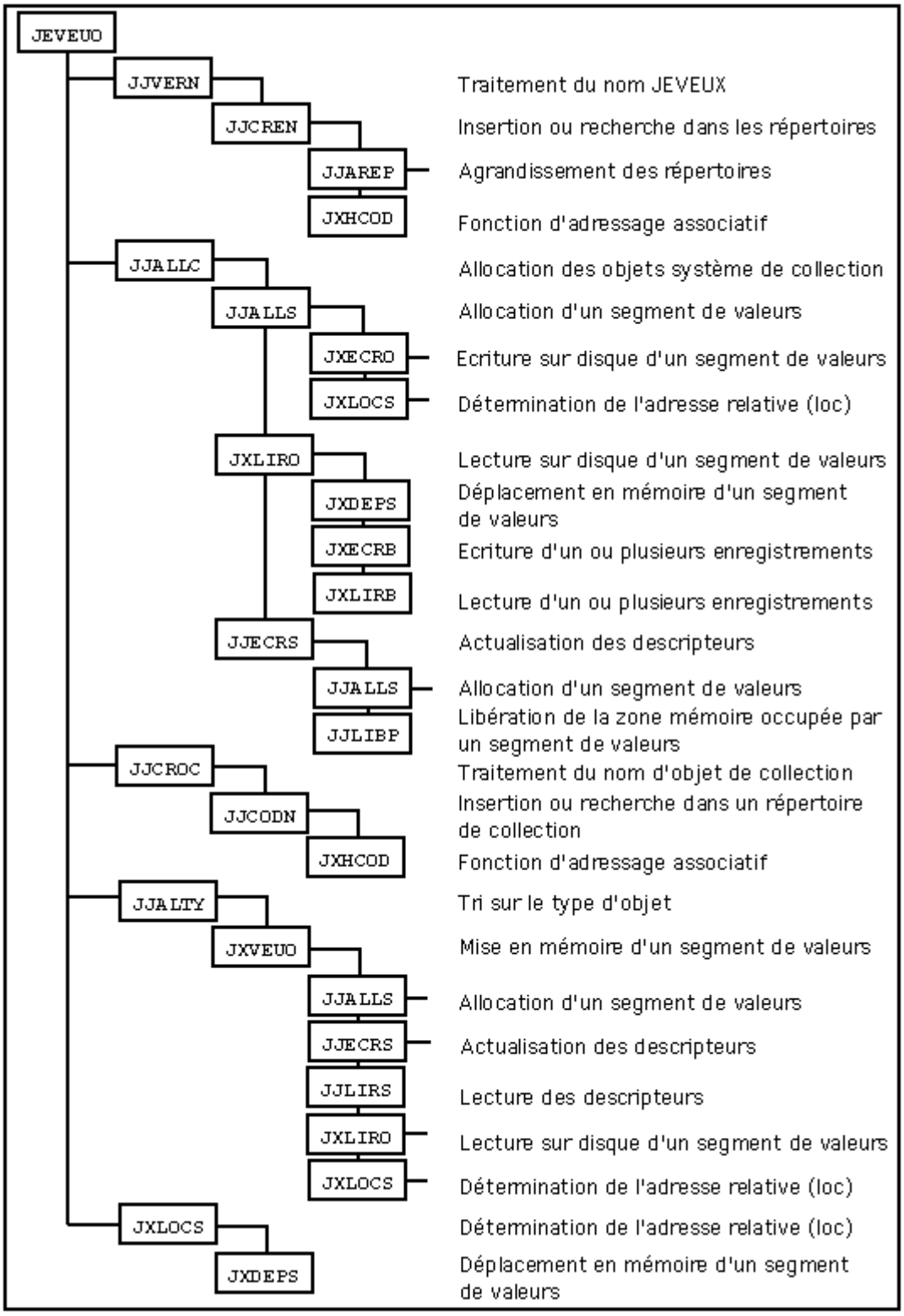
One presents below the trees of principal subroutines JEVEUX, one voluntarily limited to three levels of subroutines to facilitate comprehension. The truncated branches indicate that there exist other calls JEVEUX in the subroutine.



