Application View of Save and Load
(Dump and Restore)

OPD/SPEC/D1

0 Contents

0.1 Changes in this issue

The section on CMOS RAM Protection has been removed.

1 Introduction

This paper presents a view of the Data Record Facility (Dump & Restore) from the point of view of the application programmer and the implementor of the Facility.

The user viewpoint is described in PSD 76.97.2.1, 'OPD Base Functional Software User Interface'.

2 Terminology

This section defines the terminology to be used in user-visible interfaces and documentation.

The name of the subsystem of the Base Functional Software which supports the facilities described is the Data Record Facility.

The action of writing a security copy of the store contents to a microdrive cartridge is called Saving.

The action of reading from a security copy on a microdrive cartridge into store, establishing new store contents is called Loading.

The use of the words save and (particularly) load is not unique within the OPD system and documentation to their specific meanings within the context of the data record facility.

The words dump and restore are used freely in this paper in place of save and load. This is for reasons of preference, history and that they are widely used in the industry.

3 Description

Dump & Restore is a security feature which is provided for the benefit of the user. It provides the means to make a copy on a microdrive cartridge of all permanent databases held in the store of the OPD at the time and the means to reinstate such saved databases from a microdrive cartridge to the OPD store.

What may be referred to in this paper as a 'permanent database' or just a 'database' is called a permanent segment in the Application Handler specification.

Alternatively, it provides the means to make a copy on a microdrive cartridge of the entire contents of the CMOS RAM and the means to reinstate those contents in the CMOS RAM from a microdrive cartridge.

Three types of dump are allowed. The full save dumps all permanent segments in the store. The selective save (which is available only as a procedural interface) allows the calling application to nominate which segments are to be dumped. It also performs a dual dump, the dump file is written twice to the same cartridge, to provide extra security. The CMOS RAM save dumps only the CMOS RAM entries.

A dump may be caused, or triggered, in one of three ways. The user may directly ask for one, using the Housekeeping application, or any application may ask for one at any time. It is also possible for the user to request that a dump takes place at a specified time every day. In at least two of these cases, the dump initiation comes as a surprise to applications which are running at the time. This has a considerable bearing on system design requirements. It is clearly not possible to take a secure dump of a database which is being changed at the same time by an application. It is equally dangerous for an application to continue processing on a database which is in the process of being destroyed and replaced by a different copy of the database, during a restore operation.

The guiding principles are therefore that during a dump operation all applications must cease updating permanent databases and that during a restore operation all applications must cease using permanent databases in the sense that when they are again allowed to proceed, a given database may have changed its address and size. The number of cells it contains and all cell tags of these cells may have changed. Only the segment name will be unchanged.

4 Scope of Dump

The dump feature makes a copy on a microdrive cartridge of what are called permanent databases in this paper. In user terms they are just that - collections of connected data which remain in the store permanently, even though the application(s) which process(es) the data may come and go. They are a variant of the familiar disc database which have the advantages of being always available (you can't lose the disc, or find both drives in use) and of offering instant access (which cannot be said for microdrive files on OPD). In implementation terms they are permanent segments, as described in PSD 76.97.3.2 (Application Handler PSD).

The contents of all permanent segments in store at the time of a full dump are dumped. The scope of a selective dump is determined by the calling application program.

The entire contents of the CMOS RAM are dumped if a permanent store save is requested.

The segment limits in force at the time of a dump are remembered and are reinstated as part of the restore process.

The dumped data on a microdrive cartridge is held in what is called a dump file (which is an ordinary microdrive file, though it is not intended to be reprocessed by anything other than the restore feature).

5 Scope of Restore

The full restore feature is almost non-selective. It attempts to restore all permanent databases found in the dump file into store. If a database in the dump file does not exist in store already then it is simply recreated. If it does already exist (i.e. if there is an existing database with the same name) then the existing database in store is destroyed and the database from the dump file recreated in its place. (It is expected, however, that full restoration will usually take place into an empty machine.)

The selective restore feature attempts to restore only those databases nominated by the calling application. Again, in the event of a clash, the one from the save file is the one which persists. See also section 9.1.7.

The CMOS RAM restore feature works on a similar merging principle.

6 Inter-cell References

6.1 The cell allocator memory management scheme on OPD raises problems if one wishes to export a database in some sense and re-import it later.

