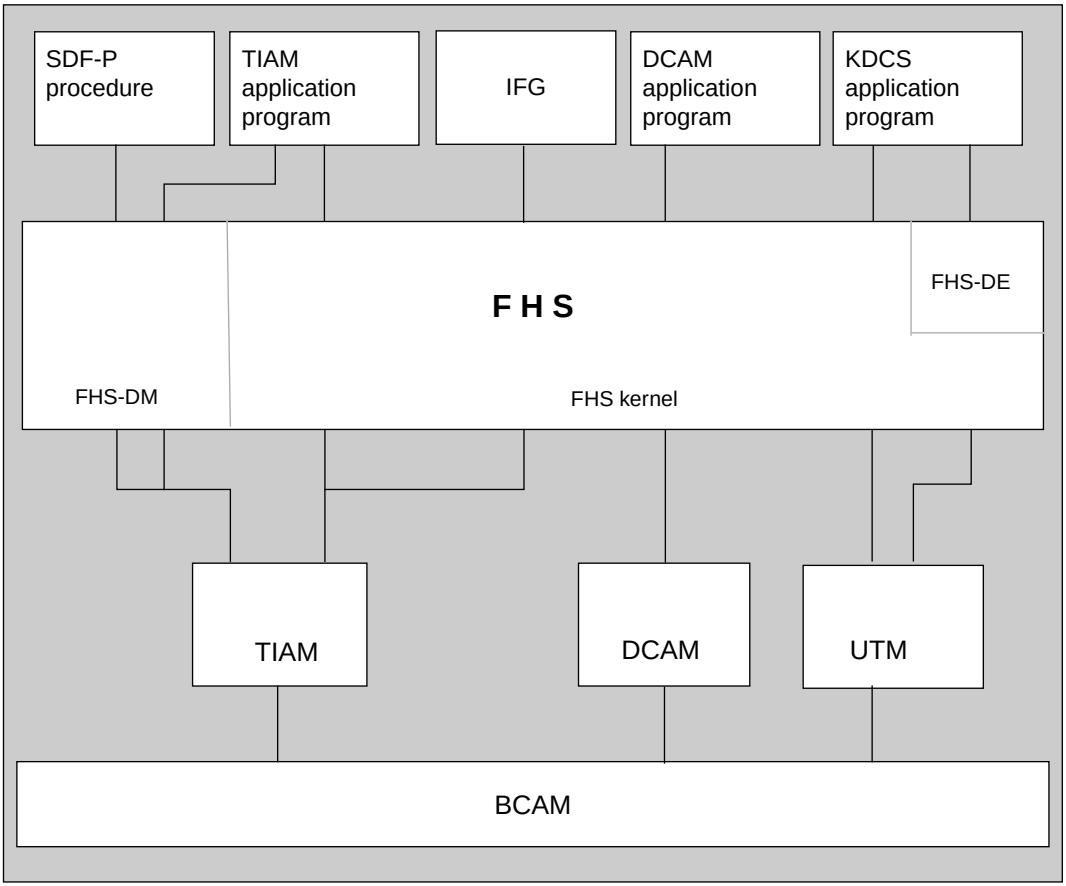
1 Preface

The **F**ormat **H**andling **S**ystem (FHS) supports the exchange of formatted messages between application programs and terminals. The use of FHS makes the application program largely independent of the physical characteristics of terminals. FHS can be used for application programs in inquiry and transaction processing and in timesharing modes. The following figure illustrates the integration of FHS in the system environment:



FHS-DM in the system environment

Components of FHS

FHS consists of three parts:

– The FHS kernel:
  The kernel handles static formatting, i.e. links program data with a format. It produces a message unit that can be printed on a screen or printer.
  Data that is received from a display terminal is edited as defined in the format and made available to the application. The kernel also includes some service modules.

– Connection modules to UTM application programs, including the dialog extension for UTM.

– FHS dialog manager:
  FHS-DM implements an independent connection between a TIAM application or SDF-P procedure and a terminal.

Application programs that use the FHS kernel and the dialog extension for UTM for formatting (TIAM, DCAM, and UTM programs) require a data transfer area to communicate with FHS. The structure of this data transfer area is defined when creating the format with the Interactive Format Generator (IFG). If the format is subsequently altered, the data transfer areas may also change, in which case the application program must be recompiled in most cases.

If data exchange is handled by the FHS dialog manager (for TIAM application programs or SDF-P procedures), data areas defined by dialog variables are used. The dialog manager connects dialog variables with the fields which are defined in the format and which were assigned a name when creating the format. A variable handler establishes the data link between the application program or SDF-P procedure and the FHS dialog manager.
As a result, the format can be changed without always needing to recompile the application program.

The recommendations given in the “SNI Alpha Style Guide” were taken into account when creating the formats and for the user interface of FHS-DE and FHS-DM.

Help and validation checks need not be programmed in the application program, but can be defined in the format and executed by FHS-DE or FHS-DM.

FHS operates with formats that are prepared in advance by using the IFG.

1.1 Target groups

This User Guide describes the dialog manager of FHS (referred to below under the name FHS-DM). It is intended for terminal users and programmers who use the TIAM interfaces for remote processing in BS2000. The earlier FHS interface is described in the manual “FHS - Formatting System for UTM, TIAM, DCAM”.

In order to understand this manual, a basic knowledge of the BS2000 operating system and of the programming language being used is required.

1.2 Summary of contents

This reference manual is arranged as follows:

- Introduction to FHS
- Introduction to dialog elements
- Data transport, validation, and editing
- Working at the terminal
- Interface to TIAM application programs
- Interface to SDF-P
- Sample programs

README file

Information on any functional changes and additions to the current product version described in this manual can be found in the product-specific README file. You will find the README file on your BS2000 computer under the file name `SYSDOC.product.version.READ-ME.E`.
The user ID under which the README file is cataloged can be obtained from your system administrator. You can view the README file using the `/SHOW-FILE` command or an editor, and print it out on a standard printer using the following command:

```
PRINT-FILE FILE-NAME=filename,LAYOUT-CONTROL=PARAMETERS(CONTROL-
CHARACTERS=EBCDIC)
```

2 Introduction to FHS

What is a format?

A format (also known as a mask) is a form displayed on the screen of a data display terminal. Just like the forms we encounter every day (e.g. application forms, order forms), a format consists of fields (boxes) in which entries can be made, and predefined texts which are part of the form itself. Such a “form” is based on a logical data structure made up of:

– fields with predefined text (text fields)

– fields in which entries can be made by the terminal user and/or the application program (variable fields)

– information on the position of these fields on the screen

– information on the characteristics of the format, e.g. the terminals on which the format can be output

– information on the characteristics (attributes) of fields in the format, e.g. underlined

– information on the editing attributes of field contents

– names of fields (names of dialog variables)

– message IDs for user messages

– links to help panels

– links to function key assignments

The following figure shows an example of a format as it is displayed on the screen.

```
                          Personal Data File
_____________________________________________________________________
PLEASE ENTER YOUR ADDRESS

Last name:   _______________
First name:_________________
Street:_____________________
ZIP:    00000
City:    ____________________
Phone: Area code: 00000   Number: 0000000

Customer No.: 0000000000
---------------------------------------------------------------------
Command:
F1 = Help   F3 = Exit   F12 = Cancel
```

Example of a format

Since FHS makes application programs independent of the physical characteristics of terminals, the user can work on different terminals without being concerned with their individual features. The user works with “virtual” terminals, while FHS provides the interface to the actual terminals.

Which terminals does FHS-DM work with?

FHS-DM V8.1 supports the 8160, 9750, 9755, 9763, and 3270 display terminals as well as equivalent devices and emulations .

For FHS to support the IBM System 3270, the software product TRANSIT-CD must be installed in the front-end processor and the terminals must be generated as system device type 3270.

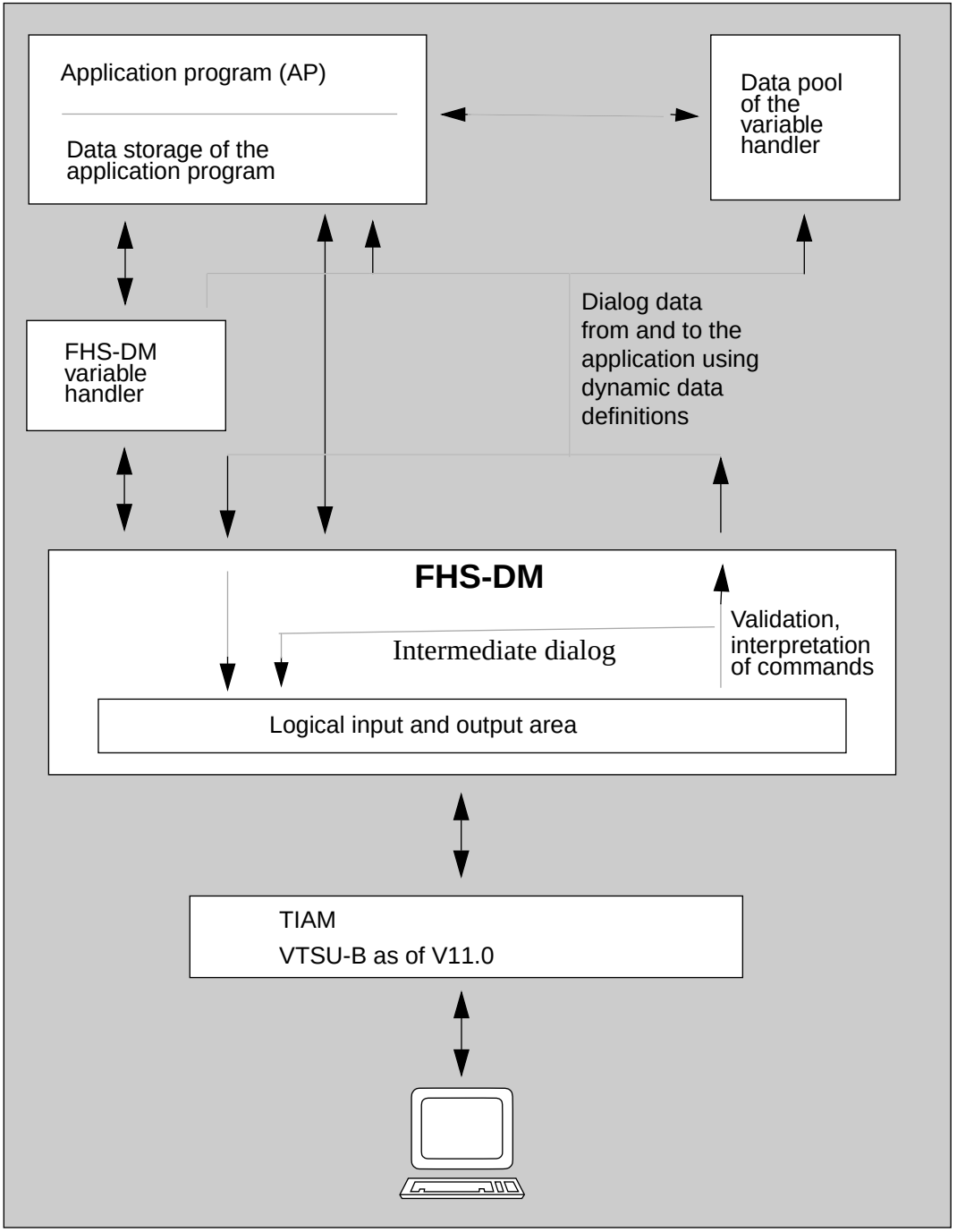
Printer terminals may be connected locally to a display terminal or centrally via a printer terminal controller.

If the terminal type is specified incorrectly in the PDN, errors may occur during formatting. The actual terminal type and the terminal type generated in the PDN must be identical.

How does FHS-DM function?

FHS-DM supports the input and output of formatted messages that are exchanged interactively (i.e. in a dialog) between the application program and terminal. It can also execute intermediate dialogs independently without a call from the application program, e.g. to issue error messages prompting the terminal user to correct invalid data entries or to display help panels.

The following figure illustrates the flow of information when working with FHS-DM.


Application program (AP)
Data storage of the application program
Data pool of the variable handler
FHS-DM variable handler
Dialog data from and to the application using dynamic data definitions
FHS-DM
Intermediate dialog
Validation, interpretation of commands
Logical input and output area
TIAM
VTSU-B as of V11.0

Variable services are used to define dialog variables in the application program. When a mask is to be output, the display service of FHS-DM is called. This service transfers the format data (constant text and the contents of variable fields) and logical output control characters to an output area, and the resulting output message in “extended line mode” is then output with VTSU-B at the appropriate device.

Input messages received by FHS-DM are always device-independent. Incoming data is associated with appropriate input fields, the entered command is then evaluated, and the input data is checked. If help was requested or invalid data was entered, a new output is initiated. Otherwise, if the command and the input data contain no errors, the data is passed to the variable handler and is thus made available to the application.

3 Introduction to the dialog extension

3.1 Structure of DE formats for the FHS dialog manager

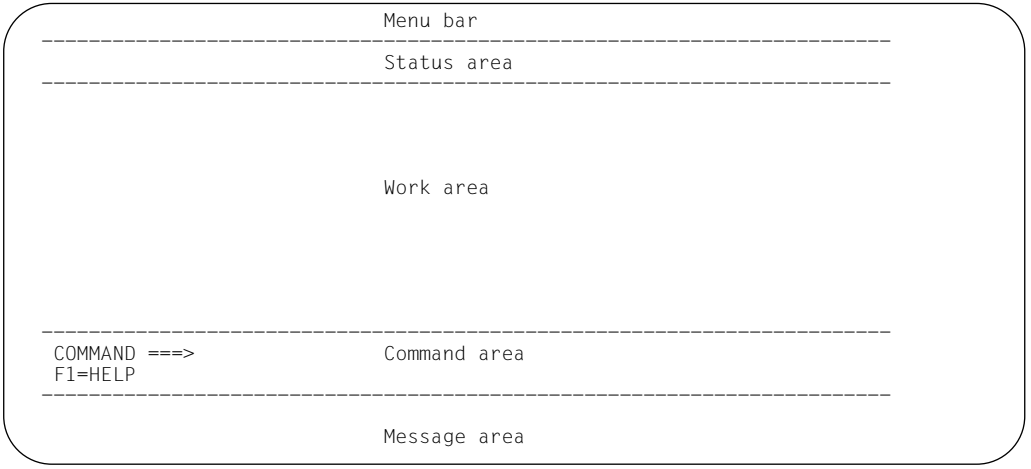
You must create DE formats using IFG V8.1 or a later version.To receive a DE format, you will need to specify the switch “Dialog extensions required? : Yes” explicitly in IFG (“User Profile/Format Display Attributes”). These DE formats can then be used with FHS-DE and FHS-DM for TIAM and for SDF-P.

There are two representations for a DE format:

– as a full format or partial format, i.e. the format occupies the entire width of the screen.

– in a box, i.e. the height and width of the format occupy only a part of the screen; see the section on “Dialog boxes” on page 13ff.

A DE format includes additional areas that are used exclusively for user prompting. It is usually composed of five areas and has the following structure:



Menu bar

The menu bar is a one-line area, delimited by a separator line, at the top margin of the screen. It contains menu titles. Each menu title represents a group of choices that are displayed in the form of a pull-down menu under that menu title (see also page 29). A menu bar is only possible in full-screen mode, not in a dialog box.

Status area

The status area contains (centered) the title of the format (or panel). The format name (PANELID) is additionally output at the beginning of the line if requested with the FHS command “PANELID ON” (see the section on “FHS commands”). The title is optional and is defined using IFG. Formats without a title do not have a status area.

Work area

The work area is the actual action area for the terminal user and contains the text fields and variable fields (as in the case of existing FHS formats). In addition, FHS-DM may include selection fields and output lists.

Command area

The command area contains the command line, as well as the display (one or two lines long) of the function key assignments. The command line consists of a text field (in the example “COMMAND ===>”) and an input field for commands known as the *command field*.

Message area

Messages are displayed in this area. The message area may only be defined for full or partial formats. Boxes do not have a message area. Long messages can be displayed in special boxes, called message boxes; see page 18.

Of all the areas described above, the work area is the only mandatory area (and must be included for reasons of compatibility with earlier versions). It is, however, strongly recommended that the DE format be defined with all the permitted areas so that the terminal user can be presented with a uniform and user-friendly screen interface.

3.2 Data exchange using dialog variables

Dialog variables enable the exchange of data between the application program, the dialog manager, and dialog elements. They assume the function of a data transfer area.
An important use of dialog variables is in the exchange of data between mask fields and application program data. The names of dialog variables are assigned to the I/O fields of a mask when defining a format with IFG,
The use of dialog variables is defined in IFG by setting the corresponding switch to “Y” in the “Application Library Specifications” mask. The same mask also allows you to set checks for the syntax of dialog variables or for SDF-P syntax.
When the dialog manager displays a mask, the values of the assigned dialog variables are shown in the mask fields. Any data that is entered in an input field of a mask is then stored as the value of the corresponding dialog variable (see also the section on “Dialog variables” on page 98.

3.3 Dialog boxes

You can implement intermediate dialogs with FHS-DM by overlaying the underlying format with dialog boxes; see the manual “Alpha Style Guide - Guidelines for the Design of Character-oriented User Interfaces”. The intermediate dialog can have multiple layers, i.e. several boxes may overlay each other on the screen.

A dialog box is a frame in the form of a “screen within a screen”. This frame, in turn, contains a format that no longer covers the entire area of the screen. You can generate the formats displayed in boxes by means of the IFG format generator.