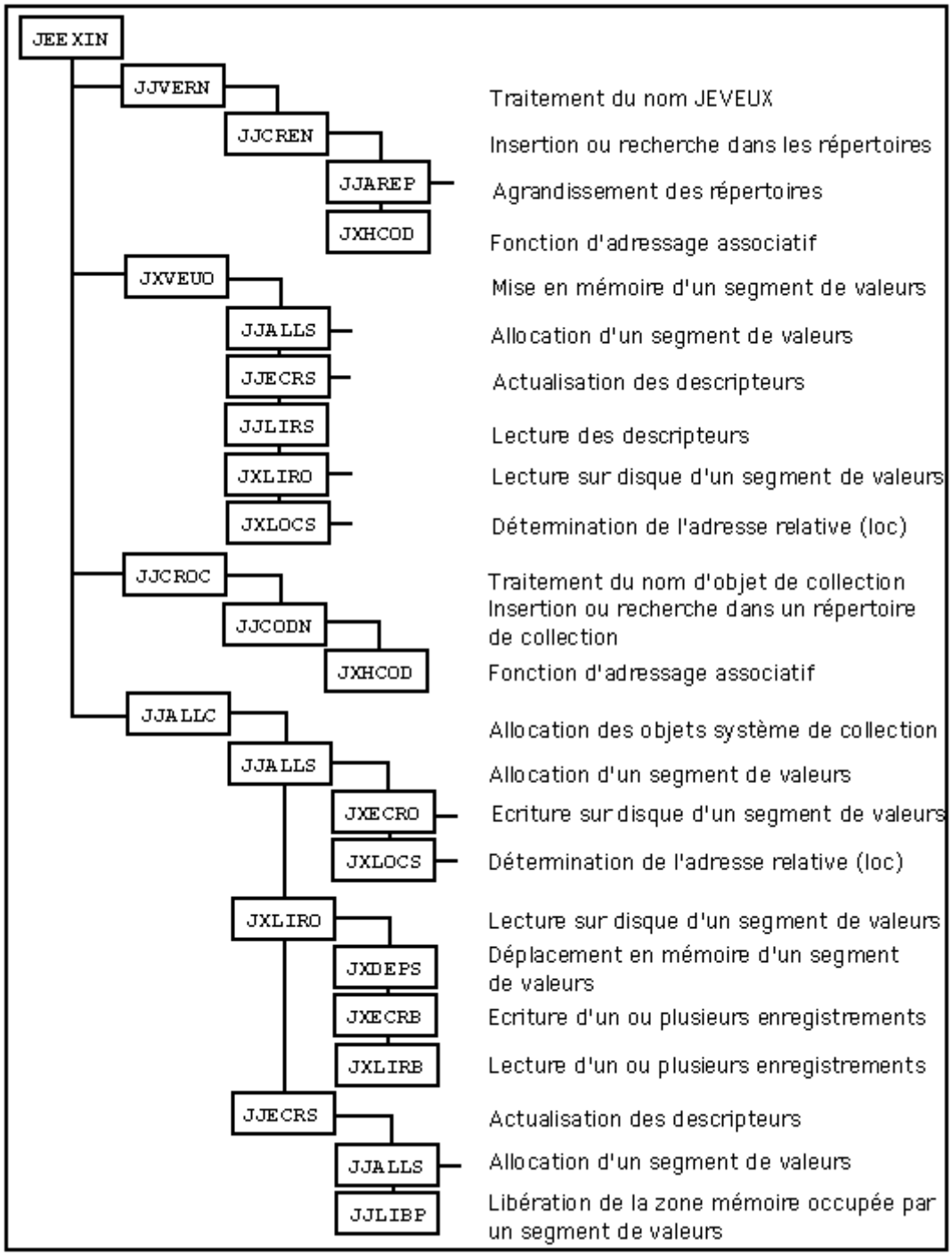
Tree of call of the routine JEECRA



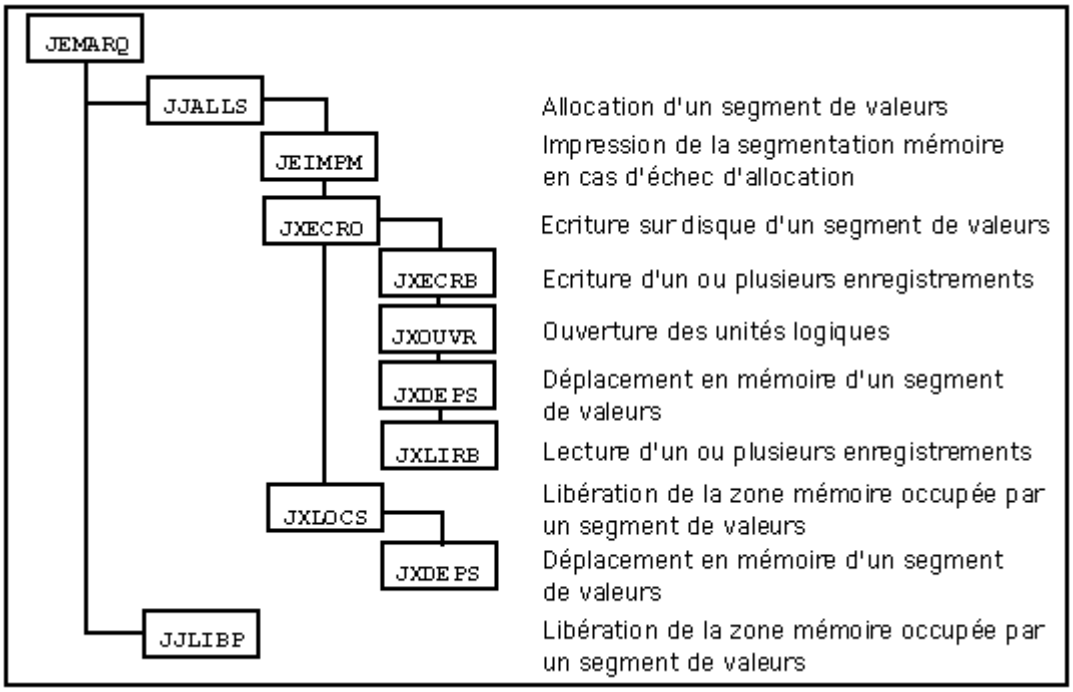
Tree of call of the routine JECREO



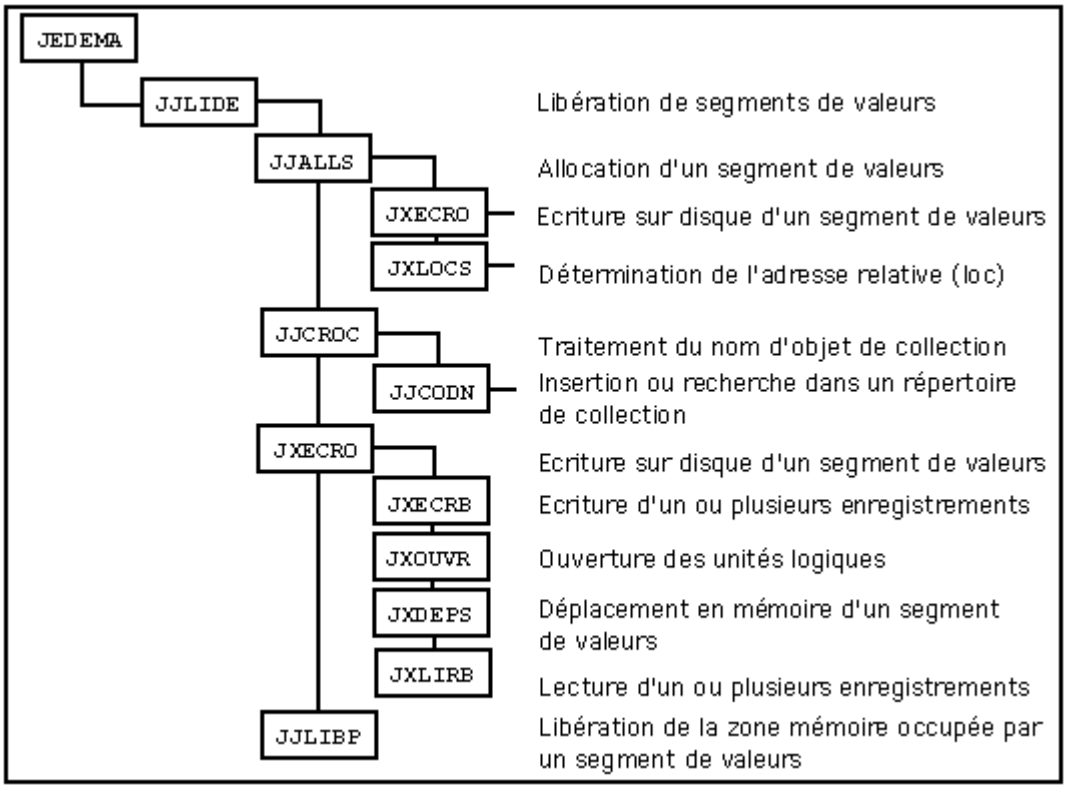
Tree of call of the routine JEVEUO



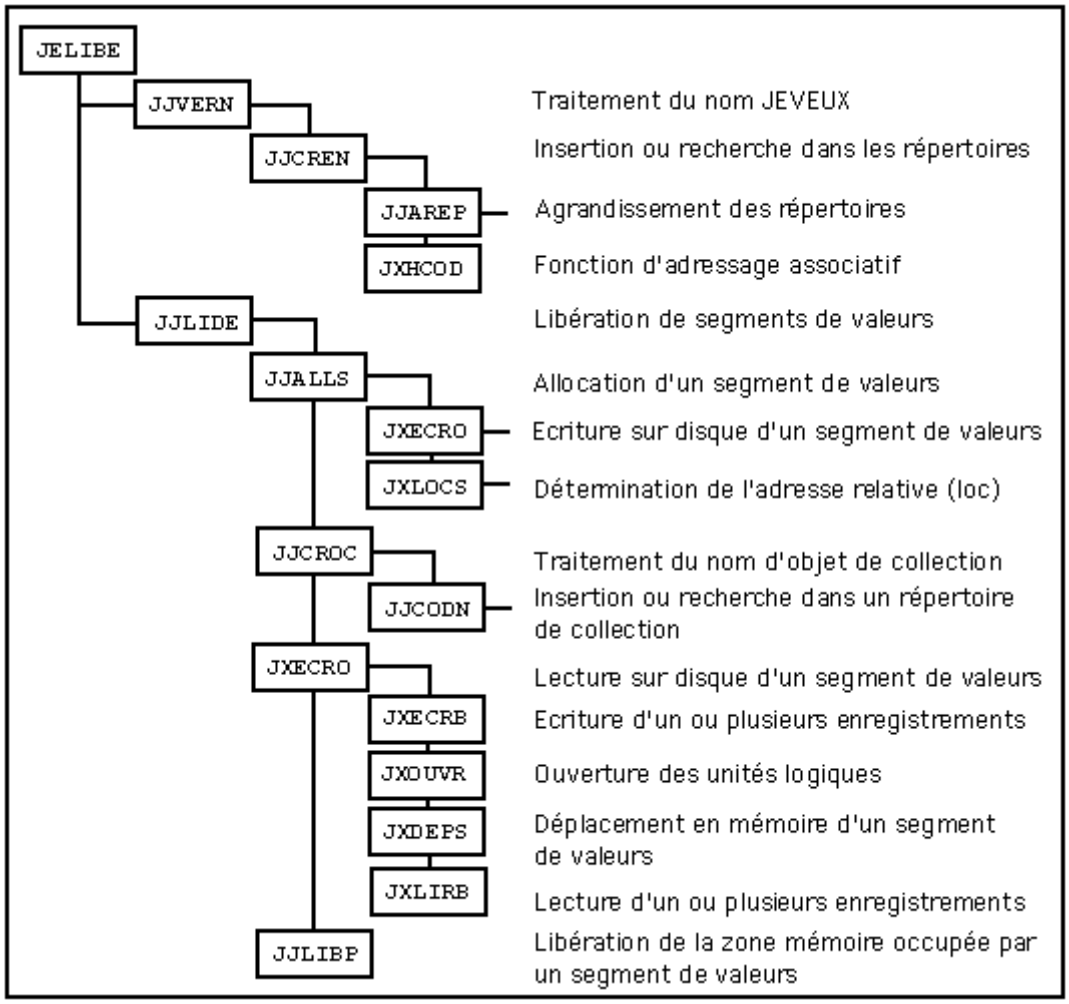
Tree of call of the routine JEEXIN



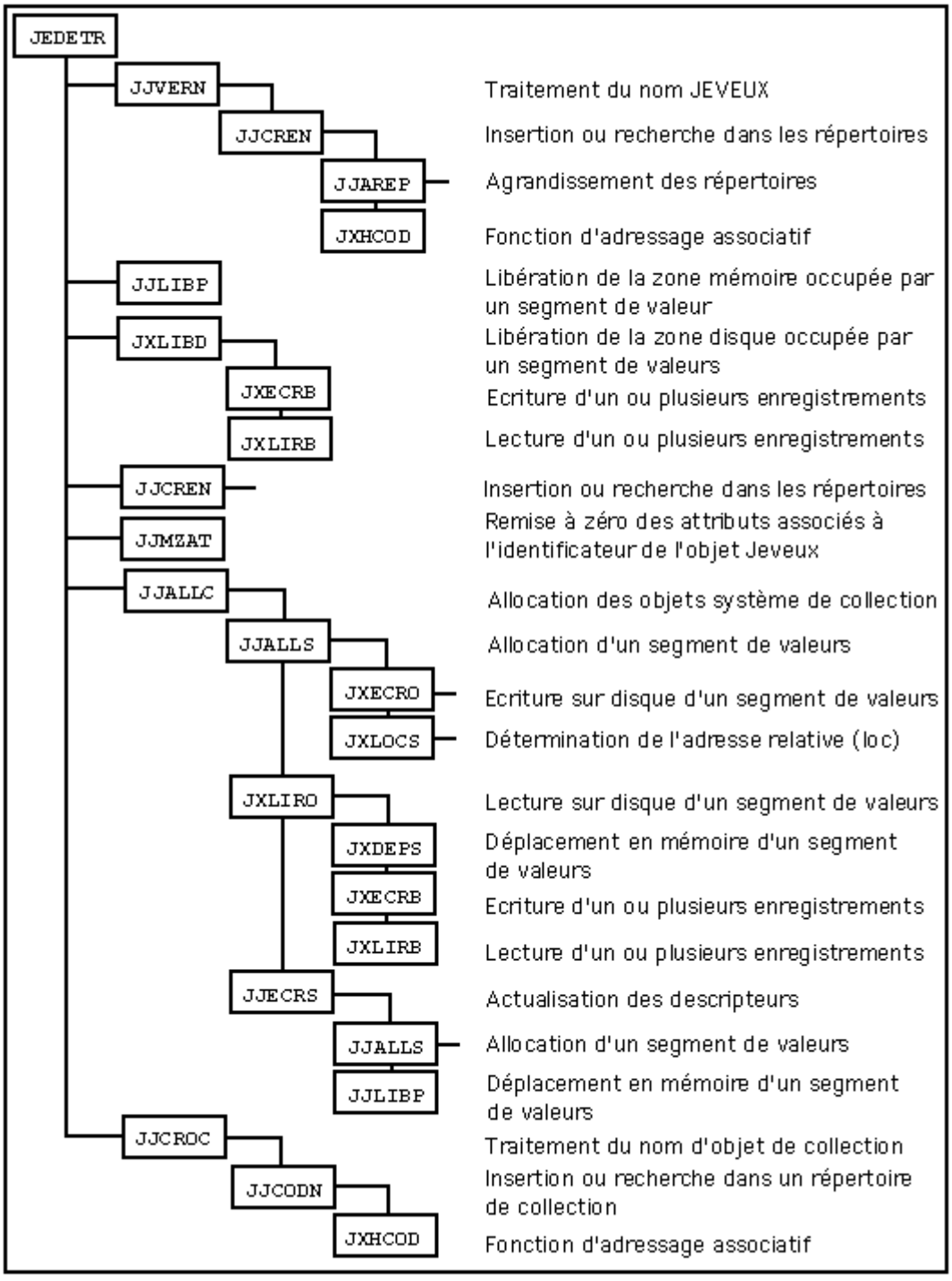
Tree of call of the routine JEMARQ



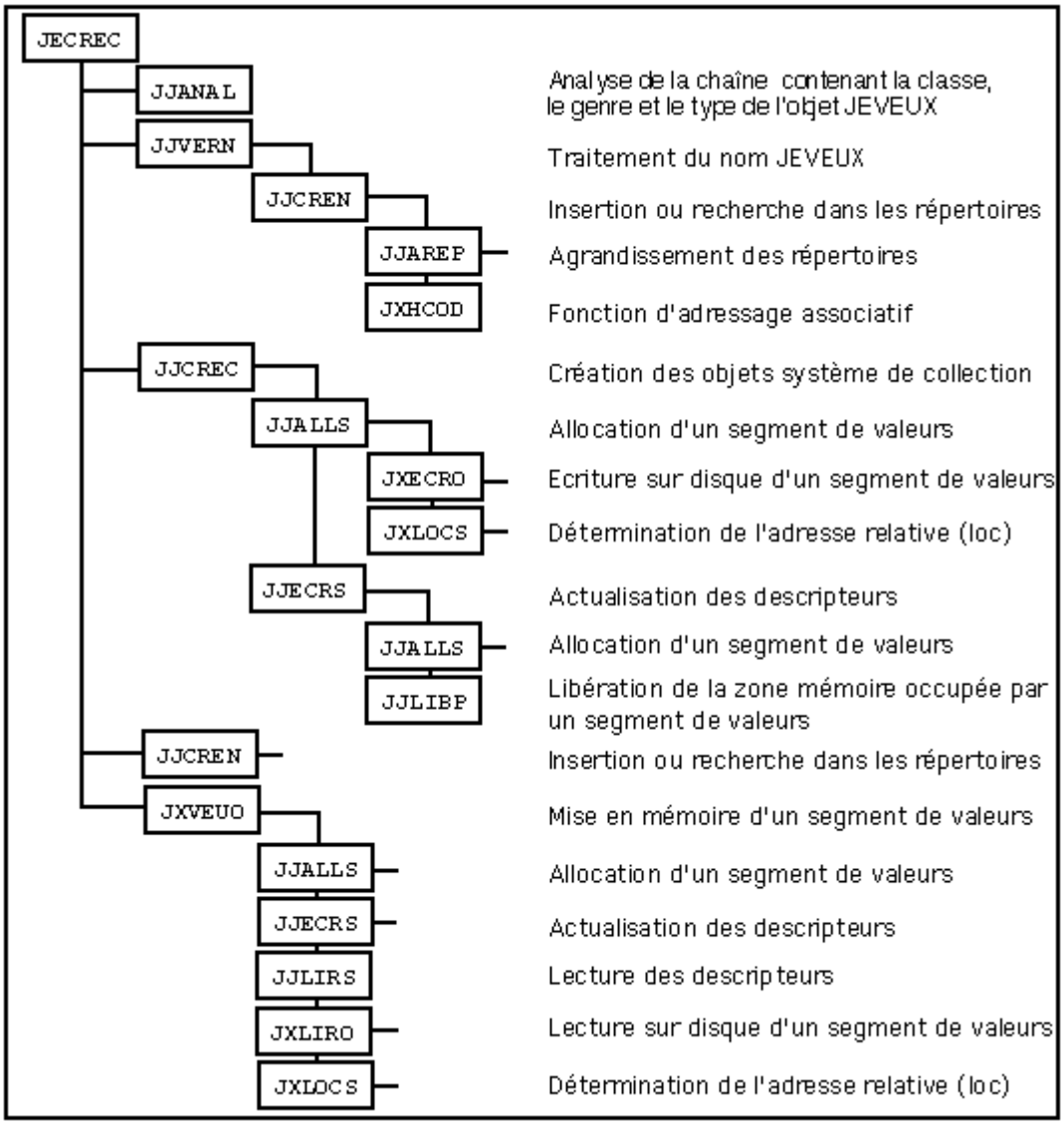
Tree of call of the routine JEDEMA



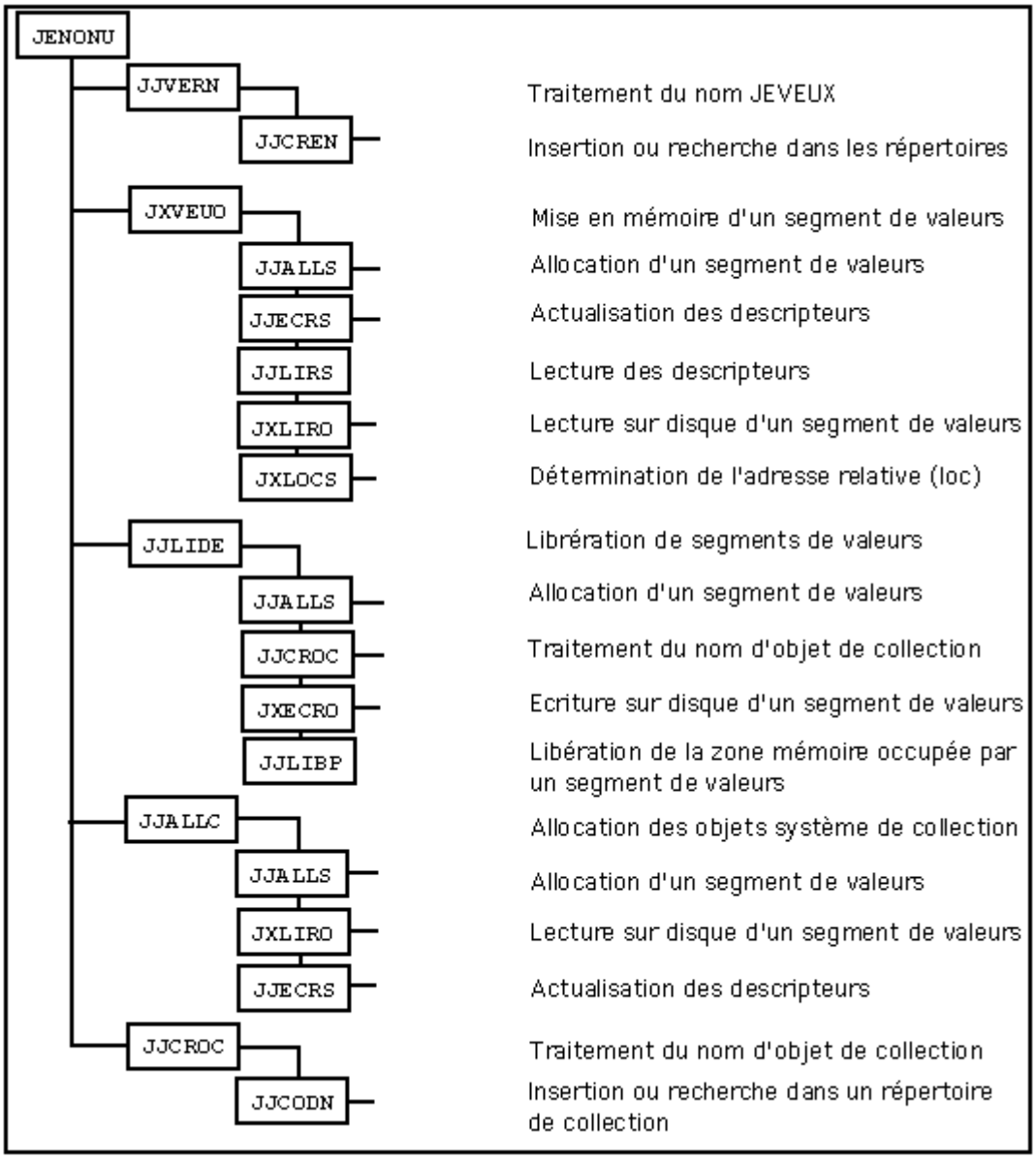
Tree of call of the routine JELIBE



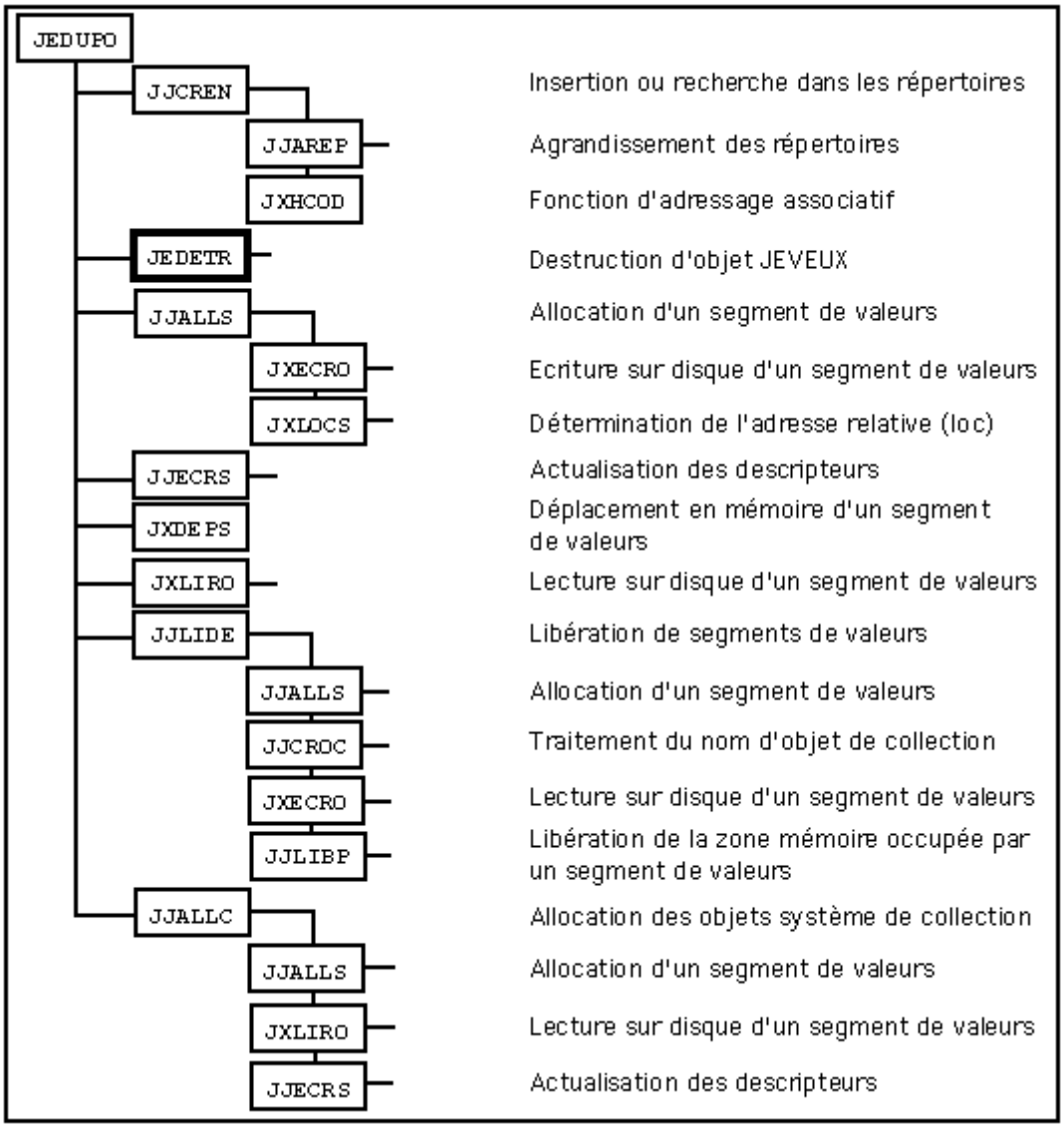
Tree of call of the routine JEDETR



Tree of call of the routine JECREC



Tree of call of the routine JENONU



Tree of call of the routine JEDUPO

15 APPENDIX 3: List of the subroutines and their principal functions

JECREC	Creation of a collection
JECREO	Creation of a simple object