The unit of store allocation is the cell. These float freely round their containing segment. The segment itself may also float freely round the store. The only means of referring in one cell to another cell (in the same or a different segment) is to use a cell tag. This is a special form of address which may be converted at any time into the equivalent real address, but this real address remains valid for only a limited time. To build any kind of data structure therefore - even a simple list - cell tags must be stored in the cells, pointing to other cells.

6.2 Unfortunately, cell tags themselves have a finite life. They are valid only until the segment containing the cell is destroyed (perhaps by switching the machine off). After this, the cell tags become invalid or may be reused to point to quite different cells. It is not adequate to dump cell tags as the only structure information, since restoration is impossible.

6.3 A scheme is therefore introduced to permit such structure information to be remembered in a form which can be processed by the dump and restore features in such a way as to recreate the structures validly after restoration.

The scheme allows dump-proof inter-cell references to be made from within a cell in a cell allocator segment to a cell in a cell allocator segment. Normal segments, if they are permanent, may be dumped and restored but no assistance is offered with structuring information. Neither is assistance offered for pointers from a cell in a cell allocator segment to normal segments.

6.4 Note that real store addresses, segment numbers, segment identifiers, cell offsets down their segments and cell tags stored without this assistance all provide means of storing structure information in a database which will fail if the database is dumped and restored. They must not be used. Only registered cell tags (see below) may be used. A registered cell tag may point to a cell in the same database or a different database.

It is also invalid to leave in a permanent segment cell tags which point to cells in transient segments.

6.5 The requirement on applications (apart from only using cell tags for pointers) is to call the procedure REGISTER CROSS REFERENCE (see below) whenever a cell tag is stored and to call the procedure DEREGISTER CROSS REFERENCE (see below) whenever a stored cell tag is destroyed, probably because the enclosing cell is being destroyed.

Two further, alternative, procedures are also provided - REGISTER SPECIAL CROSS REFERENCE and DEREGISTER SPECIAL CROSS REFERENCE. There is also a procedure DEREGISTER ALL CROSS REFERENCES IN SEGMENT.

The cell tag may be used normally, no loss of speed is involved. The normal Kernel procedure, GET CELL ADDRESS, is used.

6.6 The information passed on REGISTER CROSS REFERENCE is in a data structure of its own, called the Cross Reference Table. Each registered cross reference occupies 10 bytes in the CRT. The information in the CRT is used only during dump and restore operations.

7 Release of Databases during Dumping/Restoration

There is clearly a need for applications to desist from processing databases while a dump or a restore is in progress. There are two distinct situations.

If a dump is in progress, no application may alter a permanent database, either in direct content or in the number or size of cells in a cell allocator segment. Applications may, however, continue to read from permanent databases and to use inter-cell references (cell tags), since these remain valid. A complete cessation of application activity is not therefore necessary, applications must only refrain from altering permanent databases until the dump is over.

If a restore is in progress, no application may even read from a permanent database. All databases must be released and reacquired by name when the restore is over. (This is because the restore might substitute, for a database, an entirely different database with the same name - e.g. a different telephone directory.)

In order to police this situation, each database (permanent segment) now has two counts of the use being made of it. One is the (already existing) usage count. Each application reading from the database must first increment this count and when it has finished accessing the database must decrement the count. The count has become a count of the number of readers of the database. But, incrementing the usage count no longer confers the right to alter the database (or the number or size of cells in it). This is the function of the second count, the writers count. An application writing to a database must first increment the writers count, and be prepared for a refusal. When it has finished writing to the database it must decrement the writers count for the database.

The readers count and writers count are manipulated using Application Hander facilities.

An application should expect sometimes to receive the error response, ERR.SL, from REQUEST WRITE ACCESS TO SEGMENT even though it has not received a preceding 'Cause Save' event. This will happen if the application is started after the event has been issued but before the dump has been completed.

An application should also expect sometimes to receive the error response, ERR.SL, from CHANGE SEGMENT PROPERTIES, GET SEGMENT IDENTIFIER or USE SEGMENT, even though it has not received a preceding 'Cause Load' event. This will happen if the application is started after the event has been issued but before the restore has been completed.

The onset of a dump or a restore is a 2 stage process. The first stage starts when the event (see section 11) is issued and lasts until all writers counts are zero (or the package elects to proceed anyway). This stage is signified in the status byte as Save (or Load) Pending. During this stage attempts to gain write access to segments are failed but attempts to gain read access to segments are permitted. it is also permitted to destroy a segment or change its properties (e.g. from permanent to temporary or vice-versa).

