The ROUTINES SECTION

The optional Routines Section is coded immediately following the window body. It is introduced by the header:

ROUTINES SECTION

The routines section consists of a number of routines which are called during the various stages of window processing. Each routine is identified by a special label which determines when during the processing cycle it will be executed by the window manager.

Each routine may contain any valid procedural instruction including window management statements such as ENTER and DISPLAY, and on completion of processing must return control to the window manager by executing an EXIT instruction. It is therefore permissible to enter or display other windows **from within the Routines Section, providing these are not currently executing.**

1. The Routines

The entry-points provided by the routines section are divided into three types, entries called during field level processing, and entries called during record level processing, and entries called at window level processing as below:

Routine	Description	Function	Level
B-*name*	Before field	Suppress optional fields	Field
V-*name*	Validate field	Perform extra validation	Field
D-*name*	Default routine	Default field contents	Field
R-FETCH	Record fetch	Reject retrieved records	Record
R-SELECT	Record select	Suppress selection	Record
R-DELETE	Record delete	Suppress deletion	Record
R-WRITE	Record write	Suppress record write	Record
R-REWRITE	Record rewrite	Suppress rewrite	Record
R-PROCESS	Record process	Extended processing	Record
R-TERM	Record terminate	Release locks	Record
R-START	Record start	Mode change processing	Window
R-FUNC	Function Intercept	Function/button processing	Window
R-INIT	Window entry	Entry processing	Window

Routines Section Entry-Points

Most of the above routines may return an exception to the window manager using the EXIT statement. EXIT WITH 1 has a general "incorrect - do not proceed" meaning. Other exit conditions are also used, and these are further explained below.

2. The Before Routine (B-*name*)

The Before Routine is called immediately before the field is accepted or displayed. Returning control with exit condition 1 (i.e. performing EXIT WITH 1) causes the field to be suppressed. When suppressed, the area on the screen normally occupied by the field is cleared, and no further processing takes place for it. If the field has the OPT option coded the field will not appear. The before routines are always called before the field is displayed in the window. Note that Before routines are not called in ENQ mode processing. Returning control with exit condition 2 causes the field accept operation to be suppressed as if the PRO option were coded.

3. The Default Routine (D-*name)*

The Default Routine is called before a field is processed and allows a default to be provided. Note that the routine is only called during ADD and INS modes, since fields are regarded as pre-initialised during MNT mode. The default is simply MOVEd into the field and the operator may accept or change it. Before moving a default value into the field, the default routine should check that the field has not already been initialised. Uninitialised fields will be set to spaces if character fields and zero for computational and display numeric fields. It is also important to note that this routine is **not** called in Maintenance mode.

4. The Validation Routine (V-*name*)

The Validation Routine is called immediately after a field has been accepted, and may be used to perform additional validation such as range checks and can also be used to produced field level error messages. Returning exception 1 indicates that the field is invalid, and causes it to be re-input. Note that the validation routine is called even if it is not possible to ACCEPT the field (e.g. a protected field).

5. The Fetch Routine (R-FETCH)

The Fetch Routine is called whenever the window manager fetches a record from the database. The routine may be used to create derived fields before the record is displayed. Where special display formats are required, the routine may also be used to convert fields from database to external formats. It is called immediately after retrieval of the target record, but before any other processing has taken place.

5.1 Suppressing Record Retrieval

The R-FETCH routine can also be used to suppress the retrieval of certain records, such as suppressing inactive customers. This is achieved by returning exit condition 1, and causes the window manager to proceed as if the record did not exist. This allows a selection of records to be displayed during enquiry operations.

5.2 Range Checking

It is preferable to use the DEPENDENT on clause in the WINDOW statement to restrict the range of records shown in a window. However there may be some occasion where this facility is not sufficient.

The R-FETCH routine may also be used to perform extended range checking Of greater flexibility then that available in the DEPENDENT clause in the WINDOW statement.

To process a given range of keys, the R-FETCH routine tests whether the supplied record (i.e. the record currently in the I/O channel) is within the required range. If in range, all is well and no further action is required. If not, the program repositions the I/O channel at the first record or last record within the permitted range.

Where the current key value precedes the required range, the program reads the first record in the required range, and returns Exception 100. This indicates to Window Manager that the key value has been ADVANCED (in terms of ASCII Index Value) to a further point in the index.