JECROC	Creation of an object of collection
JEDEBU	Initialization of the parameters of the software
JEDEMA	Décrémente the mark and releases the objects
JEDETC	Destroyed a set of objects
JEDETR	Destroyed an object
JEDETV	Destroyed the objects of the Volatile base
JEDISP	Determine more big spaces available in memory
JEDUPC	Duplicate a set of objects
JEDUPO	Duplicate an object
JEECRA	Affect the attributes of an object
JEEXIN	Test the existence of a descriptor of object
JEFINI	Finish the execution of the software
JEIMPA	Print the attributes of an object
JEIMPD	Print the list of the objects present on a basis
JEIMPM	Print the contents of the segmentation memory
JEIMPO	Print the contents of the segment of value of an object
JEIMPR	Print the contents of a repertoire
JEINIF	Initialize the parameters associated with a base
JELIBE	Release an object
JELIBF	Release the whole of the objects associated with a base
JELIBZ	Release the whole of the objects associated with the mark -1
JELIRA	Reading of the value of an attribute of an object
JELSTC	Turn over the list of the objects containing a character string in their identifier
JEMARQ	Increment the current mark
JENONU	Determine the number of object of collection according to the name
JENUNO	Determine the name of object of collection according to the number
JEPRAT	Print the contents of the system objects
JERAZO	To 0 the contents of an object give
JETASS	Move the recordings within the file of direct access in order to recover free space
JEVEMA	Deliver the value of the current mark
JEVEUO	Return the position in table Z. segment of values associated with an object
JEVEUS	Return the position in table Z. segment of values associated with an object and positions the mark with -3