A box contains the following:

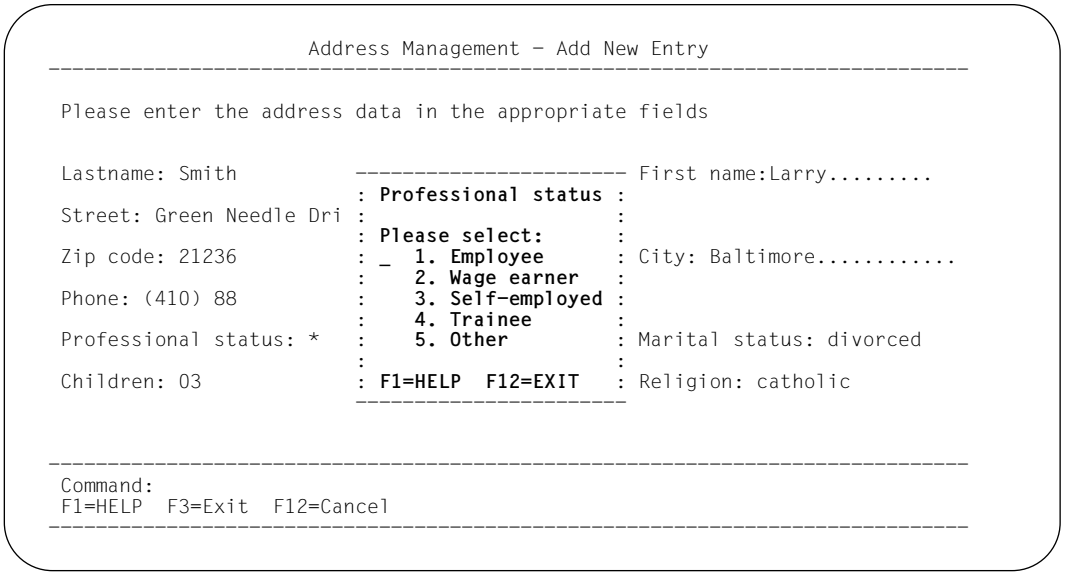
– a format with input, output, and selection fields
– messages; see the section on “Message boxes”
– help texts; see the section on “Help boxes”.

Some boxes are used solely for information purposes and require no input, e.g. certain help or message boxes. These boxes are referred to as *modeless* boxes. When these boxes are displayed, entries may still be made in the underlying areas of the screen.

Modal boxes, by contrast, expect input. Once a modal box is output, all the other sections of the screen are protected against input.

When FHS-DM is used, boxes may be output by the program or by FHS. Boxes output by the program are referred to as *explicit* boxes, while those output by FHS (independently of the program) are called *implicit* boxes. Help texts are typically output as implicit boxes, i.e. you can create a complete help system without increasing the application load.

The following is a simple example of full format with a dialog box.



In the first dialog step, the terminal user entered nothing in the field “Professional status”, which was output on the screen with the default “*”. The program unit of the second dialog step interprets the “*” character appropriately and displays the box for the selection field “Professional status”.

Entries in dialog boxes

If one or more boxes appear on the screen (as in the example), entries are only permitted in the uppermost modal box. All modal boxes/formats below it are inactive, i.e. the input fields become protected fields and are displayed accordingly on the screen. An underlying box will only be activated when all of the overlaid boxes have been removed.

If the uppermost box is modeless, e.g. a help box for a specific field, entries may be made in the underlying box/format, provided the required input fields are not completely or partly concealed. For input in concealed fields, the box must be first removed.

Removing dialog boxes

Implicit boxes are removed by using the FHS commands CANCEL and EXIT. CANCEL removes the uppermost implicit box; EXIT removes all implicit boxes; see also the section on “FHS commands” starting on page 63.

Explicit boxes can only be removed by the program units of the application.

3.3.1 Explicit boxes

The dialog manager enables application programs to output masks in dialog boxes. Such boxes are known as explicit boxes.

The position of a dialog box on the screen is defined by the application program by means of the ADDPOP service. It can be specified with reference to the top-left corner or a field name of the underlying mask.

The size of a box is determined by the size of the mask to be displayed; each mask shown in a box is additionally enclosed within a frame. The width and the maximum height of the mask are contained in the format definition. They are determined by the number of lines and the maximum width of the fixed areas (status area, command area, etc.) in the mask.

If you are creating the mask to be shown in an explicit box yourself, you should ensure that it fits in the box completely, since scrolling of the work area is not supported for explicit boxes.

If the position of a box is specified explicitly by means of a row and column, the desired position should be selected based on the size of the box so as to allow the box to be fully displayed.

When a field name is specified as the position of a box, an attempt is made to display the box below that field. If not enough space is available under the field, the box is positioned about it if possible, and otherwise to the right of the field. As far as possible, the reference field is left uncovered.

When a box is output by the application program, fields in the underlying mask are rendered inactive, i.e. the input fields of that mask become protected fields. Color or other forms of highlighting are reset.

A box output by an application program is always a modal box.

Generating and removing explicit dialog boxes

Dialog boxes generated by the application program are usually used to extend the dialog in the main window (full screen). The following general rules apply to these boxes:

– A dialog program must output a mask as a full screen before a box can be displayed.

– The box is always modal, i.e. the underlying mask is locked from input.

– The position of the box can be defined in relation to a field or the start of the underlying mask (offset positioning).

The application program must first call the ADDPOP service to initialize output to a box. The DISPLAY service must then be called to display a mask in the box. The size of the box is determined by the size of the mask to be shown (defined using IFG). The mask output (in a dialog box) initialized by the ADDPOP service remains in effect for all subsequent DISPLAY calls until a REMPOP call occurs or a further ADDPOP service is invoked.

Subsequent DISPLAY calls show masks in a box with the same starting position. The output to a box is closed by the REMPOP service; however, the box is not actually removed until the next DISPLAY call. When a box is removed, some other mask may be displayed by specifying its format name, or the underlying mask with its original field contents can be shown again (by omitting the format name). Multiple ADDPOP calls, followed by one DISPLAY call each, can be used to generate several layers of boxes (cascades).

The optional operand ALL for a REMPOP call removes all boxes up to the main window. Without it, only the topmost box is removed. If the ALL operand is not used, the ADDPOP and REMPOP calls must be paired.

The following example demonstrates, in simplified code, the output of a mask in the primary window, followed by the display of two boxes in a cascade, the removal of all boxes, and a return to the original mask in the primary window.

```
DISPLAY PANEL(PRIM)
 .
 .
ADDPOP
   DISPLAY PANEL(BOX1)
   ADDPOP
      DISPLAY PANEL(BOX2)
REMPOP ALL
DISPLAY
```

3.3.2 Implicit boxes

Implicit boxes are controlled by FHS and are used to output messages and help panels. In the case of implicit boxes, neither the name nor the fields of the format are known to the program.

Implicit boxes are output by FHS as modal or modeless boxes. For modal boxes, the underlying format is inactive, whereas for modeless boxes, it is possible to make entries if the relevant field is not concealed.

Position of implicit boxes

The position of implicit boxes is defined by FHS. Boxes that have no reference point are output in the middle of the screen. For boxes that have reference points (e.g. field-related help or messages), FHS first tries to display them by using the default shift, i.e. two lines below and two columns to the right of the reference point. If there is not enough space, FHS tries to output the box in a way that leaves the reference field completely visible. Otherwise, if the field needs to be partly or totally concealed, the reference field becomes a protected field for modeless boxes as well.

3.3.3 Message boxes

A message box is an implicit box that is generated by FHS. The height and width of the message box depend on the length of the message text. If the prescribed maximum size of the box (six lines of 56 characters each) is exceeded due to editing, the message text is transferred to the message box in unedited form.

The text of the message is highlighted, e.g. bright, reverse video, or in color, depending on the terminal and the type of message. The other areas of the box are displayed with normal intensity. The command line is only generated for modal messages.

A message may be output either by FHS (implicitly) or by the program (explicitly). See the section on “Message output” starting on page 84 for details.

Codes to edit a message may be specified when creating a message text. The output position of the help or message box is determined by FHS. If a field-related help or message text is involved, the box is output at the field for which it was activated. The actual position depends on the amount of space available above, below and on both sides of the field. An attempt is always made to ensure that the field is not covered.

3.3.4 Help boxes

A help box is a modeless implicit box. You can define the size of any application-specific help box when generating it with IFG. A help box may also occupy the entire screen.

Field-related help boxes should be smaller. As far as possible, FHS outputs them below the field; see the section on “Position of implicit boxes”. If the field remains visible, the user can read the help and fill the field at the same time.

The size of a help box is defined by the help panel. In the definition of the help panel you can specify whether a help box is to have a fixed height or whether FHS-DM may increase the height of the box depending on its position. In the latter case, the defined height is interpreted as the minimum height of the box. The message area of the screen is not overwritten when extending the box, and the width of the box is not changed.

If a help text does not fit in a box, the box will include scrolling information, e.g. “More: +”. Additional text can then be requested by using the “+” command.

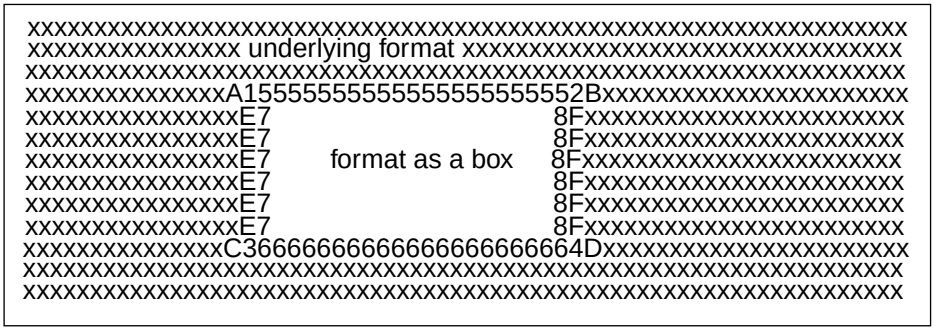
A help box is generated as a DE format with IFG. You must explicitly specify that the format is a help panel.

More information on help can be found starting on page 88.

3.3.5 Frame of a box

The frame of a box is created by default with periods (horizontal lines) and colons (vertical lines). These characters may be changed as described below in the section on “Modifying frame characters”.

FHS inserts blanks in certain places between frames and the underlying format. The following diagram shows how a format fits in a box and how the box is positioned on the underlying format.



In the above diagram, frame characters can be assigned for characters 1 - 8 and A - F. If the terminal can only represent the Alpha character set, the Alpha Style Guide recommends the use of “.” and “:” (periods and colons). If the terminal can display graphics characters, you can use the SNI product ICE (Interactive Character Set Generator) to define frame characters (see also page 22).
Characters A - F are ignored on 3270 terminals. Blanks are used by default.

The first and last column of the format in the box should contain blanks.

Defining characters for the frame of a box

Every box shown on the screen consists of a frame and the mask enclosed within it. On most Alpha screens, this frame can only be constructed using characters of the normal character set.

An application may change this default and assign other frame characters; however, an Assembler module, which on assembly produces an R module, must be coded for this purpose. The CSECT name IDHBORD is mandatory. The R module must be made available in the format library.

The macro IDHMBDR is supplied to define the frame layout.

Multiple frame definitions may be coded in the macro to define different frames for different types of terminals. The values specified for the DEV, DIM, CCSNAME, and COLORED operands are used to select a frame definition. When a box is to be output, FHS-DM compares the attributes of the current terminal with the specifications to select the frame definition. If an appropriate frame definition is found, that definition is used to display the box frame; otherwise, the default layout is generated.

If a 9763 terminal with a color screen is to be used when working with FHS-DM, the following must be observed when preparing the frame definition:

1. When specifying the ICE color character set, white must be entered as the COLOR operand.
2. The representation of color is determined by the device setting (by SIDATA). Characters in the specified color can be shown on a black background or black characters can be shown on a color background. In the latter case, inverse colors are used for the box frame (e.g. yellow becomes blue and cyan becomes red). For this reason, two color character sets for the frame definition are included in the delivery package: IDHTSD1C should be used with a black background, and IDHCSD1D with a color background.

   Note:
   The best results are obtained when cyan characters are displayed in normal intensity (SIDATA setting). White should not be specified as the „HOLE COLOR“ when defining the format.
3. If a monochrome ICE character set is defined, the box frame is displayed in black on white.

Character sets can be defined by using the SNI product “Interactive Character Set Editor (ICE)”.

Format and operands of the macro IDHMBDR

Operation	Operands
IDHMBDR	[BORDER=frame-characters] [,NEXT=name [,DEV=device] [,CCSNAME=ccsname] [,ICENAME=icename] only for 9763 terminals 1) [,DIM=24X80/27X132/32X80] only for 9763 terminals [,COLORED=YES/NO/BOTH] only for 9763 terminals [,COLOR=color] only for 9763 terminals

1) also applies to other terminals with same characteristics

BORDER=frame-characters

frame-characters: 14 characters in the following order:

- 1 top left corner
- 2 top right corner
- 3 bottom left corner
- 4 bottom right corner
- 5 upper horizontal line
- 6 lower horizontal line
- 7 left vertical line
- 8 right vertical line
- A border character for top left corner
- B border character for top right corner
- C border character for bottom left corner
- D border character for bottom right corner
- E border character for left vertical line
- F border character for right vertical line

The default characters used are C'..::..:: '.

If an ICE character set is used, these characters are the internal code.

NEXT=name
This entry must be specified if further macro definitions follow; "name" indicates the name at which the next macro definition is specified.

DEV=device-type
A value must be specified here if the frame characters are to be associated with a particular type of terminal. The entry is needed if multiple frame definitions are desired. The parameter may be omitted for the last macro definition.

9750 8160 9751 9752 9753 9754 9755 9763 3270	Type of terminal / device
XHCS	Default definition for 8-bit terminals (last 8-bit definition)
DEF	Default for all devices (only allowed in last macro call)

CCSNAME=ccsname
Only for 8-bit terminals; specifies that the frame defined with this macro definition is to be used when processing a format with the specified CCSNAME.

ICENAME=icename
Name of an ICE character set (for 7-bit formats only)

DIM=24X80/27X132/32X80
Dimensions of the screen (for 9763 terminals only)

COLORED=YES/NO/BOTH
Applies to color, monochrome, or both types of screens (for 9763 terminals only).

COLOR=color
color: BLUE, RED, MAGENTA, GREEN, CYAN, YELLOW, WHITE
Color of the frame characters if no ICE character set is specified.
If an ICE data set is specified, COLOR=WHITE must be coded (for 9763 terminals only).

The example given below shows the default module for the frame definition.The module can be found in the default format library of the software delivery package.
The following ICE character sets are supplied:

– IDHTSD1B for 9763 terminals, monochrome, screen dimensions 24x80
– IDHTSD1C for 9763 terminals, color screen, black background, 24x80
– IDHTSD1D for 9763 terminals, color screen, colored background (cyan), 24x80
– IDHTSD4B for 9763 terminals, monochrome, screen dimensions 27x132

```
IDHBORD  START
IDHBORD  RMODE ANY
IDHBORD  AMODE ANY
                     ENTRY IDHBORD
IDHBORD  CSECT
*
* Device 9763; color screen; screen dimension 24 x 80;
*        ICE-Format IDHTSD1C.
* T9763BC  IDHMBDR   NEXT=T9763B7,DEV=9763,ICENAME=IDHTSD1C,
                     BORDER=ABCDEFGHabcdef,COLOR=WHITE,COLORED=YES
*
* Device 9763; monochrome screen; screen dimension 24 x 80;
*        ICE-Format IDHTSD1B.
*
T9763B7  IDHMBDR   NEXT=T9763D4,DEV=9763,ICENAME=IDHTSD1B,
                   BORDER=ABCDEFGHabcdef
*
* Device 9763; monochrome screen; screen dimension 27 x 132;
*        ICE-Format IDHTSD4B.
*
T9763D4  IDHMBDR   NEXT=T9758B8,DEV=9763,ICENAME=IDHTSD4B,
                   BORDER=ABCDEFGHabcdef,DIM=27X132
*
* Device 9758; with CCS name
*        any CCS-name
*
T9758B8  IDHMBDR   NEXT=T9763B8,DEV=9755,CCSNAME=*ANY
*
* Device 9763; with CCS name; screen dimension 24 x 80;
*        any CCS-name        (no ICE)
*
T9763B8  IDHMBDR   NEXT=DEF,DEV=9763,CCSNAME=*ANY
*
* for all other devices
*        default: no ICE, no color, border signs '..::..::      '
*
DEF      IDHMBDR   DEV=DEFAULT
END
```

3.4 Formats with CCS names

When creating formats with IFG, you can assign a CCS name (coded character set) to a format. This enables you to change or extend your character set (see the section on “Code tables” starting on page 44). The SNI product XHCS (extended host code support) and 8-bit terminals are prerequisite to using this facility.

If formats are to be output as a box, a check is made to verify compatibility of the CCS name. The following basic rule applies to explicit and implicit boxes:

If a format that is to be displayed as a box contains a CCS name, this CCS name must already be used by the full screen. If no CCS name is specified in the format to be displayed as a box, it may be output on a screen with any CCS name. The CCS name is inherited by the box.

3.5 Selection fields

A selection field provides the terminal user with a simple way of choosing from a number of options. There are two types of selection fields:

– single-choice field: the terminal user selects *one* of several options.

– multiple-choice field: the terminal user can select *several* entries.

A selection field is part of the work area of a format and usually has a number of lines. It consists of a header, a series of entries, and one or more selection input fields.

Selection fields are created using IFG You can assign a help panel (defined when using IFG) to both the selection field and each entry.

3.5.1 Single-choice field

A single-choice field always has *one* input field in which the terminal user enters a selection character to indicate a particular choice.

The number of the desired option must be entered in the input selection field. The selection is activated on pressing the ENTER key. The input selection field may consist of one or two characters, depending on how the format was created with IFG. In accordance with the Alpha Style Guide, digits should be used for the choices offered: one digit for up to 9 choices (see below) and two digits for 10 or more. Choices that are not available in the current dialog situation can be identified as locked by the application program (indicated by an exclusion character in the mask). If the input selection field remains empty, no selection occurs.

The input selection field can be preset to a value if a particular entry is frequently used. This value can be modified by entering a number for some other choice or a space.

The dialog manager checks whether the entered value represents a valid choice. If it does, the value is passed as a dialog variable to the application program for evaluation.

