The ENTER WINDOW Statement

The ENTER WINDOW statement causes a window to be executed.

These statements can be coded in four divisions within each frame. These are:

WINDOW DIVISION	within the ROUTINES SECTION
PROCEDURE DIVISION	within the SECTION/ENTRY
LOAD DIVISION	
UNLOAD DIVISION	

When a window is entered, the operator is normally able to add, maintain or delete a number of records. Enquiries may also be performed in order to select a record from the database for maintenance or deletion.

During this processing, routines coded in the Routines Section may be called to perform various tasks such as field validation. Provided these routines do not detect errors, the record will then be written, or rewritten, to the database. This normally concludes processing and control is returned to the statement immediately following the ENTER statement. Alternatively, the operator may abort the window by keying

1. Statement Construct

The ENTER statement causes a window to be executed. It is coded:

```
ENTER WINDOW window-id [INDEX | ENQUIRE rtinx] [(line:col)]
```

where *window-id* is the ID assigned by the window statement. This normally allows the operator to add, select, maintain, or delete one or more records stored on a database. The functions actually available to the operator will depend on the options coded for the window.

The optional *line* and *col* clause is used to define a new position on the screen for the window, and thus allows the window to be dynamically positioned. The variables define the top left-hand corner of the window, and may be coded as constants or as 9(4) comp variables. If either the line or column number is zero, the window will revert to its normal position.

The optional INDEX (GSM SP-19 and later) clause causes the window to be entered with an initial index of *rtinx*. This index must be available in enquiry mode in the window. (See the section on window indexes below).

The optional ENQUIRE (GSM SP-19 and later) clause causes the window to be entered in <ENQ> mode with an initial index of *rtinx*. This index must be available in enquiry mode in the window.

2. ENTERX Statement

The ENTERX statement is a specialised version of the ENTER statement available on GSM SP-25 and later . It is coded:

```
ENTERX WINDOW wd ENQUIRE|INDEX|MAINTAIN rtinx  [(line:col)]
```

The *window-id*, *line* and *col* is as above. The ENQUIRE clause causes the window to be entered in ENQ mode with an initial index of *rtinx*. This index must be available in enquiry mode in the window.

The INDEX clause causes the window to be entered with an initial index of *rtinx* and with the current record at the top of the window. The index specified must be available in enquiry mode within the window (see the section on window indexes below) and the current record must be a valid record.

The MAINTAIN clause causes the window to be entered with an initial index of *rtinx* and with the current record at the top of the window. The index specified must be available in enquiry mode within the window (see the section on window indexes below) , the current record must be a valid record, and MNT mode must be available in the window.

3. Successful Completion

Depending on the options specified in the window construct, one or more records may have been processed, and the window will have been successfully completed. Control will be returned on successful completion only when this is permitted by the window's sequence statement.

4. Exception Conditions

Unsuccessful Completion occurs when the window is aborted by the user keying <ABO> or <BCK> or by the window's process routine or function routine returning any exception. When the <ABO> function is keyed, or any exception other than 1 is returned by the process routine, control is immediately returned to the statement following the ENTER instruction. Otherwise the window specified in the sequence statement, if any, is entered next.

These conditions can be differentiated by testing the system variable $$COND. If $$COND=1, <BCK> was keyed, if $$COND=2, <ABO> was keyed. If an exception is passed back by the window's process or function routine, this exception number will also be passed back in $$COND. These exception conditions must be trapped by coding an ON EXCEPTION statement immediately following the ENTER verb.

5. Window Indexes

This section applies to windows with a USING *rt* | *rtinx* clause in the window statement where *rt* is the record-id and *rtinx* is the window index.

Enquiry Indexes

These are the indexes available in the window for ordering records. The indexes can be switched in ENQ mode BY SELECTING THE <ENQ> key/button . If there is no DEPENDENT clause in the window statement then any index whose first field is available in the window can be enquired on. The initial index (see below) is always available.

For windows where a DEPENDENT on clause is coded, only indexes beginning with the fields in the DEPENDENT clause in the specified order that have no further fields, or whose next field segment is present in the window, will be made available in the window as an ENQ index. If an index contains more than eight segments only the first

8 segments of the index will be considered. The initial index (see below) is always available.

Initial Index

The initial index used for ordering the records in the window if no INDEX clause is present in the ENTER statement, depends on whether the WINDOW statement contains a DEPENDENT ON clause or not.

Windows without a DEPENDENT ON Clause

If the WINDOW statement has a USING *rtinx* clause this index will be the initial index unless its first field segment is not present in the window. If this index cannot be used, the first index whose first field is present in the window will be the initial index. If there are no such indexes then the first index in the record set will become the initial index.