The second stage is when the save or load is actually in progress. The controls applied are more stringent in this case, all access requests, property change requests and segment destruction request are refused in the case of a load operation. In the case of a save operation, read access requests, segment information requests and changes in the reviewable property of an existing permanent segment are permitted.

A full-scale disengagement is requested by dumper even for a selective dump or restore, or for a CMOS RAM dump or restore.

It is not intended to be dogmatic about application response to an incipient dump or restore. An application designer could choose to react in the same way to both dump and restore, or to treat a 'Cause Load' event in the same way as an 'Abandon' event.

Permanent cell-allocator segments are always Squashed (see Kernel spec.) before being dumped [though this cannot be achieved if the segment remains locked at the time of the dump, see section 8.1].

8 Action Sequences

8.1 Dumping Sequence

When a dump initiation request is accepted by INITIATE DATA SAVE, it issues a global event, 'Cause save'. This asks all applications to relinquish write access to the permanent segments (The position is similar if the event is issued by Housekeeping, for a manually-initiated dump, or by dumper itself, for a timed dump.)

Dumper then waits until all permanent segments and the CMOS RAM have writers counts of zero. There is a time limit on this waiting process of about 7 minutes, after which (if one or more segments are still not available) the dump attempt proceeds anyway. This should only occur if there is a rogue application in the OPD which does not release its write access to segments on request.

The status of a segment which was dumped even though its writers count was not zero is difficult to ascertain. Much depends on the reason for the failure of the application to release write access. The dump is taken anyway, even though this particular segment may be dumped corruptly, as it seems overall to be the best of an unappealing set of possibilities.

When the waiting process has finished dumper obtains a suitable cartridge for writing the new dump file on (see PSD 76.97.2.1). It then, if necessary, erases the old dump.

When the segments are available, the dump is made. When the dump file has been written and closed a Save/Load Finished event is caused, the last operation status is set to 'OK' and a message is sent to the user.

The dump is now over and applications are free to (re-)request write access to the permanent segments.

8.2 Restoring Sequence

When a restore initiation request is accepted by INITIATE DATA LOAD it issues a global event, 'Cause load'. This asks all applications to relinquish all access to permanent segments. (Again, the position is similar if the event is issued by Housekeeping or by dumper itself).

Dumper then waits until all permanent segments and the CMOS RAM have usage counts of zero (and, by implication, writers counts of zero). The waiting process is time-limited, as with dumping.

The remainder of the process is similar to that for dumping, except that when it is over applications should (re-)request read access, and perhaps write access, as required.

8.3 Corrupt Inter-cell References

There is no rigid control over the contents of the Cross Reference Table. It is possible for a wrong cross reference to be put into the table or for an initially-valid cross reference to become invalid as a result of subsequent application actions. It is also possible for the cell tag which should be present at the point of a registered cross reference to be invalid.

Broadly speaking, corrupt references of this kind are ignored by the dump/restore process. A registered cross reference which is found to be invalid is simply not restored.

A stored cell tag which is invalid at the time of the dump is restored as a zero cell tag.

It is permissible and may be useful, for stored cell tags to be invalid at the time of registration of the cross references or subsequently, providing that either they are valid by the time a dump is taken or that the relevant applications are prepared to cater for such tags being returned as zeros after a restore.
In particular, applicaitons may use zero cell tags in registered cross references to indicate (to themselves) that no pointer is in fact present at this position, subject to the previous paragraph.

8.4 Cross References in Transient Segments

This is nothing to prevent cross references being registered in transient segments. There is, however, no point in doing this unless the transient segments are going to be turned into permanent segments before the next dump.

Cross references registered in transient segments do not survive the dump/restore process.

9 Procedural Interfaces to Data Record Facility

9.1 Eight procedures are available for use by applications. There is an additional "procedure" which is used to set up the dump & restore package at the start of the day. This is described elsewhere.

9.1.1 REGISTER CROSS REFERENCE

Trap Name: T.DATARCH
Action Value (D0.B): DA.RXREF

Additional Call Parameters:

D1.W : cell tag
D2.W : displacement

Error Returns:

BP : bad parameter
OM : out of memory (Cross Reference Table is full and cannot be extended)
SL : save or load in progress

This call notifies the Data Record Facility that there is an inter-cell reference which is located at (displacement) bytes down the cell referred to by the cell tag passed as a parameter. The inter-cell reference itself is not passed as a parameter to this procedure.