Example of a single-choice field:

```
            Professional status

         Please select:
         _  1. Salaried employee
            2. Wage earner
            3. Self-employed
            *. Trainee
            5. Other

        Command:
        F1=Help F3=Exit F12=Cancel
```

The options are numbered 1 through 5 in this example. The user enters the appropriate number in the input field, which is indicated by an "_" (underscore). To select "Wage earner", for example, a 2 must be entered in this field. Choice 4 ("Trainee") is no longer available. Note that help can be requested for the available choices. The characters used for the displayed choices are specified when creating the field with IFG.

When a single-choice field is defined using IFG, a dialog variable must be assigned to each selection field. The text for the choices can also be assigned a dialog variable. The variable assigned to a choice is used to lock the corresponding entry in the display. In the following description, the variables used for the selection field and for locking are referred to as the SELVAR and LOCK variables, respectively. The LOCK variable is a control variable that is given a name as a LOCK control variable using IFG. The name of the SELVAR variable is defined as a field name in the format definition.

The LOCK variable should be an elementary dialog variable of type CHAR with a length of one byte.

If one of the choices is currently unavailable to the terminal user, the value of the corresponding dialog variable must be set to X'F0'. Such choices are identified in the display by the exclusion character "*" and are locked from selection. Any change in the value of the dialog variable will not take effect until the next DISPLAY call with the PANEL operand (see also the description of the DISPLAY service).

If no LOCK variable was allocated to a choice when defining the format or if the corresponding dialog variable does not exist, no exclusion is possible.

External and internal values:

IFG can be used to define an external value as well as an optional internal value for each selection field to be displayed.

The value entered by the terminal user in the input selection field is made available to the application program in the dialog variable assigned to the mask field. If an internal value exists for the entered value (external value), the internal value is transferred to the dialog variable instead of the value entered.

When the DISPLAY service is called, the value of the SELVAR variable is used to set the default. If an internal value was defined, the internal value must be specified, but the external value is displayed. If the value of the dialog variable is zero, and zero does not appear as a reference value, the selection field is displayed with fill characters. Fill characters are also shown if the value of the dialog variable consists of blanks or has a length of zero.

3.5.2 Multiple-choice field

A multiple-choice field consists of a title (optional) and a number of input selection fields with appropriate entries to indicate choices. The choices are not numbered, but each choice is preceded by an input selection field. The user selects an option by marking the corresponding input field.

The DISPLAY service checks whether a legal value has been specified in the input field of a multiple-choice field. This value can be a defined selection character or a blank.

Example of a multiple-choice field:

```
     Childhood illnesses


        Please enter:
          _ Mumps
          _ Whooping cough
          / German measles
          _ Scarlet fever
          / Measles
          _ TB
          * Other illnesses

         Command:
         F1=Help F3= Exit F12=Cancel
```

Mumps and Measles are childhood illnesses that were selected by the / (slash). “Other illnesses” cannot be selected. Help on the available choices can be requested.

A SELVAR variable with a unique name must be specified for each choice of a multiple-choice field when defining a format (see also “Single-choice field”).

The corresponding elementary dialog variable should be of type CHAR with a length of 1 byte. Its name is defined using IFG as a field name for the respective input field. These dialog variables can contain the following values before the mask is displayed:

Binary zero / Blank	Selectable, not prefilled
’1’	preselected
’0’	selection locked

The exclusion character “*” is output in the selection fields of locked entries when the mask is displayed. Each such field then becomes a protected field.

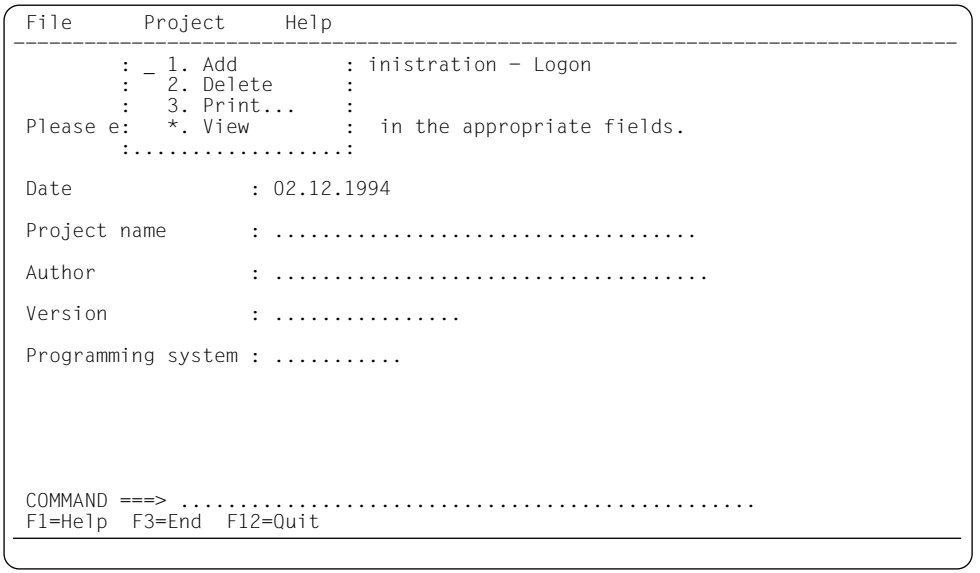
Preselected entries are shown in the mask with the selection character “/” displayed in the input selection field.

The selection characters „/“, „X“ or „x“ are accepted by the dialog manager as a selection character on input (in accordance with the SNI Alpha Style Guide).

On returning to the application program, the SELVAR variables of selected entries will have a value of 1. The application program must then respond accordingly.

3.6 Menu bar and pull-down menus

Formats may include a one-line menu bar in which menu titles are shown. A menu title can be selected by moving the cursor to that menu title and pressing the Enter key.



Positioning the cursor in the menu bar

One method of placing the cursor in the menu bar to press the F key that was assigned to the system command ACTIONS (F10 by default).
The ACTIONS key initiates cursor positioning by FHS-DM. The current position of the cursor is saved, and the cursor is moved to the first character of the first menu title of the menu bar. When the ACTIONS key is pressed again, the cursor is reset to the saved initial position.

Another method is to use the cursor keys. Since this function is implemented by hardware, the cursor position cannot be saved by FHS-DM in this case.

Selecting a menu title

Within the menu bar, you select a menu title by moving the cursor with the Tab key or the cursor keys to a character of the desired menu title and by pressing the Enter key. This causes a pull-down menu for that menu title to be displayed. During the period that the pull-down menu is shown, the cursor is positioned within the choice fields of that menu, and all

fields of the underlying mask are locked (i.e. cannot accept inputs).
Markers or inputs in the menu title are ignored, i.e. cannot be used to select another pull-down menu.

Displaying another pull-down menu

You can switch to another pull-down menu by selecting some other menu title with the cursor keys.

The CANCEL key can also be used to return to the menu bar when a pull-down menu is displayed, but the pull-down menu is deleted in this case, and any selection entered in it is ignored. The cursor is positioned at the first menu title in the menu bar, and any of the menu titles may be then selected.

Canceling a pull-down menu and exiting the menu bar

If the ACTIONS key was used to enter the menu bar, you can cancel the displayed pull-down menu by pressing the same key again. The pull-down menu is then deleted, and the cursor is reset to the saved position in the work area. Any selection made in the pull-down menu will be ignored. The CANCEL key may also be used to cancel the display of a pull-down menu. In this case, however, the cursor is moved to the first menu title of the menu bar, and you can then exit the menu bar by using the cursor keys.

> **Warning**:
> If you create your own key assignment table (key list), you must define an ACTIONS or CANCEL key for a format with a menu bar!
> Reason: a pull-down menu is not assigned a separate key, and no command can be entered in the command input field when it is displayed. In other words, it is not possible to exit the pull-down menu without the ACTIONS or CANCEL key.

Working with pull-down menus

Pull-down menus are only possible in full-screen mode. The menu bar and the pull-down menus are generated using the Interactive Format Generator IFG. The menu titles in the menu bar are markable input fields. When the cursor is in a pull-down menu, a selection can be activated by entering a choice in the selection input field or by using the default value and pressing the Enter key. Inputs and markers in menu titles are ignored; selection occurs solely on the basis of the cursor position.

The entries made in the pull-down menu are transferred to the input fields. These entries do not occur if the ACTIONS command was given following input in a pull-down menu or if some other pull-down menu was selected.

One single-choice selection field can be defined in each pull-down menu. This field can be preset to a default value, and choices can be locked. Furthermore, internal values may be defined (see also page 27).

Processing a menu bar

When a menu bar is defined with the aid of IFG, a name of an elementary dialog variable must be assigned for each menu title. This variable is called the CHECK variable. In addition, each menu title must be assigned a single-choice selection field that presents the content of the pull-down menu. The definition of the selection field for a menu title is subject to the same conditions as for any selection field.

Within the context of processing the menu bar by the application program, an active pull-down box can be considered an additional selection field of the format. The CHECK variable is used to detect which menu title was selected. If this variable does not exist, it is created implicitly as a variable of type CHAR.

All CHECK variables are set to the value 0 (C'0') by the application program before the DISPLAY service is called.

On return from the DISPLAY service, the CHECK variable of the selected title contains the value 1 (C'1').

Entries selected in pull-down boxes are processed in the same way as conventional single-choice selection fields (see also "Single-choice selection field").

3.7 List processing

Definition of a list area

The list display outputs one or more arrays of dialog variables as a list within a list area. A list area is a scrollable portion of the work area of a format.

The list display is implemented by means of the DISPLAY service. The format to be shown must contain a list area.

A format may only contain one list area, and only blanks are permitted to the left and right of that list area within the work area.

A list area consists of the following elements, which are defined using IFG:

– list title with list position and scrolling information
– column title
– list records (also called “list lines”) and list fields
– end of data marker (termination line)

List records (lines)

A list line is declared as a model line when defining a list area. The model line may consist of input, output, and text fields. Each input, output, and text field of the model line represents a list field. All list lines are output by the DISPLAY service in accordance with the model line.

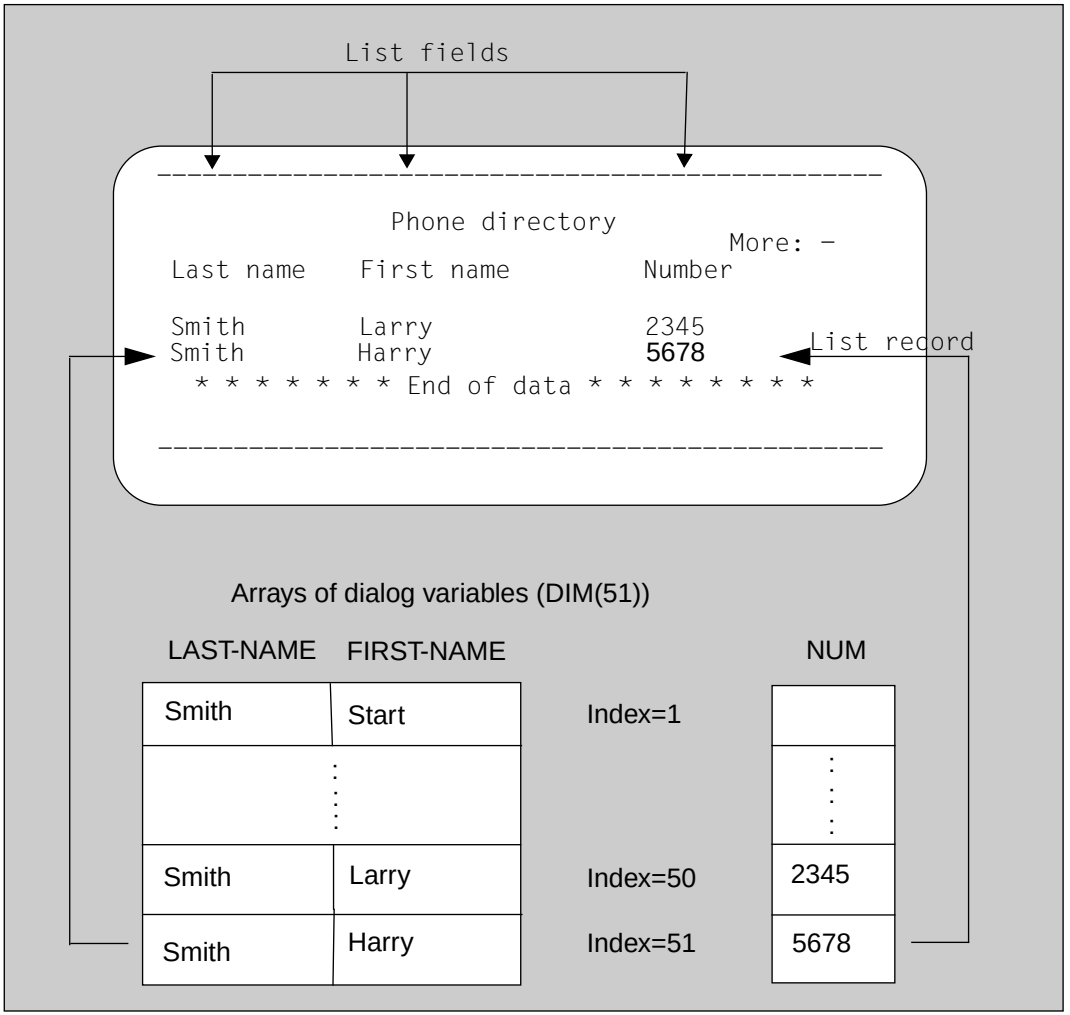
If the name of a dialog variable that is part of an array of dialog variables is assigned to an input or output field of a list, the values of that variable make up the list lines for that list field (see diagram on next page). The dialog variables with the same index provide the data for each individual list line (see also section on “Types of dialog variables”). The names of the dialog variables are declared in the model line without an index specification (e.g. #3).

Specific conventions apply in the model line for the dialog variables of a TIAM or SDF-P application program. These conventions are described on pages 108 (TIAM) and 187 (SDF-P).

If an elementary dialog variable is assigned to an output field in the model line, that field will have the same value in each list line.

Text fields have the same value in each list line as in the model line. Input fields can be modified by the terminal user of the application via entries from the keyboard. On exiting the DISPLAY service, the values of the modified input fields will be available in the corresponding dialog variables.

The following diagram illustrates the relationship between an array of dialog variables and a list field of a list line. LAST NAME, FIRST NAME, and NUM are the names of the dialog variables that were declared in the model line for the list fields. Note that the index entry always references the array element although it is also designated as the line number below.



Each list record can occupy one line of the list area or be split into multiple lines (multi-line list records). If desired, a separator line can be defined in the model line to separate the list records. The separator line will then be displayed after each set of entries that belong to the record.

The following examples show different ways of arranging list lines (records).

Example of a list with single-line records

Last name	First name	Phone	ZIP	City
Doe	John	281-6506	22181	Vienna, VA
Doe	Mary	221-6509	22181	Vienna, VA

Example of a list with multi-line records:

```
Last name   Doe                 First name    : John
Telephone   : 281-6506
ZIP         : 22181             City          : Vienna, VA
-------------------------------------------------------------------------------------------
Last name   : Doe               First name    : Mary
Phone       : 221-6509
ZIP         : 22181             City          : Vienna, VA
-------------------------------------------------------------------------------------------
```

Scrolling in a list area

If the list contains more lines than can be simultaneously displayed in the list area, scrolling commands can be given to display the invisible parts.

Scrolling can be achieved by entering a command in the command input area or by pressing an appropriate function key. The DISPLAY service supports forward and backward scrolling. The following commands are available for this purpose:

FORWARD [num] / +[num]	Scroll toward end of list
BACKWARD [num] / -[num]	Scroll toward start of list
++	Scroll to end of list
--	Scroll to start of list

The following options are available when using these commands:

– If the cursor is not in a list line

FORWARD/+	Scrolls the lowest list line to the top
FORWARD *num*	Scrolls *num* lines forward toward the end of the list with reference to the topmost list line
+*num*	Scrolls *num* lines forward toward the end of the list with reference to the topmost list line
BACKWARD	The topmost list line is scrolled to the bottom
BACKWARD *num*	Scrolls *num* lines backward toward the start of the list with reference to the topmost list line
-*num*	Scrolls *num* lines backward toward the end of the list with reference to the topmost list line

– If the cursor is not in a list line:

FORWARD/+	The list line marked by the cursor becomes the topmost line
BACKWARD/-	The list line marked by the cursor becomes the lowest line

Note: If the cursor is positioned in the first or last line of the list, scrolling occurs as if the cursor were not in a list line.

Before a scrolling command is executed, the values of modified list fields are checked and stored in the appropriate dialog variables. Mask fields that do not form part of the list area are only checked when there are no more scrolling commands. Furthermore, these field contents are only stored in dialog variables on returning to the application. If the CANCEL command is issued, the contents of non-list fields are not stored in dialog variables either.

Other definitions for a list

Special dialog variables to control the list display (called control variables) can be defined when creating the format with IFG via the “control variables” field) and associated with the list. The names of these dialog variables can be freely selected at the time of designing the format.

The designations of the control variables described below are only used to identify them; the actual names may be freely selected. These dialog variables serve the following purposes:

- NUMROW variable

  An elementary explicit or implicit dialog variable, preferably of type FIXEDS(TIAM) / INTEGER(SDF-P). Its value defines the current maximum index for an array of dialog variables. This value must be less than or equal to the dimension (DIM operand) of the dialog variable assigned to a list field. If the value is greater than the dimension, the value of the smallest dimension is used. This entry allows an application program to partially fill the space defined with VDEFINE for the dialog variables of the list fields or to partially process an SDF-P list.

  If no NUMROW variable is specified in the format definition or if the declared dialog variable does not exist, the smallest value of the dimension of the dialog variables assigned to list fields is used instead.

- TOPINDEX variable

  Specifies the index (line number) of the dialog variable to be displayed as the first line in the list area.