Where the key value exceeds the required range, the program reads the last record in the required range and this time returns Exception 101. This indicates to Window Manager that the ASCII key value has been SET BACK to a prior position in the index.

When there are no records stored on the database within the required range, the program returns Exception 102 in the R-FETCH routine.

The following example demonstrates the coding technique. Using the DEMO database TR record, we want a Window which displays TR records in the range TR04 and TR08.

```
R-FETCH.
*
     IF TRTRNO < "TR04"                  * BEFORE START OF RANGE
       FETCH FIRST TRTRN KEY "TR04"      * RETURN RECORD > = TR04 OR EOF
       IGNORE EXCEPTION                  * SOF CONDITION CAUGHT BY NEXT LINE
       IF TRTRNO > "TR08" EXIT WITH 102  * NO RECORDS IN RANGE
       EXIT WITH 100                     * KEY VALUE ADVANCED EXCEPTION
     END
*
     IF TRTRNO > "TR08"                  * AFTER END OF RANGE
       FETCH LAST TRTRN KEY "TR08"       * GET THE LAST IN RANGE OR SOF
       IGNORE EXCEPTION                  * SOF CONDITION CAUGHT BY NEXT LINE
       IF TRTRNO < "TR04" EXIT WITH 102  * NO RECORDS IN RANGE
       EXIT WITH 101                     * KEY VALUE RETARDED EXCEPTION
     END
     EXIT
```

This technique is can only be used when the enquiry index is in the same order as the range you want to select. If it is not, the required value range is spread all over the index, meaning that there will be no start or end point you can position on.

A window written using this technique must therefore be restricted to allow enquiry only using the relevant index(es). In the above example, it is necessary to restrict the window to the TRTRN index only, or alternatively to disable the selection code when that index is not the currently selected enquiry index.

6. The Select Routine (R-SELECT)

The Select Routine is called when the operator attempts to select a record, usually to enter MNT mode. The record will already have been fetched and is displayed. The routine may be used to stop the operator from selecting the record, and this is achieved by returning exit condition 1. For example, this might be useful to stop the operator from attempting to amend an invoiced order.

7. The Delete Routine (R-DELETE)

The Delete Routine is called after the operator has keyed <DEL> to delete the current record, but before the deletion is performed by the window manager. The routine may reject the deletion request by returning exception condition 1. For example, this may be required to stop the deletion of an invoice before it has appeared on a statement.

The routine may also be used to remove unwanted servant records. For example, when the operator requests the deletion of an order header, the routine might automatically delete all servant order lines thus allowing the deletion to succeed.

8. The Write Routine (R-WRITE)

The Write Routine is called immediately before a new record is written to the database during ADD or INS mode. The routine may be used to complete fields on the record, before it is actually written (e.g. calculate the extended value of a line item). Returning exit condition 1 returns the operator to the last input field on the record, and suppresses the write operation. A specialise EXIT WITH 5 is also available. This exception causes the actual write operation to be ignored carrying on to the next record. The must be used with care and only in windows where the fields in the record can never actually be entered, although other fields may be available.

9. The Rewrite Routine (R-REWRITE)

The Rewrite Routine is called immediately before an existing record is re-written to the database during MNT mode. Returning exit condition 1 returns the operator to the last input field on the record, and suppresses the rewrite operation. This routine is otherwise identical to the write routine A specialise EXIT WITH 5 is also available. This exception causes the actual rewrite operation to be ignored carrying on to the next record. The must be used with care and only in windows where the fields in the record can never actually be updated i although other fields may be available.

10. The Process Routine (R-PROCESS)

The Process Routine is called **after** processing has been completed (i.e. on completion of ADD, INS, EDT, MNT or DEL modes). It may be used to perform additional updates, such as writing details of the transaction to an audit log. The system variable $MODE may be examined by the process routine to determine in which mode the record was processed if this is important (See $MODE).

When the routine is called, the record that has just been processed by the window manager is still contained within the I/O channel. The fields of the record may therefore be examined by the routine. It is important to note, however, that the I/O operation (i.e. write, rewrite or delete) will already have taken place. In the case of deletion, this means that the record no longer exists on the database at the time that the process routine is called.

Following ADD, INS, and MNT modes, the current record in the I/O channel will normally be locked, to ensure that it is not modified or deleted while the process routine executes. For ADD, INS and MNT modes, the record will be locked exclusively. For SEL mode the record will be protected (locked non-exclusively) unless the LOCK or NOLOCK options are in force.