The location of the inter-cell reference is stored in the Cross Reference Table.

If error response ERR.OM is received, neither the segment nor the inter-cell reference are corrupt, though the segment can no longer be dumped and restored correctly unless the stored cell tag is removed.

9.1.2 DEREGISTER CROSS REFERENCE

Trap Name: T.DATARCH
Action Value (D0.B): DA.XXREF

Additional Call Parameters:

D1.W : cell tag
D2.W : displacement

Error Returns:

BP	:	bad parameter
SL	:	save or load in progress

This call notifies the Data Record Facility that the inter-cell reference which is located at (displacement) bytes down the cell referred to by the cell tag passed as a parameter is no longer required.

The location of the inter-cell reference is deleted from the Cross Reference Table. The inter-cell reference itself (a cell tag) is not altered by this procedure, though it could no longer be used validly following a dump and a restore of the segment holding it.

9.1.3 DEREGISTER ALL CROSS REFERENCES IN SEGMENT

Trap Name:	T.DATARCH
Action Value (D0.B) :	DA.XAXREF

Additional Call Parameters :

D1.L : segment identifier

Return Parameters :

None

Error Returns :

SL : save or load in progress

This call notifies the Data Record Facility that all the inter-cell references which have been registered as being in the segment whose identifier is passed as a parameter are to be deregistered i.e. are no longer required.

The locations of all inter-cell references within the segment are deleted from the Cross Reference Table. The inter-cell references themselves are not altered by this procedure, though they could no longer be used validly following a dump and restore of the segment.

This call is intended to be used prior to total deletion of segment. It may be used without harm if the segment is not a cell allocator segment or if it contains no registered inter-cell references, though this achieves nothing useful.

9.1.4 REGISTER SPECIAL CROSS REFERENCE

Trap Name: T.DATARCH
Action Value (D0.B): DA.RSXREF

Action Call Parameters:

D1.W : cell tag

Return Parameters:

None

Error Returns :

BP : bad parameter
OM : out of memory (Cross Reference Table is full and cannot be extended)
SL : save or load in progress

This call notifies the Data Record Facility that there is a Special Cross Reference Pair at the start of the cell whose tag is passed as a parameter. The inter-cell references themselves are not passed as parameters to this procedure.

The location of the Special Cross Reference Pair is stored in the Cross Reference Table.

A Special Cross Reference Pair comprises two inter-cell references which obey the following implicit rules,

a) the cell tags comprising the references are stored in bytes 0 & 1 and bytes 2 & 3, respectively, of the cell whose tag is passed as a parameter.

b) Both inter-cell references are to cells in the same segment as the segment containing inter-cell references.

This special case is provided primarily to reduce the overhead incurred in the Cross Reference Table by the Telephone Directory.

9.1.5 DEREGISTER SPECIAL CROSS REFERENCE

Trap Name: T.DATARCH
Action Value (D0.B): DA.XSXREF

Additional Call Parameters:

D1.W : cell tag

Return Parameters:

None

Error Returns:

BP : bad parameter

SL : save or load in progress

This call notifies the Data Record Facility that the Special Cross Reference Pair at the start of the cell whose tag is passed as a parameter is to be deregistered, i.e. is no longer required.

The location of the Special Cross Reference Pair is deleted from the Cross Reference Table. The inter-cell references themselves (two cell tags) are not altered by this procedure, though they could no longer be used validly following a dump and a restore of the segment holding them.

9.1.6 INITIATE DATA SAVE

Trap Name: T.DATARCH
Action Value (D0.B): DA.SAVE

Additional Call Parameters:

D1.W : #DA.FULL for a full store save
or #DA.DSELECT for a dual selective store save
or #DA.SSELECT for a single selective store save
or #DA.NVM for a CMOS RAM save

D2.L : If, and only if, D1.W = #DA.DSELECT or #DA.SSELECT each byte in D2 should contain a segment number of a segment to be saved. Unused byte positions in D2 should be zero. The segment number may be extracted as bits 0-7 of the segment identifier.

D3.L : Needed if, and only if D1.W = #DA.DSELECT or DA.SSELECT.

Bits 0-15 : NVM identifier. If non zero, is taken as a CMOS RAM identifer. A CMOS RAM entry will be created or updated and the name of volume holding the dump file written during the dump will be stored in the entry.