JEVEUT	Return the position in 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 allowing to reach by name an object of collection
JEXNUM	Function of synchronization allowing to reach by number an object of collection
JJALLC	Allowance of the system objects of collection
JJALLS	Allowance of a segment of values
JJALTY	Sorting on the type of object (before setting in memory of a segment of values)
JJANAL	Analysis of the chain containing the class, the kind and the type of the object
JJAREP	Enlarging of the repertoires
JJCODN	Insertion or research in a repertoire of collection
JJCREC	Creation of the system objects of collection
JJCREN	Insertion or research in the repertoires of the bases
JJCROC	Treatment of the name of object of collection

JJECRS	Actualization of the descriptors
JJIMPO	Impression of the contents of a segment of values
JJLIDE	Release of the segments of values
JJLIRS	Reading of the descriptors
JJMZAT	Restoring of the attributes associated with the identifier with a segment with values
JJPREM	At the time of the initialization of a repertoire, the nearest prime number returns (appearing in one dated)
JJVERN	Treatment of the name Jeveux
JXALLM	Dynamic allocation of the zone memory managed by the software
JXCOPY	Recopy of the file of direct access associated with a base with elimination with the recording marked unutilised
JXDEPS	Displacement in memory of a segment of values
JXECRB	Writing of one or more recordings
JXECRO	Writing on disc of a segment of values
JXFERM	Closing of a file of direct access
JXHCOD	Function of associative addressing
JXLIBD	Release of the zone disc occupied by a segment of values
JXLIBM	Final release of the zone dynamically allocated memory
JXLIR1	Reading of the first recording associated with a base in order to recover its characteristics