  The TOPINDEX variable can be an elementary implicit or explicit dialog variable (explicit variables should be preferably of type FIXEDS / INTEGER). Its value must be a positive number and must not exceed 32767. An invalid value leads to an error. If the value for TOPINDEX is greater than the value for NUMROW or the dimension (the smallest dimension for multiple arrays of list data), the smallest value of these specifications is used as the TOPINDEX.

  Rules for TOPINDEX:

  – The DISPLAY service or the ADDPOP service terminates with an error code if the TOPINDEX variable exists and an index is specified that cannot be displayed in the list area under the conditions described below, given the value of the TOPINDEX variable.

    For example, if TOPINDEX has the value 11 and the DISPLAY service is called with an index of 15 in the MSGLOC operand (e.g. MSGLOC(FIELD3#15)), the message box will be placed on the fifth list line in the list area. If a maximum of 10 list lines can be shown in the list area, the index entry for the MSGLOC operand in the above example may have the values 11 to 21.

  – If the TOPINDEX is 0, an index of 1 is assumed for the first line on executing a DISPLAY call with the PANEL operand. Without the PANEL operand, the value of the first line in the current list area is set as the TOPINDEX.

  – If no TOPINDEX variable is specified in the format definition or if the declared dialog variable does not exist, a value of 1 is assumed for it at the start of the list displayed. If a TOPINDEX variable was not assigned in the format definition, the dialog manager cannot return the index of the first line in the list area to the application program. (The variable is not created automatically.)

    If an index is specified in this case by the CURSOR or MSGLOC operand in a DISPLAY call with the PANEL operand or by the POPLOC operand in an ADDPOP call, scrolling occurs automatically.

  – The value of the TOPINDEX variable is not taken into account when a DISPLAY call with no PANEL operand is executed for a list. The list area is shown in its existing state at the time the previous DISPLAY call was exited. Conflicts with an index entry in the CURSOR or MSGLOC operand do not result in an error. The dialog manager selects a representable form in such cases.

- MODINDEX variable

  An array of dialog variables of type FIXEDS (2 or 4 bytes) containing the indices of dialog variables of modified list lines.

  On calling the display service, the first element normally contains the value 0. Upon return from the DISPLAY service, the elements contain the indices of the modified list lines (the first list line has an index of 1). The sequence of indices ends with the first element containing a 0 or at the end of the array.

  If the lines with the indices 1, 3, and 51 were modified by the user in the list display, for example, the first element of the MODINDEX variable will contain the value 1 on return from the DISPLAY service; the second element will have the value 3, the third element the value 51, and the fourth element the value 0 (as an end marker). The dimension of the MODINDEX variable determines the maximum number of lines that can be reported as modified by the dialog manager. The application program can also preset the MODINDEX variable and then call the DISPLAY service. In this case, the corresponding list lines are predefined as modified. This can be used by the application program in combination with the “mandatory input” specification for a list field (defined using IFG) to force input for a list field (see also “Checks for mandatory input in list fields”).

  The values of the MODINDEX variables need not be stored in sorted order on calling the display service. The indices are always sorted in ascending order on return from the DISPLAY call.

  If no MODINDEX variable is specified in the format definition or if the declared dialog variable does not exist, the function associated with it cannot be used.

Calling the list display

The list display is implemented by a call to the DISPLAY service. The effects of a DISPLAY call differ depending on whether or not a PANEL operand is specified.

DISPLAY call with PANEL operand:

The following values are used by the DISPLAY service to display a list in the list area:

– The TOPINDEX variable defines the index of the elements in the list fields whose values appear in the first line of the list area.
If the value of the index is invalid (too high or negative), the DISPLAY service terminates with an error code.

– The remaining lines of the list area contain the values of elements with indices incremented by 1 from the top index. If no further elements exist, the “end of data” line if specified, is shown. Any free space that may remain in the list area is left empty.

– The position of the cursor in the displayed mask is determined by the CURSOR and CSRPOS operands of the DISPLAY service.
If the CURSOR operand specifies the name of a dialog variable associated with a list field, the cursor is positioned in that field. The precise line of the list area in which it is positioned can be defined by specifying an index for the field name (e.g. CURSOR(FIELD#3)). If no index is present, the topmost list line in the list area is assumed. The CSRPOS operand defines the position of the cursor within the list field.

DISPLAY call with no PANEL operand:

The value of the TOPINDEX variable is ignored for a DISPLAY call with no PANEL operand. The list area is shown in the same state in which the previous DISPLAY call was exited. Conflicts with an index entry in the CURSOR or MSGLOC operand do not result in an error; the dialog manager selects an appropriate form of representation in such cases.

Checking list fields

A validation check for input data can be defined for input list fields when creating a format using IFG. If such a check has been defined for a list field, every modified input field in the list line is checked.

The validation checks are performed by the dialog manager on exiting the format upon completion (not for the CANCEL command) and when a scrolling command is executed (for navigation within the list area).

If an error is detected, a message is output, and the terminal user can then correct his or her entries. The scrolling command or the command to terminate processing of the mask must be repeated. List fields with the “mandatory input” attribute are subject to special checks.

Checking list fields for mandatory input

Checks for mandatory input are only performed for modified list lines. A list line is considered modified if at least one input field in it was modified by the terminal user via a keyboard entry or if the line was predefined as modified by means of the MODINDEX variable.

To output a list containing a field for which a data entry from the terminal user is mandatory, for example, the application programmer can proceed as follows:

To begin with, the check for “mandatory input” must be defined for this field in the model line in the format definition. On calling the DISPLAY service, the MODINDEX variable is supplied with all indices of the dialog variable array (from 1 to the max. index). The list display is implemented as follows: when the list is shown, the terminal user can begin entering the missing data in the list lines. The user is allowed to scroll (forward or backward) within the list without being requested for input for fields in other lines that have not yet been filled. The modified data in the list area is written to the corresponding dialog variables before scrolling. It is only on completion of the display (i.e. when the ENTER key is pressed) that the dialog manager checks whether all fields marked for mandatory input have been changed in the lines that were predefined as modified. If some unaltered fields still exist, a message is output by the dialog manager instructing the terminal user to enter the missing data. The first line for which data is still required is shown as the first line in the list area. Scrolling is possible in this case as well. The check for compliance with the “mandatory input“ attribute occurs, as described earlier, on exiting the display (but not with the CANCEL command).

Exiting a list display

A list display is exited when the mask containing the list area has been completely processed.

Processing of the mask does not terminate on executing a command to scroll the list lines.

Return from the list display:

The following values are supplied by the DISPLAY service on completion of a mask containing a list area:

– The TOPINDEX variable returns the index of the dialog variables that are shown in the topmost line of the list area. This value may differ from the one passed in the DISPLAY call as a result of scrolling commands.

– The system variables SYS-CURSOR-FIELD, SYS-CURSOR-INDEX and SYS-CURSOR-POS contain the position of the cursor. The value of SYS-CURSOR-INDEX is non-zero only if SYS-CURSOR-FIELD contains the name of a list field. The value is the index of the dialog variable associated with this field.

– The indices of list lines in which an input was made or which were preset with defaults are supplied in the elements of the MODINDEX variable. The sequence of these indices ends with the value 0 or with the last element of the MODINDEX variable. They are sorted in ascending order. If no inputs have occurred, the first element will already have a value of 0.

– The values of the modified list fields are entered in the dialog variables.

3.8 Data editing and checking

The dialog manager can edit and check the field contents of a format in accordance with certain defaults. How the field contents are to be edited and checked is defined during format generation using IFG (see the IFG manual).

When a format is output (via a call to the DISPLAY service), data is read from the dialog variable associated with each respective mask field and converted into its external representation, taking the data editing attributes defined for that mask field into account. In addition, the output data is checked for compliance with the attributes defined for the mask field (e.g. numeric data for an arithmetic field). If an error is detected (due to incompatibility between the data type of the mask field and that of the dialog variable, for example), it is advisable to output an error mask before returning to the application program with a corresponding error code.

As far as converting the internal form into the external representation (and vice versa) is concerned, the application programmer should note that every dialog variable has a data type (see also the VDEFINE service). On the other hand, attributes are also defined for a mask field when defining a format. Consequently, compatibility between the data type of the dialog variable and the attributes of the mask field is a factor to be considered.

When an input is made in the mask (i.e. the terminal user presses an input key) or a scrolling command is given for a list area, the contents of mask fields are checked in accordance with the attributes defined for each such field, and if an error is detected, the dialog manager outputs a message box with an appropriate message. The user must then correct the field in question. Control is not returned to the application until valid data has been entered or the display is aborted. The input data is converted from its external representation to the internal format and is made available in the dialog variable corresponding to the mask field on return to the application program.

If the display is aborted with the CANCEL command, no checking of input data occurs.

The following section shows which attributes for mask fields are supported by the dialog manager. The meanings of the individual attributes are explained in detail in the IFG manual.

Editing attributes

The following options are available for editing mask fields:

- decimal separator
- digit separator
- representation of date
- representation of time
- alignment of the contents of a mask field
- fill characters
- zero suppression
- null-value representation as an empty field
- floating sign
- conversion to uppercase

Input validation and checking attributes

This includes the following input attributes:

- mandatory input
- minimum input length
- automatic input
- markable/non-markable field
- protected/unprotected field
- hardware-supported NUM lock

The following checks are available for field contents:

- any string
- alphabetic string
- arithmetic string
  - with or without decimal places
  - with or without sign
  - with or without digit grouping
- date
- time
- value list
- value range
- character list

3.8.1 Code tables

Code tables are used for editing and checking data in FSH-DM. The code tables are generated from tables of the XHCS table record.

The following tables are generated:

– table for converting lowercase letters to uppercase
– table for characters that can be represented
– table of alphabetic characters
– table for numeric characters

When creating a format with IFG, you can allocate a table record by means of a CCS name. The creation and modification of these tables are explained in the manual “XHCS Extended Host Code Support”.

A CCS name can also be specified via the CONTROL service of FHS-DM or may be defined by the terminal.

If no CCS name was specified for a format or if the named CCS is not available at the XHCS host, then tables corresponding to the CCS name EDF03IRV or the CCS name EDF03DRV, (if a German keyboard is detected) are used.

3.8.2 Data editing

The following specifications can be defined globally for all formats:

Decimal separator

The permitted characters are ".", "," and ' '. The decimal separator and digit separator must not be the same. Decimal separators are used to indicate decimal places for numbers in arithmetic mask fields (e.g. 123.45). The decimal separator is not transferred to the dialog variable.

Digit separators

The permitted characters are ".", "," and ' '. The digit separator and decimal separator must not be the same. Digit separators are used to group numbers preceding the decimal point in arithmetic mask fields (e.g. 123,456.78). The digit separator is not transferred to the dialog variable.

Representation of date

The following specifications must be given to represent the date in a mask field:

– order of day, month, and year
– separator for date components
– representation of year using two or four digits

Any combination may be specified for the order of the day, month, and year.

All characters except for the digits 0-9 are permitted as the separator. The separator delimits the day, month, and year (e.g. 21.10.1994).

Representation of time

The following editing specifications are required for representation of time:

– time separator
– representation of time with or without seconds

All characters except for the digits 0-9 are permitted as the separator.

The time separator is used to delimit hours, minutes, and seconds when representing the time (e.g. 12:24:30).

The following editing attributes can be specified for each mask field:

Alignment and fill character on output

When defining a format using IFG, you can specify if and how data is to be aligned on output of a dialog variable to a mask field and can also define which character is to be used as a fill character for free positions in the mask field.

The specification of an alignment and a fill character is not supported for the contents of dialog variables, since fixed values are used for them by the dialog manager based on the data type of the dialog variables. The fill character defined using IFG is not used by FHS-DM. Note that the term “fill character” is used in the following text to refer to IFG output fill characters only.

Alignment for a mask field:

The type of alignment specifies how the value of a dialog variable is to be justified in the mask field when it is displayed (output) and also defines how fill characters in a mask field are to be handled on input.

The following types of alignment can be set in the format definition for any alphabetic string:

– Alignment to the left
– Alignment to the right
– No alignment

Arithmetic strings are always right-aligned.

Alignment of the mask field occurs when the relevant length of the data of a dialog variable is less than the length of the mask field. If the relevant data of the dialog variable is longer than the mask field, the behavior of the DISPLAY service depends on whether a mask field for input or output is involved.

In the case of an output field, the data is truncated, and processing continues with a warning. By contrast, if an input field is too short, the DISPLAY service terminates with an error code. The left or right alignment defines how the contents of the mask field are to be aligned by the dialog manager when displayed on the screen. A left or right alignment means that the data of a dialog variable of type CHAR or BINSTR is displayed without leading blanks. If “no alignment” is specified, the leading blanks are shown, and the data is aligned to the left.

Output fill characters:

If the data of a dialog variable does not fill the entire mask field, the output fill characters using IFG are entered in the remaining positions in the mask field. If the dialog variable is empty, the entire mask field will contain only fill characters when displayed.

On output to a mask field, the field is first filled with the aligned data and then with fill characters.

Fill character handling on input

The terminal user can enter data into a mask field without being concerned about the alignment that was defined for it. When the dialog manager processes the input data, the relevant data and its length are determined. Leading and trailing fill characters in alphabetic or any other strings are handled as follows:

– Leading and trailing fill characters, blanks, and NULL characters are removed for left or right-aligned mask fields.

– For a mask field that is not aligned, trailing fill characters, blanks, and NULL characters are removed; leading fill characters, blanks, and NULL characters are converted to blanks and stored.

– If the input field contains only blanks, fill characters or NULL characters, the resulting length of the relevant data is zero (null value).

– Fill characters, blanks, or NULL characters enclosed within the string are never removed.

In the case of an arithmetic mask field, leading and trailing fill characters are handled as follows when determining the relevant data:

– The mask field is always treated as right-aligned. If the fill character is not zero, leading and trailing fill character and NULL characters are removed.

– Leading zeros are always removed even if the fill character is zero.

– If the input field contains only blanks, fill characters or NULL characters, the resulting length of the relevant data is zero.

Leading and trailing fill characters are removed for a command input field regardless of the type of alignment.

After input handling, the relevant data is stored in the corresponding dialog variable in accordance with the alignment that applies to that variable based on its data type and with a length based on the determined relevant data. Explicit dialog variables are subsequently padded with default fill characters if the data length is less than the defined length of the dialog variable. Implicit dialog variables are always stored with their relevant length (except when the length is zero).

Options for fill characters and alignment:

Mask field	Output fill character	Alignment
Any string	Any character	right, left, or none
Alphabetic string	Any character	right, left, or none
Arithmetic string	No digits from 1 to 9, no sign	right (always)
Date	No digits from 1 to 9	not possible
Time	No digits from 1 to 9	not possible

Dialog variable	Input fill character	Alignment
CHAR / STRING	blank	left
BINSTR	X’00’	left
NUMS	X’F0’	right
NUMU	X’F0’	right
FIXEDS / INTEGER	X’00’	right
FIXEDU	X’00’	right
PACK	X’00’	right

The data types are defined under the VDEFINE service.

3.8.3 Input attributes

Minimum input length

The minimum input length defines the least number of characters that must be entered into a field on input.

Mandatory input

Mandatory input for a mask field means that the terminal user must modify that field via a keyboard entry. If this requirement is not fulfilled, the dialog manager will prompt the user for an input by means of a message. If the mandatory input requirement for a field has been satisfied by the terminal user, the fact that an input has occurred is registered internally so no prompt for an input in that field is given in a following DISPLAY call without the PANEL operand or if the same format name is specified again.

If the mandatory input attribute is to be activated again in a subsequent display of the same format, an ATTR service must be called with TYPE(MANDATORY) for that field before the DISPLAY call or, alternatively, the MANDATORY operand must be specified in the DISPLAY call. Mandatory operands for list fields are subject to special handling (see “list processing”).

3.8.4 Validation checks for the contents for mask fields

Data is checked and edited on input and output as described below.

Any string

This means that the mask field may contain any characters. No validation checks and editing are performed by the dialog manager. Non-printable characters (other than NULL) are represented by SUB (X'3F').

Alphabetic string

The mask field may contain only letters included in the code tables and blanks.

Arithmetic string

In order to use the editing attributes of an arithmetic mask field meaningfully, the corresponding dialog variable must be of type NUMS, NUMU, FIXEDS, FIXEDU or PACK. An arithmetic mask field must not contain more than 15 digit positions. The number of digit positions is calculated from the field length minus the number of positions for the sign, decimal separator, and digit separator if the corresponding attributes were defined.

General input validation check:

The field may only include digits (0-9) and, depending on the set attributes for decimal places, digit grouping, and sign, the characters ",", ".", " ", "+" , and "-" at certain positions. Blanks, followed by output fill characters before the number, or output fill characters, followed by blanks after the number produce an error if the output fill character is not a blank.

Output validation check:

The dialog variable must be of type FIXEDS, FIXEDU, PACK, NUMS or NUMU and contain type-compatible data.

Arithmetic string with decimal places

The number of decimal places can be defined for a mask field.

Input validation check:

A maximum of n digits (defined using IFG) may follow the decimal separator (zeros on the right are ignored).

Input editing:

The entered decimal places are transferred to the extreme left position for decimal places and padded with “0”.

Output editing:

n digits on the extreme right are interpreted as decimal places and output to the right of the defined decimal separator.

Arithmetic string with sign

This means that a sign may be specified for the field. The sign can be defined as a floating sign.

Input validation check:

If no sign is allowed, neither “+” nor “-” may appear in the mask field. If a sign is permitted, it may be placed to the left or right of the input. Blanks are permitted between the sign and the number. The use of only a sign in the mask field (without digits) is illegal.

Output validation check:

The dialog variable must be of type FIXEDS, NUMS or PACK.

Input editing:

The “-” sign is set at the last position of a NUMS dialog variable; a “+” or no sign results in a “+” at the last position.

Mask field		Dialog variable
Two decimal places		Type NUMS
-123	--------->	0012300-

Output editing:

A positive sign is represented as a blank (“ ”), a negative sign as “-”.