Bits 16-23 : File letter. This parameter must be an upper case letter. It is used as the second letter of the dump file name. It must be supplied if D1.W = #DA.DSELECT or #DA.SSELECT

Return Parameters:

```
D1.W :  Data Record Facility Status Register
            bit 0 = 1 - a save operation is pending
                  = 0 - otherwise
            bit 1 = 1 - a load operation is pending
                  = 0 - otherwise
            bit 2 = 1 - a save operation is in progress
                  = 0 - otherwise
            bit 3 = 1 - a load operation is in progress
                  = 0 - otherwise
            bit 4 = 1 - the last operation was a save
                  = 0 - otherwise
            bit 5 = 1 - the last operation was a load
                  = 0 - otherwise
            bit 6 = 1 - the last operation failed
                  = 0 - the last operation ended OK
```

Error Returns:

SL : initiation request rejected as an operation is already in progress.

If a save or a load operation is already in progress then the request is rejected. The current status is available in D1.W. (This should be regarded as an abnormal condition since all applications present at the time when the current operation was started should presumably be moribund in this respect until it has ended).

If the request is accepted a Cause Save event is issued to all applications and a return is made to the caller immediately with the requested operation shown as pending in the Status Register.

The requested operation starts when all the segments and a cartridge are available and then proceeds autonomously. Termination, successful or not, is signalled by a Save/Load Finished event. It is not specifically reported to the activity which initiated the operation. Any application may then enquire the result by calling GIVE DATA STATUS.

9.1.7 INITIATE DATA LOAD

Trap Name: T.DATARCH
Action Value (D0.B): DA.LOAD

Additional Call Parameters:

```
D1.W :      #DA.FULL     for a full store load
         or #DA.DSELECT  for a dual selective store load
         or #DA.SSELECT  for a single selective store load
         or #DA.NVM      for a CMOS RAM load
```

D2.L : Should be set to -1 if D1.W = #DA.DSELECT or #DA.SSELECT.

Not needed if D1.W = #DA.FULL or #DA.NVM and is ignored in these cases.

D3.L : Needed if, and only if, D1.W = #DA.DSELECT

Bits 8-15 : Substitute segment name letter. If and only if D1.W=#DA.DSELECT or #DA.SSELECT this byte is substituted for the third letter of every segment name contained in the dump file. The effect is that all the segments are restored into permanent segments whose names differ in the respect from their original names and that copies of the databases with original names are not deleted from the store by the restore process.

For example, if bits 8-15 = $48, a segment in the dump file with a name of ABCDE is restored as a permanent segment with a name of ABHDE, and any existing segment in the store with a name of ABCDE remains there.

If no substitution is required, set bits 8-15 =0.

Bits 16-23 : File letter. This is used as the second letter of the dump file name. It must be supplied if D1.W = #DA.DSELECT or #DA.SSELECT.

Bits 0-7 and 24-31 : must be 0

Return Parameters :

```
D1.W :  Data Archive Facility Status Register
            bit 0 = 1 - a save operation is pending
                  = 0 - otherwise
            bit 1 = 1 - a load operation is pending
```

```
      = 0 - otherwise
bit 2 = 1 - a save operation is in progress
      = 0 - otherwise
bit 3 = 1 - a load operation is in progress
      = 0 - otherwise
bit 4 = 1 - the last operation was a save
      = 0 - otherwise
bit 5 = 1 - the last operation was a load
      = 0 - otherwise
bit 6 = 1 - the last operation failed
      = 0 - the last operation ended OK
```

Error Returns:

SL : initiation request rejected as an operation is already in progress.

If a save or a load operation is already in progress then the request is rejected. The current status is available in D1.W. (This should be regarded as an abnormal condition since all applications present at the time when the current operation was started should presumably be moribund in this respect until it has ended).

If the request is accepted a Cause Load event is issued to all applications and a return is made to the caller immediately with the requested operation shown as current in the Status Register.

The requested operation starts when all the segments and a cartridge are available and then proceeds autonomously. Termination, successful or not, is signalled by a Save/Load Finished event. It is not specifically reported to the activity which initiated the operation. Any application may then enquire the result by calling GIVE DATA STATUS.

A single selective load allows the application to specify the file name of the save file to be sought (within the usual convention). If the load does not succeed then a failure message is output and the operation is abandoned.