Windows with a DEPENDENT ON Clause

If the WINDOW statement has a USING *rtinx* clause, this index will be the initial index unless the first field segments of the index do not match those in the DEPENDENT ON clause. If there are no further field in the index then this index will be used. However. If there are further fields and the next field segment of this index key is not present in the window then the index will not be used as the initial index. In this case or if there is no index present in the WINDOW statement , the first index whose first field segments match those in the DEPENDENT ON clause and whose next segment is present in the window will be the initial index. If there are no such indexes then the first index in the record set will become the initial index.

Examples

With the following dictionary:

```
.BOOK aa
**********************************************************************
**                                                                  **
** REC TYPE: aa  NAME:z2aa   Test master                            **
** DICT:DI$$Z2   DBID:$$Z2   GEN#:    4   CREATED:19/09/2007        **
**                                                                  **
**  INDEX  1  aaidx1  (  4) seg1/seg2
**  INDEX  2  aaidx2  (  2) seg2                                   **
**                                                                  **
**********************************************************************
*
  03 &&seg1             PIC 9(4)     C  *   1/      Segment 1
  03 &&seg2             PIC 9(4)     C  *   3/      Segment 2
  03 &&text             PIC X           *   5/      Text
  03 &&noab             PIC 9(4)     C  *   6/ GVA  No of linked ab's
*
.END
.BOOK ab
**********************************************************************
**                                                                  **
** REC TYPE: ab  NAME:z2ab   Test slave                             **
** DICT:DI$$Z2   DBID:$$Z2   GEN#:    4   CREATED:19/09/2007        **
**                                                                  **
**  INDEX  1  ab$sq   (  8) seg1/seg2/$seq                          **

**  INDEX  2  abidx2  (  6) seg1/seg2/seg3                          **
**                                                                  **
**  MASTER  1  aa  z2aa    Test master                              **
**                                                                      **
**************************************************************************
*
  03 &&seg1              PIC 9(4)     C  *   1/      Segment 1
  03 &&seg2              PIC 9(4)     C  *   3/      Segment 2
  03 &&$seq              PIC 9(9)     C  *   5/      Sequencer
  03 &&seg3              PIC 9(4)     C  *   9/      Segment 3
  03 &&text              PIC X           *  11/      Text
  03 &&aone              PIC 9(4)     C  *  12/      Always=1
*
.END
```

For the following window:

```
WINDOW W1 USING aa
SEQUENCE EXIT, EXIT
REPEAT UNTIL CURRENT RECORD
AUTOPGE
BASE AT  2  2
SCROLL 10 BY  1 SPLIT  1 OFFSET  1
02 02  "Masters"                                  A12
                04 02  aaseg1      9(4) C         NOE
                04 07  aaseg2      9(4) C         NOE  CHK
                04 12  aatext      X              NOE
                04 22  aanoab      9(4) C         DIS
03 02  "Seg1"                                     NSC
03 07  "Seg2"                                     NSC
03 12  "Text"                                     NSC
03 18  "Servants"                                 NSC
ENDWINDOW
```

The first key segment of the first index of the target record set aa is present in the window. Index aaidx1 will therefore be chosen as the initial index. If aaseg1 is removed from the window, then aaidx2 will be chosen as the initial index because the first key segment of aaidx1 would not be present but that of aaidx2 would be. If both aaseg1 and aaseg2 were removed from the window, aaidx1 would be chosen, as there would be no index whose first segment is present in the window.

For the following window:

```
WINDOW W2 USING ab$SQ DEPENDENT ON aa
SEQUENCE EXIT, EXIT
REPEAT
AUTOPGE
BASE AT 12 16
SCROLL 10 BY  1 SPLIT  1 OFFSET  1
02 02  "Servants of;"                             A12
                03 13  aaseg1      9(4) C         NSC DIS
                04 13  aaseg2      9(4) C         NSC DIS
                07 02  abseg3      9(4) C         NUL
                07 10  abtext      X
06 08  "Text"                                     NSC
06 02  "Seg3"                                     NSC
03 07  "Seg1"                                     NSC
04 07  "Seg2"                                     NSC
ENDWINDOW
```

The target index ab$SQ is looked at first. The first significant segment of the key, i.e. the first segment that is not part of the dependent on index ab$SEQ is not present in the window. The first significant key of the second index, that is abseg3, is present on the window so this index, abidx2, will be chosen as the initial index. If abseg3 is removed from the window the initial index would then be ab$SQ as there would be no indexes whose first significant segment is present in the window.

6. See Also

WINDOW statement
DISPLAY WINDOW statement
CLEAR WINDOW statement
WINDOW Options
WINDOW Body
ROUTINES SECTION