Depending on whether or not the sign was specified as “floating”, it will either be placed in the last position of the mask field or before the number. The remaining positions on the left are either filled with output fill characters or with blanks (for a “floating sign”).

Dialog variable		Mask field
Type NUMS		Fixed sign; output fill character '$'; two decimal places
0012300-	--------->	$$$123,00-
		Floating sign; output fill characters are always blanks
0012300-	--------->	-123,00

Arithmetic string with digit grouping

Digit separators set off positions before the decimal point in groups of three digits each.

Input validation check:

The positions before the decimal point can be split in sequence from right to left by the digit separator into groups of 3 digits each. Leading and continuous zeros are ignored. If a digit separator appears in the field, all positions before the decimal point must be correctly grouped.

Input editing:

The digit separators are not transferred.

Mask field		Dialog variable
Digit grouping; two decimal places		Type NUMS
-1,234	--------->	0123400

Output editing:

Digit separators are inserted at the appropriate positions: the positions before the decimal point are transferred to the field, and a digit separator is inserted before each third digit if it is followed by a further digit. (If zero suppression is not allowed, a digit separator can also appear at the first position of the field.)

Dialog variable		Mask field
0123400+	--------->	1,234.00

Arithmetic string with zero suppression

Leading zeros are omitted.

Input validation check:

Any number of fill characters may precede the number.

A leading zero before the decimal separator may be omitted; an input such as ".5"instead of "0.5" is legal.

Input editing:

Fill characters are not transferred.

Output editing:

If zero suppression is allowed, all leading zeros in pre-decimal positions are replaced by output fill characters; otherwise, all leading zeros are retained.

Exception:
If there is no remaining space before the decimal separator, no leading zero is placed before it.

Dialog variable		Mask field
Type NUMS		Fixed sign; output fill character;'$'; two decimal places; zero suppression
0000012-	--------->	$$$$$0.12-
		No zero suppression
0000012-	--------->	000000.12-

Value zero as blank:

The application programmer can use IFG to define whether the value zero of a numeric dialog variable can be represented as a blank in an arithmetic mask field. Only blanks are displayed in this case.

Editing of arithmetic mask fields

Output editing occurs as shown below (always with right alignment):

Digit grouping not allowed:	
Fixed sign	[F...][9...]9[D9[9...]][-]
Floating sign	[b...][-][9...]9[D9[9...]]

Digit grouping allowed:	
Fixed sign	[F...][9[9]]9[S999...][D9[9...]][-]
Floating sign	[b...][-][9[9]]9[S999...][D9[9...]]

Examples of values in an arithmetic mask field:

```
  123.5
1,234.5
   -123
 123.4-
 000123
 000.12
   0.12
```

Inputs in an arithmetic mask field are possible in the following formats:

Digit grouping not allowed:
[F...][b/n...][+/-][b...]][9...][D[9...]][b/n...][F...] or [F...][b/n...][9...][D[9...]][b...][+/-][b/n...][F...]

Digit grouping allowed:
[F...][b/n...][+/-][b/n...][9[9[9]]][S999...][D[9...]][b/n...][F...] or [F...][b/n...][9[9[9]]][S999...][D[9...]][b...][+/-][b/n...][F...]

Digit grouping, if allowed, is not mandatory and may be omitted on input. The following codes are used in the format above:

- n NULL (X'00')
- b Blank
- 9 Digits 0-9
- S Digit separator
- D Decimal separator
- F Output fill character

Examples of valid input data (no alignment required):

```
  .5
1.234,5
  1.234,
.234
+ 1234
 123-
```

Examples of invalid input data:

1,2345
0.123
2,123,
123,

Reasons: digit grouping must occur in groups of three; too many decimal places (two were defined); invalid separator (as per definition).

Date

The mask field contains a date entry. The dialog variable for the date must be of type CHAR and have a length of 12 or 14 bytes.

Input validation check:

The field may only include the digits 0 to9, blanks, and two occurrences of the separator for the specified date. Leading zeros may be omitted for the day, month, and two-digit entry for the year. The separator may be preceded and followed by blanks. The presence of blanks followed by output fill characters before the date, or of output fill characters followed by blanks after the date leads to an error if the output fill character is not a blank. Blanks are allowed at the start or end of the date (regardless of the output fill character).

When the year is specified by two digits, the check for a valid date is based on the time period from 1901 to 2099. For a four-digit year entry, the date may only lie within the period from 10-15-1582 (start of the Gregorian Calendar) up to 12-31-2099.

Output validation check:

The dialog variable must contain a valid date in ISO or ISO4 format. The accuracy of the current day and the last position (blank) are not verified. The (internal) representation of the date in the dialog variable corresponds to that of the GDATE/GTIME macro (see the manual “BS2000 Executive Macros”).

ISO format:

0	2	3	5	6	8	11
y y	X’60’	m m	X’60’	d d	n n n	X’40’
Year	-	Month	-	Day	Current day of year	

ISO4 format:

0	4	5	7	8	10	13
y y y y	X’60’	m m	X’60’	d d	n n n	X’40’
Year	-	Month	-	Day	Current day of year	

Input editing:

The dialog manager calculates the current day of the year and returns the date in ISO or ISO4 format to the corresponding dialog variable.

Mask field		Dialog variable
Date with calendar check; 2-digit year; Separator “/”		Type CHAR; Length 12 BYTES
1 / 4/88	--------->	88-04-01092

Without calendar check		
33/0/88	--------->	88-00-33000

Output editing:

The day, month, year, and the separators are transferred to the appropriate position of the mark field. The current day is not output.

The positions for day, month, and year are padded with leading zeros if required so that the output always has a length of 8 or 10 positions.

Dialog variable		Mask field
88-04-01....	--------->	01/04/88

Without calendar check		
88-00-33....	--------->	33/00/88

Time

The mask field contains a time entry. The dialog variable must be of type CHAR.

Input validation check:

Only the character defined as the time separator may appear at specific positions in the mask field. Leading zeros may be omitted for hours, minutes, and seconds. The separator may be preceded and followed by blanks. The presence of blanks followed by output fill characters before the time or of output fill characters followed by blanks after the time leads to an error if the output fill character is not a blank. Blanks are permitted at the start and end of the time (regardless of the output fill character).

Output validation check:

The dialog variable must contain the value HH:MM:SS or HH:MM, depending on the specification for seconds. The following values are possible: 00-23 hours for HH, 00-59 minutes for MM, and 00-59 seconds for SS.

Input editing:

A ":" is always used as the time separator.

Mask field		Dialog variable
Time with seconds; Time separator ":"		Type CHAR;
12:08:36	--------->	12:08:36

Output editing:

The time separators are inserted at the appropriate position.

Dialog variable		Mask field
Type CHAR; length 8		Time with seconds; Time separator "."
12:08:36	--------->	12.08.36

Value list

This entry is possible for every input field of the mask.

Input validation check:

The entered value must either match a value of a value list (check for “equal”) or not be contained in the value list (check for “not equal”). The validation conditions and the value list are specified in the format definition.

Example:

Entered value: DOG
Predefined value list: CAT DOG HEN SHEEP
Check for equal: no error

Value range

This entry is only allowed for arithmetic mask fields.

Input validation check:

The entered value must be included in a predefined numeric value range (including the upper and lower bounds). Signs may be used. The minus sign must be specified for negative values.

Example:

Entered value: 123
Predefined limits: 222- 222+
Result of the check: no error

Character set

This entry is possible for all input fields of a mask.

Input validation check:

All entered characters must either be contained in a character set defined using IFG (check for “equal” or the characters must not appear in the character set (check for “not equal”). The validation conditions and the list of characters are specified in the format definition.

Entered value: AB
Predefined character set: ABCDEF
Check for “not equal” error message

Data conversions in the DISPLAY service

The following table contains the possible data conversions between dialog variables and the mask field for TIAM.

Dialog variable	Mask field				
	String	Arithmetic string		Date	Time
		Signed	Unsigned		
CHAR	+	+ 1)	+ 1)	specific data	specific data
BINSTR	+	+ 1)	+ 1)	specific data	specific data
NUMS	+ 2)	+	+	-	-
NUMU	+ 2)	+	+	-	-
FIXEDS	+ 2)	+	+	-	-
FIXEDU	+ 2)	+	+	-	-
PACK	+ 2)	+	+	-	-

The following table contains the possible data conversions between dialog variables and the mask field for SDF-P.

SDF-P variable	Mask field				
	String	Arithmetic string		Date	Time
		Signed	Unsigned		
STRING	+	+ 1)	+ 1)	specific data	specific data
INTEGER	+ 2)	+	+	-	-

Key:

+ Data conversion is possible (but sign errors may occur)

- Data conversion is not supported (conversion error)

1) The value of the dialog variable to be displayed must be numeric.

2) The content of the mask field on input must be numeric.

Restrictions and incompatibility

Floating sign:	Only possible in combination with “sign allowed”.
Mandatory input:	The “mandatory” attribute cannot be combined with the “protected”, “markable”, and “automatic input” attributes:
Protected	A “minimum input length > 0” must not be requested for a protected field.
Automatic input:	Is ignored for menu titles and input selection fields.
Zero suppression:	A combination of “zero suppression” and “floating sign” is not meaningful with output fill characters other than blanks, since “floating sign” always uses blanks for padding. “Zero suppression” and the output fill character “0” are mutually exclusive.

3.9 Commands

The dialog manager supports the use of commands. Depending on how commands are processed, the following options are available:

– Systems commands are processed by the dialog manager and not passed to the application program.

– Application commands are passed to the application program and must be processed by it.

A further distinction is made based on the method by which the command is entered. The options in this case are:

– Input of a command in the command input field.

– Activation of a command by pressing a function key. A key assignment table in which commands are assigned to function keys (called a key list in IFG) must be defined for this purpose.

– Command input by combining a command activated via a function key with the input in the command input field.

Command area

The command area in a mask is specified when defining the format using IFG. A command area is not mandatory.

If a system command is entered in the command input field, the command is processed by the dialog manager. The command input field is then deleted unless the command name was entered with a preceding “&” character.

If the entered string is not a system command, it is interpreted as an application command. The input data in the command input field is written to the dialog variable assigned to this mask field. If the name of this dialog variable is not SYS-COMMAND, the input data is also made available in the SYS-COMMAND dialog variable in the function pool. The variable is created implicitly if it does not exist. SYS-COMMAND is written even if no command input field exists and an application command is entered by means of a function key.

If the dialog variable assigned to the command input field of the mask is not empty at the time of a DISPLAY call, its value is shown in the command area. This allows an application program to implement a predetermined setting.

Note:

By assigning the dialog variable SYS-COMMAND to all command input fields, it is possible to have application commands written to and read from the dialog variable SYS-COMMAND throughout the entire application.

Entering commands via function keys

System and application commands can be assigned to F keys. If there aren't enough F keys available, the system command SETP can be used to simulate F keys using P keys. Pressing an F key has the same effect as entering the command in the command input field and hitting the ENTER key. Application commands entered by this method are made available to the application program in the system variable SYS-COMMAND.

Entering composite commands

If a command has been assigned to an F key via the key assignment table (referred to below as the key list) and if a command input field exists in the mask, a string can be entered in this field before pressing the F key. The dialog manager connects the data of the F key, separated by a space, with the data from the command input field. FHS combines the two strings with one another as follows:

F key	␣	Command field

Example:

- If "KEYAREA" is assigned to function key F20
- and "OFF" is entered in the command input field,
- pressing the function key F20 will execute the command "KEYAREA OFF".

Activating an FHS command from an application

You can activate an FHS command from an application by defining a format with a command area and by assigning the desired command (by means of a string assignment) to the dialog variable that is associated with the command input field. When you subsequently call the DISPLAY service with PANEL(name) and the operand NODISPLAY, FHS-DM will process the command without displaying the format.

Example 1:

Let us assume you want to use P keys 1,3 and 12 to simulate F keys. The appropriate command for this purpose is:

```
SETP (P1,P3,P12) ON.
```

Assign this string to the dialog variable associated with the command input area and call the FHS-DM service with DISPLAY PANEL(name) NODISPLAY.

The P1, P3 and P12 keys will then be available for FHS.

Example 2:

Command “SYS”

If you assign the dialog variable the string:

```
“SYS FILE-STATUS A*“
```

and call the display service as above, all files of the specified template will be shown. In addition, the prompt characters “ACK” will be output before the next display of the DISPLAY service in order to obtain an input.

3.9.1 System commands of the dialog manager

Operation	Meaning
ABORT	Terminate application
ACTIONS	Place cursor in menu bar / mask fields
CANCEL	Cancel display
EXIT	Exit application segment
EXTHELP	Request extended help
HARDCOPY	Output screen contents to printer
HELP	Request help
HELPHELP	Help on help system
INDEX	Overview of available help
KEYAREA	Turn on / turn off display of key list
KEYSHELP	Help on key list
PANELID	Turn on / turn off display of format name and message code
RMSG	Redisplay a message
SETP	Assign P keys
SYS	Call BS2000 commands
RESHOW (K key)	Redisplay previous mask
FORWARD BACKWARD + - ++ --	Scrolling and navigation commands

System commands may be entered in lowercase or in uppercase.

The commands KEYAREA, PANELID and SETP apply to all subsequent outputs for the entire dialog complex regardless of the communication area (see page 118). These specifications are saved in the profile on a DMCLOSE for the dialog complex and supplied again at the next DMOPEN with this profile.

ABORT - Terminate application

Operation	Operands
ABORT	

When FHS-DM detects an error that prevents normal continuation of an FHS service being processed, an error mask can be output. The return code associated with the error is then supplied to the application.

If the ABORT command is given when the error mask is displayed, the MAINCODE will be set to 399999 on return to the application. The application should be terminated for this return code. (FHS-DM does not normally terminate processing independently!)

If ABORT is entered in an error mask which was output as a result of implicit actions and which normally allows the application to be continued, the display service terminates with MC=399999 in this case as well.

ACTIONS - Place cursor in menu bar

Operation	Operands
ACTIONS	

ACTIONS places the cursor in the menu bar. If the cursor is already in a menu bar or in a pull-down menu, it is returned to the field in which it was positioned before the preceding ACTIONS command. If no preceding ACTIONS command was issued, the cursor is placed in the first input field of the work area.
This command can only be meaningfully used as an F key (the default is F10). It has no effect if no menu bar exists.

Operands, if specified, are ignored.

CANCEL - Exit/terminate display

Operation	Operands
CANCEL	

If CANCEL is issued for an implicit box (i.e. with the cursor positioned in a box), the topmost implicit box is removed, and the underlying implicit box is displayed (if one exists). Otherwise, the basic format is shown. The application program is not notified.
In order to prevent the cursor from being inadvertently placed at an incorrect position, all implicit boxes are removed even if the cursor is in the basic format or in an explicit box. In this case, the command has no effect on the basic format or explicit box. It is only after removing all implicit boxes that a repeated CANCEL command will have an effect on the basic format.

If CANCEL is issued for a pull-down box, the pull-down box is removed, and the cursor is placed on the first menu title. Any selection that may have been entered in the pull-down box is ignored.

If CANCEL is issued for an explicit box or the basic format, the current display call is terminated with an appropriate return code. All input data is lost; the input fields are not checked by the dialog manager. On return to the application program, dialog variables have the same contents as before the DISPLAY service, except in the case of a list. When a list is displayed, the modified list fields are saved in the dialog variables as soon as the list is scrolled. Consequently, it is only the current data of the list area that is not processed as a result of a CANCEL command. The application program must respond to the return code appropriately (e.g. go back one display step). The return codes are listed in the appendix starting on page 233.

Operands, if specified, are ignored.

EXIT - Terminate application segment

Operation	Operands
EXIT	

If EXIT is issued for an implicit box (i.e. with the cursor positioned in a box), all the implicit boxes are removed and the basic format is displayed. The application program is not notified.
In order to prevent the cursor from being inadvertently placed at an incorrect position, implicit boxes are removed even if the cursor is in the basic format or in an explicit box. In this case, the cursor has no effect on the basic format or explicit box. It is only after removing all implicit boxes that a repeated EXIT command will have an effect on the basic format.

If EXIT is issued for an explicit box or the basic format, an application segment or the application itself should be terminated. The DISPLAY call terminates and supplies the application program with a corresponding return code (see page 233ff.). Termination of an application segment must be implemented by the application program. If the EXIT command is given in combination with modified input data, the input fields are checked in accordance with the validation conditions for them. If an error is detected in this case, an error message is issued by the dialog manager and the EXIT command is ignored. The command can be repeated after the incorrect data has been corrected. If there are no errors in the input data, the corresponding variables are updated.

Operands, if specified, are ignored.

EXTHELP - Request extended help

Operation	Operands
EXTHELP	

Extended help returns information on the mask from which the help was requested. If there are other help panels being displayed on the screen, they are first removed.

Specified operands, if any, are ignored.

EXTHELP in the KEYSHELP mask requests help on key assignments.

HARDCOPY - Output screen contents to printer

Operation	Operands
HARDCOPY	

The current contents of the screen are output to a local hardcopy printer. The string “HARDCOPY” is removed from the command area on execution.

Operands, if specified, are ignored.

HELP - Request help

Operation	Operands
HELP	command

HELP returns field-related help on an input field, output field, menu title, markable field, or command input field. Help can also be defined for protected output fields and text fields, provided the cursor can be positioned on protected fields (NOAUTOTAB).

Depending on the type of field, the following applies:

– If the cursor is on a single-choice field with a valid entry, help on that entry is returned.

– If the cursor is on an empty single-choice field, help on the entire-choice field (= global help) is output.

– If the cursor is in an input field of a multiple-choice field, help for that input field appears. A repeated HELP command in this case outputs global help for the multiple-choice field (if available) or, alternatively, the extended help.

– If the cursor is in the command field, and that field is empty or contains an application command, global help on the command field is displayed. If no such help is available, FHS outputs the default help for the command area. If the command field contains an FHS command, then help on that command is output.

– If no help for a normal input field exists, the extended help for the format is output. If this is not available either, FHS issues a corresponding message.