A dual selective load allows the application to specify the file name of the save file to be sought. If the load of the newest save file does not succeed then no failure message is output. Instead an attempt is made to load from the second newest save file on the cartridge containing the newest save file (which is assumed to exist - the pair are assumed to have been produced as a dual dump). If this load succeeds then no comment is made to the user. If this load also fails then the normal failure message appears to the user.

9.1.8 GIVE DATA STATUS

Trap Name: T. DATARCH

Action Value (D0.B): DA.STATUS

Additional Call Parameters:

None

Return Parameters:

```
D1.W :  Data Archive Facility Status Register
            bit 0 = 1 - a save operation is pending
                  = 0 - otherwise
            bit 1 = 1 - a load operation is pending
                  = 0 - otherwise
            bit 2 = 1 - a save operation is in progress
                  = 0 - otherwise
            bit 3 = 1 - a load operation is in progress
                  = 0 - otherwise
            bit 4 = 1 - the last operation was a save
                  = 0 - otherwise
            bit 5 = 1 - the last operation was a load
                  = 0 - otherwise
            bit 6 = 1 - the last operation failed
                  = 0 - the last operation ended OK
```

Error Returns:

None

9.2 Use of Procedural Interfaces

Few applications should expect to need to use INITIATE DATA SAVE. The only currently recognised examples are Messaging (for its security feature) and Housekeeping (for a dump initiated directly by the user).

Similarly, few applications should expect to need to use INITIATE DATA LOAD. The only currently recognised example is Housekeeping (for a restore initiated directly by the user).

Any application may use GIVE DATA STATUS, though it seems probable that only Messaging and Housekeeping will do so in the near future.

All applications which maintain or use permanent segments should expect to use REGISTER CROSS REFERENCE and DEREGISTER CROSS REFERENCE (and also to have to take notice of the 3 global events used for signalling).

The user facility to request an automatic load on power-up is handled entirely by the Data Record Facility.

The user facility to request a dump at a particular time of day is handled entirely by the Data Record Facility.

10 New Application Handler Procedural Interfaces

These are [now] documented in PSD 76.97.3.2 (Application Handler PSD).

11 Event Usage

Three global events are dedicated to communication between dump/restore and other activities. All are classified as Global/Internal.

11.1 Event 24 - name: Cause Save

This has two effects

a) asks the dump/restore package to initiate a save attempt when the permanent segments become free.

and b) notifies all applications that they are to make free all permanent segments they are using so that they may be saved. The segments must remain free until the dump/restore package has caused an event 26.

11.2 Event 25 - name: Cause Load

This has two effects

a) asks the dump/restore package to initiate a load attempt when all permanent segments are no longer in use.

and b) notifies all applications that they are to discontinue use of all permanent segments they are using (since the load may introduce an entirely different segment with the same name - i.e. a different copy of the same segment). The segments must not be used again until the dump/restore package has caused an event 26.

11.3 Event 26 - name: Save/Load Finished

This notifies all activities that the save or load attempt has now been completed and that the permanent segments are again available for use by activities.

Any activity may enquire the status of the terminated attempt by using the procedure supplied.

Applications should <u>not</u> cause any of these events themselves, but should use the procedures supplied (which will also attempt to ensure that only one is outstanding at a time).

Applications using permanent segments must have set in their Event Request Registers the bits corresponding to these three events.

12 Application Handler Interlocks

Application Handler applies certain checks to calls on its procedures concerned with permanent segments. The purpose of these checks is the prevention of mutual interference between Data Record Facility and applications using permanaent segments. The checks are outlined below.

12.1 A call on REQUEST READ ACCESS TO SEGMENT is rejected, ERR.SL, if a restore (load) is either in progress or is pending.

12.2 A call on REQUEST WRITE ACCESS TO SEGMENT is rejected, ERR.SL, if a restore is either in progress or is pending or if a dump (save) is in progress is pending.

12.3 Calls on RELEASE WRITE ACCESS TO SEGMENT and RELEASE ACCESS TO SEGMENT are always permitted.

12.4 A call on GET SEGMENT IDENTIFIER (OF PERMANENT SEGMENT) is rejected, ERR.SL, if a restore is in progress but is permitted at other times, and in particular when a restore is pending.

12.5 A call on DESTROY SEGMENT is rejected, ERR.SL, if a dump or a restore is in progress but is permitted at other times, and in particular when a dump or a restore is pending.

12.6 A call on CHANGE SEGMENT PROPERTIES is rejected, ERR.SL, if a dump or a restore is either pending or in progress.