Returning Exit condition 1 (i.e. performing an EXIT WITH 1) from the process routine returns unsuccessful completion of the window, and causes the back-action to be taken as defined by the window's optional Sequence statement. It should be noted that this has no effect on the processing of the last transaction, which will already have been written to the database.

Returning Exit condition 2 causes window processing to be terminated just as if the operator had keyed a series of <BCK> keys to terminate the current series of windows as defined by the window's sequence statement. Where no sequence statement has been coded, the effect of this is identical to returning exception condition 1.

11. The Terminate Routine (R-TERM)

The Terminate Routine is called when the operator terminates record processing in MNT, EDT, ADD or INS modes by keying a function such as <BCK>, <HME>, <CLR>. It may be used, for example, to release locks established by earlier, now aborted, record processing, such as may have occurred within the window's validation routine. System variable $FUNC may be examined to determine the function used to terminate record processing. Note that the variable $MODE is not defined when the R-TERM routine is processed. Note that the R-TERM routine must not be used to unlock any window target record type.

12. The Start Routine (R-START)

The R-START entry is called whenever Window Manager begins processing for a given operating mode ($MODE). The entry-point has been provided to allow you to Enable or Disable buttons appropriate for the current Mode using the ENABLE and DISABLE statements (see ENABLE/DISABLE documentation). Your code should simply test $MODE, and then enable the buttons you want enabled, and disable the buttons that are invalid for the mode. In general, you should code all of your non-Temporary ENABLE and DISABLE statements in this routine.

The Entry-point is called immediately before the first accept operation takes place as processing begins in all modes. For symmetry and completeness, the R-START entry-point is called in Delete mode even though no accept operation normally takes following on a record deletion request.

Note that unlike the R-FETCH routine, the entry-point is NOT called each time a record is read. You cannot therefore use this entry-point to set field Display Attributes, except for the record that is currently being processed.

You can use the R-START Routine even when using a mixture of both System and Application Button functions. Immediately before calling the R-START routine, Window Manager will already have enabled the appropriate System Function buttons, and will have set the Default System Button according to its rules. You can then use the ENABLE .. DEFAULT statement to over-ride the standard Window Manager default button, which then remains in force until R-START is next called.

13. The Function Routine (R-FUNC)

The Function Routine is called by Window Manager whenever any System or Application Function is requested. The R-FUNC Routines Section Entry-point allows ALL functions to be intercepted and processed by the Application Program. It is used to process Application Buttons, and to augment or over-ride the processing of standard System Functions.

The routine is called irrespective of the source of the request and returns all functions generated by all Key-stroke, Button-click, Toolbar, TYP$ and Mouse operations.

The requested function number is returned in System Variable $FUNC. Thus when the button coded as BTN U78 is clicked, $FUNC will contain the value 78, or following a <PGE> keystroke, the value 5. Window Manager supplies the Symbol name of the Variable being accepted when the function was requested in Variable $VARNAME. Thus if the button was clicked while on field TRNAME, system variable $VARNAME will contain the value "TRNAME".

This can be extremely useful when the action to be taken depends on the field being processed. For example, you may want to define a "Search" button, but what is searched may depend on the field being processed. Note that when the R-FUNC routine is called in DSP mode, no particular field is in the process of being accepted, and $VARNAME is therefore null.

Having identified the requested function, the R-FUNC routine may immediately perform the requested action, which might involve invoking another window or some other process. Alternatively the routine may make use of the TYP$ interface to cause window manager to take some action, such as deleting a series of records, or forcing an exit from the window.

A EXIT WITH 1 condition returned by the R-FUNC routine will cause Window Manager to ignore the function, and resume processing at the current field. This allows you to suppress an inappropriate function at any time. An EXIT condition > 1 causes the Window to be terminated immediately, with the Exit code passed back to the calling ENTER WINDOW statement.

It should be noted that the processing to be performed for an Application Button is entirely in the hands of the application programmer. Window Manager simply passes the event to the R-FUNC routine, after which it simply resumes the last accept operation.

14. The Initial Routine (R-INIT)

This routine is called whenever the window is entered or displayed and can be used to make window level adjustments to the window. In particular, making adjustments to a button top for a window may be better done in this routine, which is only called on display or entry rather than in the R-START routine which is called more often. This making the processing of the window more efficient.

15. See Also

WINDOW Statement
WINDOW Options
WINDOW BODY
ROUTINES SECTION