HELP “command” returns help on the specified system “command”. If “command” is not a system command, help on the command line is shown. The cursor must be in the command input field.

HELPHELP - Request overview of the help system

Operation	Operands
HELPHELP	

Help on the help system returns information on using the help commands HELP, EXTHELP and KEYSHELP and on the help system itself.

Operands, if specified, are ignored.

INDEX - Display index of FHS-DM keywords

Operation	Operands
INDEX	

INDEX causes a help panel (IDHAIDX) to be displayed with FHS-DM keywords that are defined as cross-references. If you place the cursor on any of these keywords and press the ENTER or HELP key, help on that keyword is shown. You can extend this help (IDHAIDX) by adding your own keywords.

Operands, if specified, are ignored.

KEYAREA - Activate/deactivate display of key assignments

KEYAREA can be used to toggle the display of key assignments on or off. The setting remains in effect until the next time it is toggled and is saved in the profile pool as the system variable SYS-KEY-AREA.

Operation	Operands
KEYAREA	[ON / OFF]

ON Turn on display of key assignments

OFF Turn off display of key assignments. This provides additional space for the output of help texts.

If no operand is specified, the current setting is toggled.

KEYSHELP - Help on key assignments

KEYSHELP shows a key assignment in the form of a table containing the name of the key, the command assigned to it, and a brief designation (label) for the key.

Operation	Operands
KEYSHELP	[SHORT]

SHORT A short form of the key assignment is shown.

PANELID - Toggle display of format name and message code

The PANELID command can be used to turn on or turn off the display of the panel ID (i.e. the names and identifiers of formats). The setting remains in effect until the next change and is stored in the profile pool as the system variable SYS-PANEL-ID.

Operation	Operands
PANELID	[ON / OFF]

ON Turns on the display

OFF Turns off the display If no operand is specified, the setting is toggled.

RESHOW - Redisplay a mask

This command is always bound to a K key; it cannot be directly entered in a format. By default, this function is executed by pressing the K3 key (see the section on “Key lists” on page 77ff.).
The RESHOW function causes the last output of the mask to be repeated. Inputs that were already made on the screen are lost.

Operation	Operands
K key	

The RESHOW function is not an actual system command, since there is no string that can be entered in the command input field. The function is always bound to a K key (usually K3) and can only be executed by pressing that key. The function is identified by the string “RESHOW” when it is assigned to the K key in the key assignment table (called a key list in IFG).

RMSG - Redisplay a message

The RMSG function causes a message that was requested by the application in the DISPLAY command to be redisplayed. If an error mask is being displayed, the message for the error code is output.

Operation	Operands
RMSG	

Operands, if specified, are ignored.

SETP - Assign P keys

SETP can be used to bind the function performed by function keys identified by a number to P keys with the same identification number or to cancel such assignments. The assignment remains in effect until the next change and is stored in the profile pool as the system variable SYS-P-KEYS-SETTING.t

Operation	**Operands**
SETP	Pn / (Pn, ... ,Pm) / (Pn-Pm) ON / OFF [, ...]

Pn	The P key with number n.
(Pn,..,Pm)	All P keys individually listed in the form Pn, .., Pm.
(Pn-Pm)	All P keys from Pn to Pm
ON	Turns on the assignment for the specified keys.
OFF	Turns off the assignment for the specified keys.

The operands of SETP may be specified more than once, but each such assignment must be separated by a comma. The commas may be surrounded by any number of blanks.

Example

```
SETP P1 ON,     P2 OFF,  (P3,P7,P8) ON,   (P4–P6) OFF
```

After this SETP command, the following assignments will take effect:

```
P1 – F1        P3 – F3      P7 – F7      P8 – F8
```

The assignments for P2, P4, P5 and P6 were removed.

Scrolling commands

These commands can be used to scroll a list area of a mask or a help text displayed in a panel.

The characters “++” and “--” scroll the display to the end of the text in the specified direction.

Format and operand description

Operation	**Operands**
FORWARD	[number]
BACKWARD	[number]
+	[number]
+ +	
-	[number]
- -	

The “number” operand specifies the number of lines by which the list area is to be scrolled. If no number is specified, the FORWARD and BACKWARD commands move the lowest list line to the top and the topmost list line to the bottom, respectively.

The dialog manager also supports scrolling based on the current cursor position. If the cursor is located in a list line or in the help text, the “+” or FORWARD command scrolls that line to the top of the list area or work area of the help panel, whereas “-” or BACKWARD scrolls it to the bottom.

SYS - Execute BS2000 commands

The BS2000 command specified as an operand is executed, and the screen is then restored to its existing state before command execution.

Operation	**Operands**
SYS	BS2000 command

Any BS2000 command supported by the BS2000 macro CMD may be specified as the operand. It is not advisable to use a /START command since that would abort the application program.

3.10 Key lists

Key lists are created with IFG and stored in the format library as special formats (called KEY formats). The name of a KEY format can be assigned to any format when defining the format with IFG. If no specific KEY format name is specified, a default KEY format is assigned. In other words, every format is assigned a key list when it is defined with IFG.

A key list includes the following entries for each function key:

Fxx [command] [label]

“Fxx” identifies the function key; “command” is the command assigned to this key, and “label” is a freely selectable short text (of up to 12 characters) that describes the key.

The key list assigned to a format is displayed by the dialog manager in the key list area of the mask. This area is optional and consists of a maximum of two lines. The key assignments are displayed in the form “Fxx=label“.

Keys for which no labels have been specified in the key list are not displayed. This allows the application programmer to display only important keys in the key list area. All key assignments are activated even if they are not displayed in the key list area. This also applies if no key area was defined for a format. The currently active key assignments can be displayed as a table by means of the system command KEYSHELP.

If all keys in the key list do not fit in the key list area, the dialog manager indicates the presence of further keys by inserting up to three periods following the last possible complete entry.

Default key lists

Default key lists are supplied together with the dialog manager. The following KEY formats are provided in English and German:

IDHKEYS	Default KEY format
IDHKEYA	KEY format for full screen with menu bar
IDHKEYF	KEY format for field-related help
IDHKEYE	KEY format for extended help
IDHKEYH	KEY format for help on help
IDHKEYK	KEY format for help on key assignments
IDHKEYM	KEY format for messages in a box
IDHKEYN	KEY format for message boxes without help
IDHKEYI	KEY format for index format

The following key lists show the assignments for the German version (with translated labels):

IDHKEYS - General formats

Key	**Command**	**Label**
F1	HELP	Help
F3	EXIT	Exit
F4	HARDCOPY	
F7	BACKWARD	
F8	FORWARD	
F11	INDEX	
F12	CANCEL	Cancel
K3	RESHOW	

IDHKDER - Key format for error handling in dialog mode

Key	**Command**	**Label**
F1	HELP	Help
F3	EXIT	Exit
F4	HARDCOPY	Hardcopy
F5	RMSG	Fetch message
F6	ABORT	
F9	KEYSHELP	Keys
F11	INDEX	
F12	CANCEL	Cancel
K1	ABORT	
K3	RESHOW	

IDHKEYA - Key format for full screen with menu bar

Key	**Command**	**Label**
F1	HELP	Help
F3	EXIT	Exit
F4	HARDCOPY	
F7	BACKWARD	
F8	FORWARD	
F10	ACTIONS	Menu
F11	INDEX	
F12	CANCEL	Cancel
K3	RESHOW	

IDHKEYF - Field-related help

Key	**Command**	**Label**
F1	HELP	Help
F2	EXTHELP	Extended help
F3	EXIT	
F4	HARDCOPY	
F7	BACKWARD	
F8	FORWARD	
F9	KEYSHELP	Keys
F12	CANCEL	Remove
K3	RESHOW	

IDHKEYE - Extended help

Key	**Command**	**Label**
F1	HELP	Help
F3	EXIT	
F4	HARDCOPY	
F7	BACKWARD	
F8	FORWARD	
F9	KEYSHELP	Keys
F11	INDEX	
F12	CANCEL	Remove
K3	RESHOW	

IDHKEYH - Help on help

Key	**Command**	**Label**
F3	EXIT	
F4	HARDCOPY	
F7	BACKWARD	
F8	FORWARD	
F9	KEYSHELP	Key
F11	INDEX	
F12	CANCEL	Cancel
K3	RESHOW	

IDHKEYK - Help on key assignments

Key	Command	Label
F1	HELP	Help
F2	EXTHELP	Extended help
F3	EXIT	
F4	HARDCOPY	
F7	BACKWARD	-
F8	FORWARD	+
F11	INDEX	
F12	CANCEL	Cancel
K3	RESHOW	Reshow

IDHKEYM - Message box with help

Key	Command	Label
F1	HELP	Help
F3	EXIT	
F4	HARDCOPY	
F11	INDEX	
F12	CANCEL	Remove
K3	RESHOW	

IDHKEYN - Message box without help

Key	Command	Label
F1	HELP	
F3	EXIT	
F11	INDEX	
F12	CANCEL	Remove
K3	RESHOW	

IDHKEYI - Index format

Key	Command	Label
F1	HELP	Help
F3	EXIT	
F4	HARDCOPY	
F7	BACKWARD	
F8	FORWARD	
F9	KEYSHELP	Keys
F12	CANCEL	Remove
K3	RESHOW	

IDHKEYU - Format for boxes

Key	Command	Label
F1	HELP	Help
F3	EXIT	
F12	CANCEL	Cancel
K3	RESHOW	

Displaying key assignments

The assignments for function keys F1, F3 and F12 are shown in the command area by default; the assignments for F7 and F8 are also included with the output of lists. This display can be turned on or turned off with the FHS command KEYAREA.

The key assignment for every DE format can be displayed with the FHS command KEYSHELP.

Note:

Make sure that you reserve a K key for RESHOW!
Reason: when a mask is output several times in succession, only the variable fields are refreshed by FHS-DM (i.e. only an “Update output” is generated). If the mask is overwritten by an operator message, for example, it will not be refreshed at the next output. It is only by using RESHOW that the complete mask can be redisplayed (see also page 94).

3.11 Output of messages

FHS-DM can facilitate the work of terminal users by informing them of certain events by means of DM messages. These DM messages (which are referred to simply as messages below) are either output in the message area of the format or in separate message boxes. FHS distinguishes between implicit and explicit messages.

Implicit messages are output by FHS independently of the application. Typical examples of such messages include the messages output by FHS when checking input fields.

Explicit messages are issued by the program by means of a message code and a field name in the MSGLOC operand in a call to the DISPLAY service.

Creating messages

Messages must be created using IFG. This is done by calling the “Edit messages” mask in IFG and defining the text of the message in it. The message text can have a maximum length of 256 characters. Besides the text of the message, it is also possible to define other attributes. These include:

– the message code in the form AAAAnnn or AAAnnnn, where AAAA are alphabetic characters (A-Z), and nnn are digits (0-9). The IDHx designations should not be used as message codes, since they are reserved by FHS for internal messages.

– the output position of the message. This can be the message area of a format or a modal or modeless message box. For further information, see the section on “Implicit boxes”, on page 17ff.

  If “output position = message area” was defined, but the output is not possible in the message area, FHS outputs the message in the form of a modeless box. This can happen, for example, if the message area is too small or if a message area has not been defined in the format.

– the type of message: Information/Warning/Error/Danger

– the name of a help panel for the message (optional)

Editing the message text

If the string “%%” appears in the message text, the text that follows is continued in the next line of the message box as of column 2. This editing attribute can be used to reduce the width of a message box. The string “%%” is replaced by a blank in the output of the message in the message area, and the string itself is removed.
“%%%%” generates a blank line.

The character “&” must be specified as “&&” in the message text.

The width of a message box is adjusted by FHS-DM to the message text. The minimum and maximum widths for a mask are 20 and 56 characters, respectively. A message box can have up to 6 text lines. A mask width of less than 56 characters is used if the message text is shorter or if editing characters are used to set the line length. If the message text is longer, the maximum width is selected, and the message text is adjusted to it. Separation occurs at a blank.

If the edited message text exceeds the maximum size of a message box, the editing characters are replaced by blanks, and the text is split after every 56 characters. If the complete text still cannot be displayed, the last line ends with “...”.
When FHD-DM encounters “&name” in the message text, the current contents of that field variable are inserted into the message text.

Implicit messages

Implicit messages are output independently by the dialog manager without any action on the part of the application program. Such messages are typically issued when input validation checks are performed. A number of default message formats are supplied for implicit messages. These formats must be copied into the format library of the application or be made available as an alternative format library. The default messages have message codes in the form IDHxnnn, where nnn is a three-digit number, and “x” stands for the letter “F”, “I” or “S” The message text can be modified by the user with IFG.

If a field containing an error is detected by the dialog manager, a message box is output in relation to the first invalid field unless the message area was defined as the output location. If more than one field is invalid, all such fields are highlighted (e.g. underlined). The cursor is placed in the first invalid field, and subsequent scrolling commands are ignored. These commands can be repeated after the error has been corrected.

The message for the field can be specified in IFG as follows:

– *NONE or blanks: default message of FHS

– Message code: user-defined implicit message

The following additional specifications can be included for user-defined implicit messages:

&SYS-PAR0	Name of the full format or name of the active explicit box.
&SYS-PAR1	Name of the current field
&SYS-PAR2	Contents of the current field
&SYS-PAR3	Name of the current format

If the &SYS-PAR0, &SYS-PAR1, &SYS-PAR2 or &SYS-PAR3 codes are found in the message text, they are replaced in the message text by their corresponding values. This is only possible for messages that have message codes for which checking was defined using IFG.

These messages are output by default in a message box; however, you can change the output location with IFG so that the message is output in the message area of the format instead.

Explicit messages

When a message that refers to the content of a specific mask field (or to a specific list line of a list field) is to be output by the application, a message box can be positioned at that field. This is achieved by entering a message code in the MSG operand and the name of the field to be used for positioning a message box in the MSGLOC operand of the DISPLAY service.

If a message box was specified as the output location for a message when the message was defined, the following variations are possible:

– If MSGLOC is not specified, the message box is output in relation to the field specified by the CURSOR operand. If a list field is involved, the field can be specified with an index. The cursor position is set in accordance with the entries of CURSOR and CSRPOS.

– If the name of a field that is not a list field was specified for MSGLOC, the message box is output in relation to that field. The cursor is positioned as defined by the entries for CURSOR and CSRPOS.

– If the name of a list field is specified for MSGLOC, an index to select a list line can be specified. The message box is then output in relation to this field and the list line defined by the index. If no index is specified, the message box is output in relation to the field of the topmost line in the list area (see also “List processing, TOPINDEX variable”).

– It is also possible to specify an absolute position ($lll#ccc) as a value for the MSGLOC operand. The value *CENTRAL causes the message box to be output in the center of the screen.

– If the specification for MSGLOC would cause the requested cursor position to be covered by the message box, the MSGLOC operand is ignored, and the message box is positioned in accordance with the cursor specification.

– If neither MSGLOC nor the cursor position are specified, the message is output in the center of the screen.

The entry in the message definition determines whether the message box is to be output as modal or modeless.

If the message area was defined as the output location for the message to be displayed, the MSGLOC operand is ignored.

3.12 Help system

The application developer can create an extensive help system for the terminal user with FHS-DM. This help system can be customized for each application.

Help information is displayed in the form of implicit boxes and is itself a format. Application-specific help panels can be created using IFG; some default help panels are supplied with FHS. The attributes of implicit boxes are described on page 17.

The scrolling commands FORWARD and BACKWARD are permitted in help panels, since the size of the work area may not allow all information to be completely displayed in a single help box. The options for application developers are listed in the following section.

3.12.1 Help that can be created by the application developer

The application developer can create the following help:

– extended help on the format
– field-related help on input fields, output fields, and selection fields
– global help on selection fields and for the command area
– help on messages
– cross-references

Extended help on the format

When you generate a format with IFG, you can allocate a help panel (which you also generate with IFG) to that format. This help is output when the user:

– enters the command EXTHELP (also activated by a function key; see the section on “FHS commands” on page 64,

– places the cursor in a help box that was requested for field-related help and presses the “EXTHELP” key,

– places the cursor in the mask or dialog box on a field for which no field-related help exists and activates “HELP”. If AUTOTAB is set (as it is by default), the cursor can only be placed in unprotected input fields or in protected marked fields. Extended help may then need to be requested as in the first case.

– places the cursor in a field for which field-related or global help is being displayed and then presses HELP again.

If no extended help is defined but a request for it is made, a corresponding message is output.

Field-related help

When you define a format using IFG, you can assign a help panel to each field (even a protected output field).

If you define help on an output field, it must be possible to position the cursor on that field, i.e. either the output field must be markable or the format must have the tab attribute NO AUTOTAB.

The help panel is displayed when the terminal user positions the cursor in the field concerned and enters the HELP command or presses the corresponding F key.
If the field is part of a single-choice field, the user must enter the number of the desired choice in the appropriate input field and specify the HELP command. (If no number is entered, global help is provided if available; see “Global help”).

Global help

In terms of its information content, global help lies between the extended help provided for a complete format and the restricted field-related help for a single field. You can create global help by using IFG to define a help panel with the following objects :

– Single-choice field. Global help can be obtained by positioning the cursor in the *empty* input field of the single-choice field and specifying the HELP command.
If no field-related help for is available for a particular choice, the global help is output immediately; otherwise, only after the second HELP command.

– Multiple-choice field. Global help can be obtained by positioning the cursor in any input field of the multiple-choice field and specifying the HELP command once or twice. If no field-related help exists, the user is provided with global help immediately; otherwise, after the second HELP command.

– Command area of a format. Global help can be obtained by specifying the HELP command when the cursor is in the command field and command field is empty or contains an application command.

– List area of a format. Global help can be obtained by positioning the cursor in a list field and specifying the HELP command once or twice. If no field-related help exists, global help is output immediately; otherwise, after the second HELP command.
Global help is also displayed if HELP is pressed when the cursor is located in the list area outside a field.