JXLIRB	Reading of one or more recordings
JXLIRO	Reading on disc of a segment of values
JXLOCS	Determination of the relative address of a segment of values
JXOUVR	Opening of the logical units associated with the bases
JXVERI	Examine the integrity of the segmentation memory
JXVEUO	Setting in memory of a segment of values

16 APPENDIX 4: GLOSSARY

Base	Together of files of direct access: the TOTAL base is made up by the files glob.1, glob.2, glob.3,…
Class	Named by the first letter of the base, the class allows to associate an object JEVEUX with a file.
Identifier of object JEVEUX	For a simple object or a collection it is the sequence number of insertion in the repertoire 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 entireties framing a segment of values in memory or one of the 3 entireties preceding a segment by values on a recording (disc or plug)
File of direct access	File from which the recordings are accessible directly by name or number
Object JEVEUX	Indicate at the same time the simple objects, the collections and the objects of collection
Simple object	Object from which the attributes are directly accessible among the system objects associated with the various classes
Objects of collection	Objects whose attributes are shared and managed in simple objects created with the collection
Segment of values	Together values associated with an object JEVEUX and positioned in a contiguous way in memory or on disc
System object	Simple objects managed by the software and intended to collect the values of the various attributes.
Debug Jeveux	Option of use of Jeveux allowing to cause the immediate unloading of the segments of values during the releases and

the assignment with value UNDEF or not of the released segment.