Help is output in a box. For selection fields, the box appears below the corresponding input field (if possible); for the command area, it always appears above the command field.

Help on messages

If you want to define a help text for a message, you must specify the required help panel explicitly when generating the message format with IFG.

This help panel is displayed if the terminal user presses the HELP key after the message is output. If the message is to be output in a box, the cursor must be located within the message box.

Cross-references

Cross-references are hypertext links to text fields in help panels. They are similar to field-related help in action formats. The user can obtain information on a cross-reference by placing the cursor in a text field for which help is defined and pressing the ENTER key or issuing a HELP command. This causes a new help box to be output with help information displayed in it. Further cross-references may be contained in this box.

If a hierarchy of cross-references is requested, the corresponding help boxes are always output at the same position. Consequently, only the topmost box will be visible if all cross-references are of the same size. This should be taken into account when designing the help panel.

INDEX

The INDEX command allows you to search for help supplied with the FHS dialog extension. On entering the INDEX command, you will receive a help panel containing keywords for FHS-DE. If you place the cursor on any keyword and press the ENTER key, a new help panel will be output with help information displayed in it. This panel may, in turn, also contain new cross-references.

3.12.2 Help provided by FHS

A number of standard help panels are supplied with FHS. The names of these help panels all begin with IDHx. These standard help panels are contained in the library SYSFHS.FHS.081.FHS-DM.E. A LINK to BLSLIBxx is sufficient to link in the help panels (xx = 00 - 99). This LINK can take the following form:

```
SET-FILE-LINK FILE-NAME=$.SYSFHS.FHS.081.FHS-DM.E, LINK-NAME=BLSLIB00.
```

FHS offers the following help to the terminal user:

– help on messages from FHS
– help on FHS commands
– help on key assignments
– help on the help system

Help on messages

Help on messages from FHS is handled analogously to the help on application-specific messages (described above).

Help on FHS commands

The user can request help on a specific FHS command by entering HELP “fhs-command” in the command field and pressing the ENTER key or by entering the command and pressing the HELP key. The help on FHS commands appears as global help. If application-specific help exists for an FHS command, the user receives the application-specific help at the first HELP call and the FHS help at the second.

Help on key assignments

Help on key assignments can be activated by two methods:

– The terminal user issues the FHS command KEYSHELP or presses the assigned F key.

– The cursor is placed in the bottom frame of the box or in the key list display and the HELP command is issued.

FHS displays the assignments for all F keys and K keys in a table; see the description of the KEYSHELP command. Only those keys for which entries were made when creating the KEY format are shown. If all key assignments are not visible, you can scroll forward or backward as required.

When the key list is displayed, extended help may be requested:

– If a help panel was defined when assigning keys, the EXTHELP command displays that help panel. If the cursor is outside the command area, the same help can be displayed with the HELP command.

– If no help panel was defined, the EXTHELP command displays the message “No help exists”. If the cursor is outside the command area, the HELP command displays help on the KEYSHELP command.

The associated default help panel ((IDHKHLP) can be modified to some extent in IFG, e.g. by altering text fields in titles, display attributes, or the explanatory text for “*”, which is enclosed within < >. This text appears in the command field in the first data line of IDHTKHP.

You cannot, however, change the structure of this format. If the structure of the IDHTKHP format is destroyed by more radical changes, the KEYSHELP function can no longer be implemented.

Help on the help system

The help on the help system contains information on the type of help available and the ways in which it can be accessed. The FHS command HELPHELP or the corresponding F key activates this help.

3.13 Cursor positioning

Different methods are used for positioning the cursor on input and output.

Cursor positioning on output

The cursor position can be controlled by means of the CURSOR and CSRPOS operands of the DISPLAY service. If the CURSOR operand is not specified, the cursor is placed in the first input field. Alternatively, if the format does not contain an input field, the cursor is placed in the first column of the first line on the screen.

Depending on the value of CURSOR, the following rules apply:

1. CURSOR contains an absolute position ($lll#ccc). In this case, the specifications for CSRPOS are ignored.
2. CURSOR contains a field name. If the field name is invalid, i.e. does not exist in the current format, the DISPLAY service terminates with an error. An index may be specified if the field name refers to a line of a list. If no index is specified in such cases, the topmost line displayed in the list area is assumed (see also “List processing, TOPINDEX variable”). CSRPOS specifies the offset in the field. If the value of CSRPOS is invalid, it is set to 1.

Cursor specifications on input

The cursor position is returned in the system variables SYS-CURSOR-FIELD, SYS-CURSOR-POS and SYS-CURSOR-INDEX. If the cursor is positioned on an unnamed (text) field or in the intermediate space between fields, SYS-CURSOR-FIELD contains the absolute cursor position ($lll#ccc); SYS-CURSOR-POS and SYS-CURSOR-INDEX have undefined values.

3.14 Update output

If the same format needs to be output again and no implicit actions of the dialog manager (e.g. help requests, messages) have been executed, only an “update output” is sent to the screen in order to reduce the transmitted message. In other words, only the modifiable input and output fields are redisplayed; text fields are not transferred.

The display services can normally detect if an update output is possible. In some cases, however, if output from an external source (e.g. operator messages) has been inserted, the display services cannot assume control.

If the intermediate output is generated by the application program (directly or indirectly by calling system programs), the CONTROL service with the REFRESH or ACK operands can be used to prevent an update output.

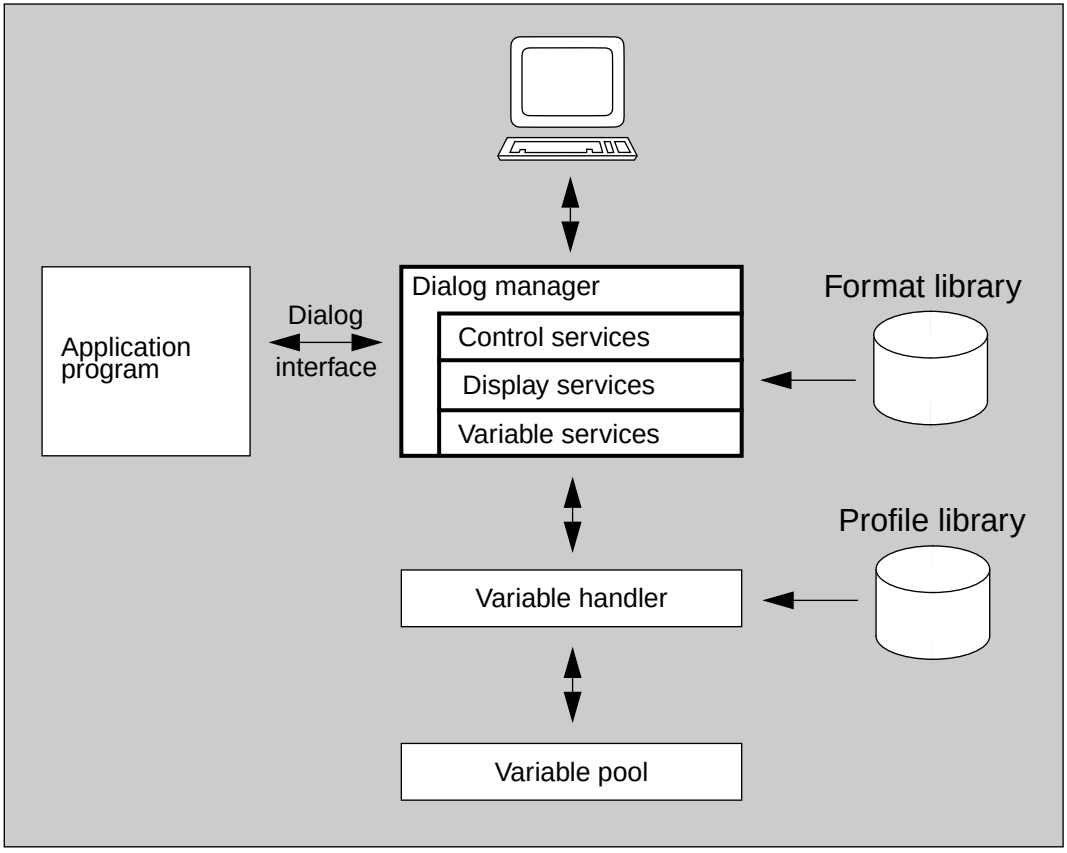
In order to handle cases when unexpected messages such as operator messages are inserted between two outputs, the RESHOW command must be assigned to a K key (K3 is recommended) in the key format so that the terminal user can fully refresh the screen by pressing the K key.

The DISPLAY service detects when outputs to different DMCOMM of a dialog complex are requested, but not an output to a second dialog complex. The CONTROL service must always be called with the REFRESH operand in such cases.

4 Program interface for TIAM application programs

The dialog manager offers a number of different dialog elements and services that greatly simplify and facilitate the development of dialog applications. The actual processing is carried out by the application program itself by using the dialog services of the dialog manager to display masks, messages, and help text at the terminal.

The following diagram shows an overview of the dialog manager components:



The following dialog elements are available:

– formats
– messages
– help texts
– key assignment tables (also called key lists)

These dialog elements are created with the aid of the Interactive Format Generator (IFG) and stored in a format library. In order to process dialog elements with the dialog manager, they must be created with an IFG version as of version 8.1.

The following dialog services are available:

Display services

Services to design the layout of the screen interface in accordance with the SNI Alpha Style Guide.

Name	Purpose
DISPLAY	Display a format and/or a message
ATTR	Define dynamic field attributes
ADDPOP	Initialize a dialog box
REMPOP	Remove a dialog box

General variable services

Services that enable the application program to define, modify, and delete dialog variables.

Name	Purpose
VCOPY	Read dialog variables
VDEFINE	Define explicit dialog variables
VDELETE	Delete dialog variables in the function pool
VREPLACE	Replace dialog variables

Variable services

Services to control an application profile.

Name	Purpose
VPUT	Write dialog variables into profile pool / SDF-P-Pool
VGET	Read dialog variables from profile pool / SDF-P-Pool
VERASE	Delete dialog variables in profile pool

Control services

Services that enable an application program to initialize and terminate dialog manager services and set the operating mode.

Name	Purpose
DMOPEN	Begin use of dialog services
DMCLOSE	End use of dialog services
CONTROL	Set operating modes

Other services

Function key support	
System commands of the dialog manager	Abort format display Display format name and message key Copy screen Execute BS2000 commands Request help Display key assignments
Provision of system variables	
Support for application commands	

4.1 Variables of the dialog manager

The following variable types are differentiated in this manual:

– dialog variables (see the following sections)

– program variables (variables that can be used in an application program)

– procedure variables / S variables (variables that can be used in an SDF-P procedure)

4.1.1 Dialog variables

Dialog variables enable the exchange of data between the application program, the dialog manager, and dialog elements. These variables can be defined, supplied with values, modified, and deleted by the application program with the aid of the variable services of the dialog manager. It is also possible to define links between dialog variables and program variables.

An important use of dialog variables is in the exchange of data between mask fields and application program data. When a format is defined using IFG, names of dialog variables are assigned to the I/O fields of a mask. When the dialog manager displays a mask, the values of the assigned dialog variables are shown in the mask fields. Any data that is entered in an input field of a mask is then stored as the value of the corresponding dialog variable.
Besides masks, dialog variables may also appear in the operands of a call to a dialog manager service and in message texts. Such dialog variables are replaced by their corresponding values before the service is executed or the message is output.

The dialog variables of the application profile are retained beyond the scope of their current use and are immediately available for the next start of the dialog program (by the same terminal user).

Dialog variables can also be used for communication between and among different application segments.

Advantages of dialog variables

All FHS applications until FHS V8.0 required predefined areas (addressing aids) for the exchange of data between the application program and mask fields. The use of dialog variables for data transport, by contrast, has the following advantages:

– Data can be processed and prepared in the application program independently of the arrangement and layout of its presentation. In other words, program data can be optimally defined for each respective programming language without any constraints imposed by the format definition (e.g. sequence of mask fields).

– It is not necessary to provide data in a specific data area for each format displayed. The variable pool can be filled at different times and is available for every format display, thus eliminating the timing constraints for the provision of data.

– Convenient list displays can be easily implemented using dialog variables. This facility is an important element of the presentation.

– Dialog variables enable the substitution of variable texts in messages and help texts. This is not possible with addressing aids, since the time of substitution is not known to the application program.

– Dialog variables support automatic type conversion of data to be displayed (e.g. if a C string in the program is to be output in a mask field). With addressing aids, by contrast, the application program must perform such conversions itself.

– Permanent storage of dialog variables (memory) is easily implemented.

– Implicit dialog variables (e.g. those only specified in the format) and the transfer of their values to another format, for instance, can be implemented without application program code.

– System services (such as date, time) are easily available to the application program or mask via FHS system variables.

– Intercommunication between individual program segments is facilitated.

Types of dialog variables

Depending on their storage structure, dialog variables can be classified as:

– Elementary dialog variables
– Structures of elementary dialog variables
– Dialog variable arrays

Elementary dialog variables cannot be subdivided further; each such variable consists of a single element.

Structure of elementary dialog variables:

A structure consists of elementary dialog variables that differ in data type and/or in length. The elements of a structure must occupy contiguous storage space. The dialog variables of a structure are addressed by their names. Structures of dialog variables must be explicit dialog variables.

Example of the layout of a structure:

LAST-NAME, FIRST-NAME and CITY are the names of the dialog variables

LAST- NAME	FIRST-NAME	CITY

Dialog variable arrays

An array consists of a single array element with multiple repetitions. Array elements must occupy contiguous storage space. An array element may be an elementary dialog variable or a structure of elementary dialog variables. The dialog variables of an array element are addressed by their names and an index consisting of a positive integer.

Arrays are of particular importance for the list display facility of the dialog manager. Arrays of dialog variables can only be explicit dialog variables.

Examples of the layout of arrays:

Array of elementary dialog variables

NAME	index=1
NAME	index=2
NAME	index=3

Array of structures

LAST-NAME	FIRST-NAME	CITY	index=1
LAST-NAME	FIRST-NAME	CITY	index=2
LAST-NAME	FIRST-NAME	CITY	index=3

A dialog variable has the following attributes:

– a name,
– a value of a specific length,
– a data type, and
– possibly a repetition factor

Naming conventions for dialog variables

Since dialog variables are accessed only by name, a high degree of independence is achieved between the application program and the dialog elements (e.g. mask).

The following syntax rules must be observed when naming dialog variables:

– The name must not exceed a maximum of 255 characters.

– The first character must be a letter (A-Z).

– Subsequent characters may include any of the following:

  A-Z, 0-9, - (hyphen), . (period).

  The hyphen and period must not be immediately followed by another hyphen or period, and the last character of the name must not be a hyphen or period.

– No distinction is made between uppercase and lowercase letters.

The naming conventions for dialog variables are essentially the same as the syntax rules for variable names in SDF-P procedures, so the same mask can be used both in an S procedure and in a program.

There is, however, a difference in the significance of the period. In contrast to SDF-P variables, where the period serves to identify structures, the period in a dialog variable has no special meaning.

A name conforming to the above rules may be followed by an index entry.

Index entry

An index entry is identified by a “#” character, followed by an integer value (1 to 32767; unsigned and without digit separators) or the name of an elementary dialog variable with a value that can be converted into a 4-byte positive binary integer (of type FIXEDS).

The string following the # symbol is analyzed on reading or writing a dialog variable: if the first character is a digit, the string is interpreted as a direct index specification.

If the first character is a letter, the string is interpreted as a variable name, and a variable with the specified name is searched. This name must designate an elementary variable containing a legal value.

If the dialog variable has been defined as an array of dialog variables by using the VDEFINE service, an index may be specified in addition to the variable name.

System variables of the dialog manager (special dialog variables) begin with “SYS-“. Users should ensure that their own dialog variables do not begin with “SYS-”.

Examples of valid names for dialog variables:

```
PHONE-NUMBER
 A
 A.123#5
CITY#INDEX
```

Value of a dialog variable

The value of a dialog variable has a specific length. This length may have a fixed value, may be defined by a length specification, or be determined by an end marker. The possible values for the length of a dialog variable are contingent on their data types.

Data type of a dialog variable

The data type defines whether the value of a dialog variable is typically a string, binary number, or decimal number.

The individual data types are described under the VDEFINE service.

4.1.1.1 Explicit dialog variables

An explicit dialog variable is defined by using the VDEFINE service. The VDEFINE service is used to assign internal storage space of the application program to a dialog variable. Any change in the content of this storage space immediately changes the value of the dialog variable. All services of the dialog manager read and write explicit dialog variables by directly accessing their corresponding storage space in the program.

This allows an application program to declare a program variable as a dialog variable, to call the DISPLAY service of the dialog manager to display any mask containing this variable, and to then process this variable as usual in the program (e.g. IF NAME = “LARRY” THEN...,).

The variable pool contains the name and the address of the value of an explicit dialog variable. This entry in the variable pool can be deleted by means of the VDELETE service.

Description of an explicit dialog variable:

One or more explicit dialog variables are described in a call to the VDEFINE service by one of the following points:

– a name or a name list,
– a data type or a list of data types,
– a length or a list of length values,
– an address (the starting address of values), and
– a multiplication factor (dimension).

Names, data types and length values at the same position in lists correspond to one another.

The address of a variable of the name list is determined from the starting address and the lengths of all variables defined earlier in this VDEFINE service.

A single call to the VDEFINE service can thus be used to define individual sections of a contiguous storage area as explicit dialog variables. Other sections of this storage area can be skipped as fill areas by entering only the length of the fill area and an “*” for the name and data type in the lists.

Such fixed sequences of program variables are essentially structures in terms of programming languages.

The following diagram illustrates the concept:

var_1	var_2	fill area	var_3	...
length_1	length_2	length_fa	length_3	
Initial address				

The layout of the storage area defined by the lists is repeated as required if a value greater than 1 was specified as the multiplication factor. Each dialog variable defined in the name list is then an indexed dialog variable.

The VDEFINE service can be used to define elementary explicit dialog variables, structures of explicit dialog variables, and arrays of explicit dialog variables.

A single dialog variable from an array of dialog variables can be addressed by:
`name_of_variable#index` (e.g. VLEN(10), see next page).
The index entry identifies a specific element of an array of dialog variables. If it is omitted, the first element is assumed.

Example for the definition of multiple explicit dialog variables:

The dialog variables VI and VC are defined by means of the following C statements:

```
DMCOMM dmcomm;
char buffer[255];
long buflen;
long lfield;

long vi;
char vc [50];

strcpy (buffer “VDEFINE (VI) FORMAT(FIXEDS)“);
lfield = 4;
buflen = strlen(buffer);
ispci2(&dmcomm,&buflen,buffer,&lfield,&vi);

strcpy (buffer “VDEFINE (VC) FORMAT(BINSTR)“);
lfield = 50;
buflen = strlen(buffer);
ispci2(&dmcomm,&buflen,buffer,&lfield,&vc);
```

The variables “vi” and “vc” are not defined within a C structure, so the location of the storage space assigned to them is unknown to the programmer. For this reason, two calls to the VDEFINE service are required in order to declare them as dialog variables.

Example for the definition of an array of structures:

The following COBOL statements declare the program variables A, B, and C of structure S as explicit dialog variables:

```
01 LBUF           PIC S9(7)  COMP VALUE 512.
01 BUF            PIC X(512) VALUE SPACE.
01 LENFELD.
   02 VLEN        PIC S9(7)  COMP OCCURS 3 TIMES.
01 S.
   02 SE          OCCURS 3 TIMES.
      03 A         PIC X(10).
      03 B         PIC S9(7)  COMP.
      03 C         PIC X(2).

MOVE 10 TO VLEN(1).
MOVE  4 TO VLEN(2).
MOVE  2 TO VLEN(3).
MOVE 'VDEFINE (X Y Z) FORMAT(CHAR FIXEDS CHAR) DIM(3)
     OPTION(LIST)' TO BUF.
CALL „ISPCI2“ USING DMCOMM LBUF BUF LENFELD S.
```

In other words, the following storage layout (with the indicated offsets) is defined:

	0	10	14	16
0	A(1)	B(1)	C(1)	
16	A(2)	B(2)	C(2)	
32	A(3)	B(3)	C(3)	

The dialog variables X, Y, and Z form an array of structures and must occupy contiguous storage space.

Data types of explicit dialog variables

Each explicit dialog variable is assigned a data type via the VDEFINE service. This data type provides information on how the storage space of the dialog variable is interpreted by the application program and thus defines the internal format of the dialog variable.

Explicit dialog variables can be defined with the following data types:

– CHAR, BINSTR, NUMS, NUMU for storage of strings,

– FIXEDS, FIXEDU for storage of binary numbers,

– PACK for storage of packed decimal numbers.

When a dialog variable is accessed, type conversions may be performed by the dialog manager to convert different data types into one another.

This may be required in the following cases:

– when a VCOPY, VREPLACE, VPUT or VGET service is called;

– when a dialog variable is displayed in a mask or a message.

4.1.1.2 Implicit dialog variables

If the dialog manager needs to write a dialog variable that is not explicitly declared by the application program, it defines a dialog variable implicitly, i.e. enters the name and the value in the function pool (or in the profile pool). A read operation on an undefined dialog variable returns the length 0 and an appropriate return code.

An implicit dialog variable is a string of type CHAR or a number of type FIXEDS (4 bytes) and is compatible with the SDF-P variable of type STRING or INTEGER. The length of an implicit dialog variable of type CHAR is restricted to 16383 characters.

Implicit dialog variables can only be processed by means of variable and display services.

The VCOPY and VREPLACE services can be used by a program to copy the value of an implicit dialog variable into a program area and to modify the value or implicitly create the dialog variable, respectively.

If an input field of a mask is associated with an implicit dialog variable, the value of the variable may be changed by input from the keyboard.

An implicit dialog variable is always an elementary dialog variable. Implicit dialog variables of type CHAR are always stored with a length based on their significant characters (i.e. without trailing blanks).

Dialog variables with identical names

The names of dialog variables in a variable pool must be unique. Potential violations of this rule could occur in the following cases:

– If an explicit dialog variable and an implicit dialog variable are given the same name. This situation occurs when the VDEFINE service is used to define a dialog variable with a name that already exists for an implicit dialog variable. If VDEFINE is called with the COPY operand, the value of the implicit dialog variable is copied, and the implicit dialog variable is deleted. Without the COPY operand, the VDEFINE call terminates with an error code.

– If two explicit dialog variables are to be assigned the same name, the VDEFINE call will terminate with an error code.

– It is not possible for two implicit dialog variables to have the same name, since only the value is changed.

4.1.1.3 Rules for dialog variables

The following rules apply in connection with dialog variables:

– Implicit dialog variables of type FIXEDS always have a length of 4 bytes.

– A dialog variable of type FIXEDS is created implicitly even if it has the value 0.

– A CHAR/BINSTR value becomes an implicit dialog variable only if the relevant length of the source value is not 0.

  Relevant length: length of the value of a dialog variable of type CHAR/BINSTR without the insignificant trailing blanks. The NOBSCAN operand in the explicit definition of a dialog variable (VDEFINE) determines whether or not trailing blanks are significant. There are no significant trailing blanks for implicit dialog variables (since trailing blanks are truncated on storage).

– The length of an existing implicit dialog variable of type CHAR is changed at the time of an update (VREPLACE) in accordance with the relevant length (i.e. truncated or extended).

– An implicit dialog variable of type FIXEDS can only be generated by VREPLACE (not by mask fields).

– VCOPY for CHAR/BINSTR as the target type returns the relevant length and thus changes the target field length defined by the user. The specified target field length is not changed for other types.

– Profile dialog variables can be of type CHAR or FIXEDS.

– Implicit dialog variables can only be deleted explicitly.

– It is best to avoid frequent type conversions in the interests of clarity. If numeric values are present and the value range is sufficient, the use of FIXEDS should be given precedence.

4.1.1.4 System variables of the dialog manager

System variables contain globally applicable data of the dialog manager or the operating system (e.g. the current date) and are stored in the function pool or in a special system pool.

The names of system variables of the dialog manager begin with the string "„SYS-".

Some system variables (e.g. time) are determined on a current basis. System variables that are contained in the function pool are treated as normal dialog variables, i.e. the variable is defined if no variable of that name exists, and if a variable of the same name is already present, then the existing variable is used regardless of whether an implicit or explicit dialog variable is involved. An overview of system variables can be found in the appendix on page 261.

4.1.1.5 Conventions for dialog variables in the model line (list processing)

Name without index entry: e.g. ABC or A.B-C

Variable definition in the TIAM application program:

```
VDEFINE (ABC,A.B-C) FORMAT(CHAR) DIM(100)
```

The term dialog variables is used in this context to avoid confusion.

4.2 Variable pool

Dialog variables are stored in variable pools as (name,value) or (name,value-address) combinations. Special variable pools containing the dialog variables of the application profile are stored in a profile library.

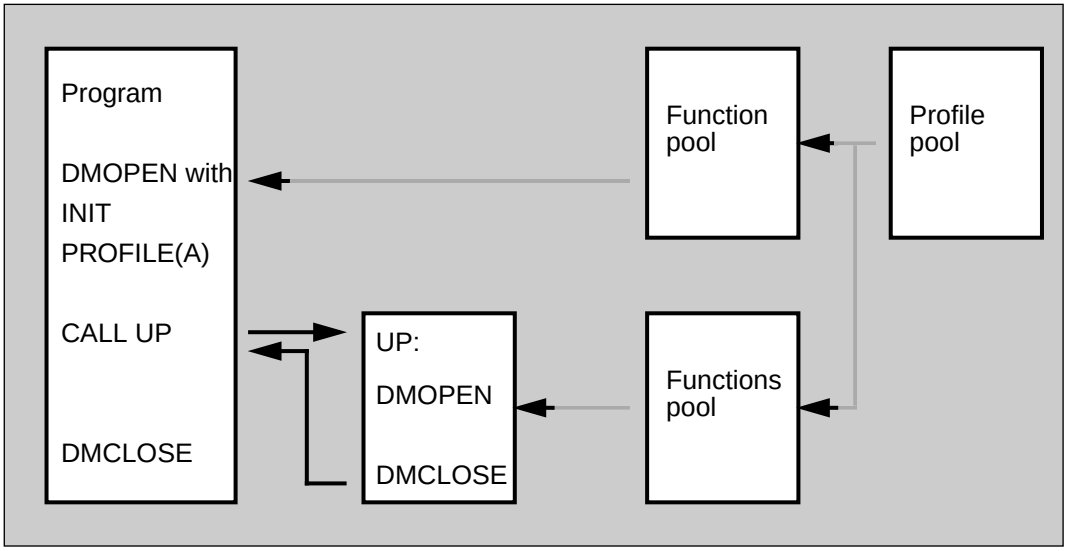
Dialog variables for an application program are defined or created by the variable services of the dialog manager. They are maintained in variable pools.

Function pool and profile pool

The dialog manager distinguishes two types of variable pools:

– function pool
– profile pool

The following diagram illustrates the inter-relationships. It shows the flow of execution of an application program in which a subprogram (SP) is active and indicates when each type of variable pool is made available (dotted line).



The function and profile pools are part of a two-level pool hierarchy in which the function pool is searched first when reading a dialog variable and the profile pool is searched thereafter if no dialog variable was found.

The existence of the function pool begins with the execution of a DMOPEN service and ends with the execution of the associated DMCLOSE service.

When a DMOPEN service is executed with the INIT operand, a profile pool can be created in addition to the function pool with the aid of the PROFILE operand.

The names and values of profile dialog variables are stored in a member of the user-specific profile library. If this member exists, the profile pool being opened is initialized with the elementary dialog variables contained in that member; otherwise, the application begins with an empty profile pool.

The values and names of the dialog variables of the profile pool are output to the library member on closing the pool, i.e. on terminating access to the dialog manager (DMCLOSE service). A profile pool always contains implicit dialog variables (i.e. the pool contains the name and value of each variable) that have values of type CHAR or FIXEDS.

If the NOSAVE operand is specified on DMCLOSE, the profile pool is not written.

The VDELETE service can be used to remove definitions of dialog variables from the function pool. An explicit dialog variable must be deleted whenever the addressed storage space is no longer available.

Dialog variables can be assigned values by means of the VREPLACE service; if a given variable does not exist in the function pool, it is automatically defined as an implicit dialog variable. Implicit dialog variables in the function pool can also be defined by the DISPLAY service.

The values of dialog variables (e.g. of implicit variables) can be read by means of the VCOPY service.

The profile variables of the dialog manager which are stored in an element are compatible with a profile element generated by SDF-P (as of Version 2.0). An FHS profile element can therefore be processed by using SDF-P commands (or an S procedure). In the following example, a profile element with the name PROF1 is processed in the library, PROFLIB:

```
/OPEN-VARIABLE-CONTAINER PROF1,*LIB(PROFLIB,PROF1),
   AUTOMATIC-DECLARE=*
```
This SDF-P command opens an SDF-P variable container with the name PROF1 and “imports” all variables.

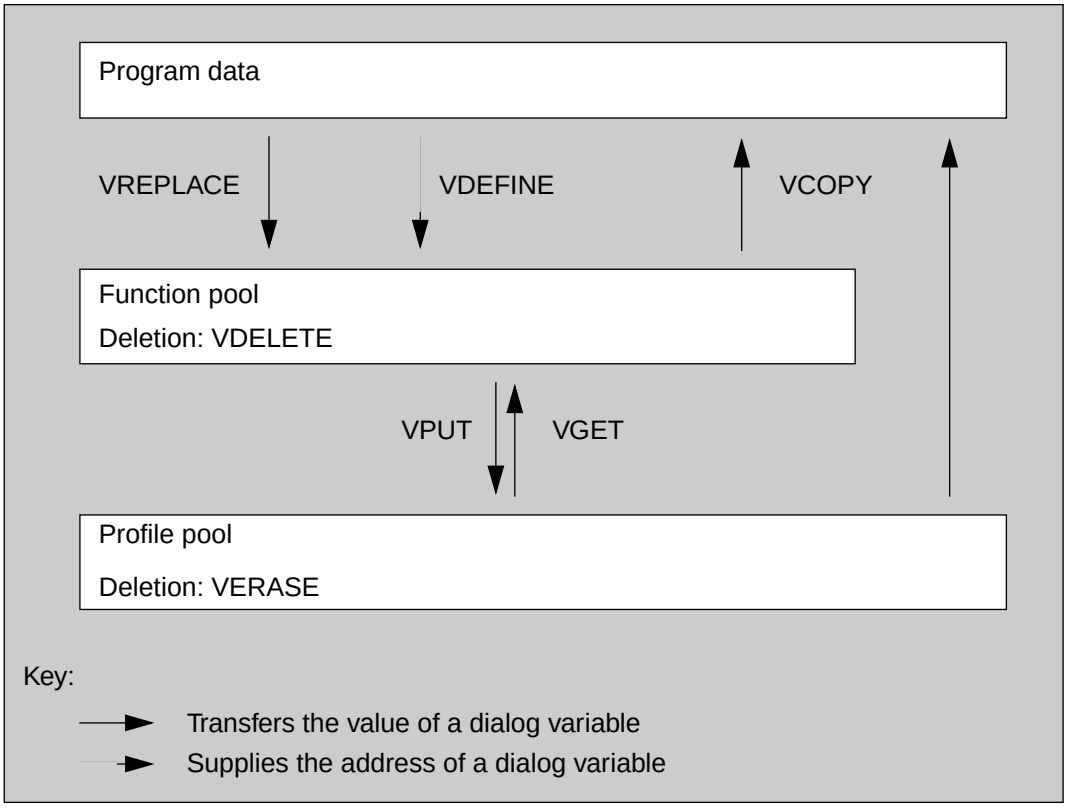
```
/SHOW-VARIABLE *ALL
```
The variables are displayed.

```
/variable-name = 'new_value'
```
A single variable is modified.

```
/SAVE-VARIABLE-CONTAINER PROF1
```
The modified profile element is saved.

Notes:
In order to process an FHS profile with SDF-P, the syntax of the variable names must correspond to that of SDF-P.

The following diagram demonstrates which services are used for data communications between the variable pools or program areas and the variable pools in which the deletion of dialog variables occurs.



The VPUT service can be used to create a dialog variable or change its value in the profile pool.The current value of the dialog variable is copied from the function pool into the profile pool without changing the dialog variable in the function pool. The VGET service can be used to copy dialog variables from the profile pool to the function pool without changing the definitions and values in the profile pool. Dialog variables in the profile pool are deleted using the VERASE service.

4.3 Program structure of a dialog application

Every dialog manager application initiates communication with the dialog manager by a call to the DMOPEN service and ends communication with a DMCLOSE call. Other services may be invoked between these two calls.

The DMOPEN call initializes the connection to the dialog manager. The application program cannot invoke any other dialog service unless the DMOPEN call was successful.

The INIT operand must be specified in the first DMOPEN call of a dialog manager application. This causes the communication area (DMCOMM), which must be supplied by the application program as a transfer parameter, to be assigned a non-initialized system section.

This first DMOPEN call requests global storage space for subsequent operations of the dialog manager and places the address of this area in the system section of the communication area. In addition, a flag is entered in the communication area to distinguish this DMCOMM from others. These entries must not be altered by the application program.

The application flag of the PROFILE operand in the first DMOPEN call can be used to optionally open a profile pool. Every DMOPEN call opens a new function pool that is concatenated with the profile pool. Concatenation means that the function pool is searched first when dialog variables are to be read and, if the variable is not found there, the profile pool is searched thereafter.

All calls for dialog services that follow the first DMOPEN call must use the communication area initialized by the dialog manager (see page 118). This ensures that such services are connected with a function pool and the profile pool concatenated with it.

The DMCLOSE service is also called with the communication area that was initialized with DMOPEN. DMCLOSE closes the corresponding function pool and writes back the dialog variables of the profile pool to the profile library. The global storage space of the dialog manager is released.

Multiple DMOPEN calls

Each application normally contains only one DMOPEN/DMCLOSE call. More complex applications may, however, call subroutines which, in turn, can also include DMOPEN/ DMCLOSE calls. It is also possible to call a BS2000 system subprogram that invokes dialog services. This results in the creation of a hierarchy of DMOPEN calls.

The use of such multiple DMOPEN calls is explained below.

– The first DMOPEN call of an application must include the INIT operand and thus open a dialog complex. If profile variables are to be used by the dialog complex, the PROFILE operand must be specified.
  This causes the dialog manager to open a function pool and a profile pool and to place global information in the system section of the passed DMCOMM. The profile pool is available to all dialog manager services of this dialog complex and is not closed until a DMCLOSE to end the dialog complex is executed.

– A dialog complex may, for example, include subroutines that wish to use a separate function pool. This is enabled by a DMOPEN call without the INIT operand. A DMOPEN without INIT opens a dialog section. In the case of such DMOPEN calls without the INIT operand, the associated DMCOMM must contain a valid system section, so the application program must copy the system section of the DMCOMM of the first DMOPEN if a separate DMCOMM is to be used for each DMOPEN.
  The provision of a separate DMCOMM at each DMOPEN is not required for strictly nested multiple DMOPEN calls; the DMCOMM of the first DMOPEN can be used for all such calls. The dialog manager retains the existing system section at the DMOPEN and saves it back at the next DMCLOSE. In this case, however, it is only possible to work with the last opened dialog section. If different function pools are to be used simultaneously, the concurrent existence of multiple communication areas is essential, i.e. copies must be passed to each DMOPEN.
  A DMCLOSE with the communication area of the dialog complex (with INIT) will also terminate all subordinate dialog sections.
  Each DMOPEN call in the same dialog complex opens a separate function pool and concatenates it with the profile pool of the dialog complex.

– DMCLOSE releases the function pool requested with DMOPEN. The DMCLOSE associated with the first DMOPEN also releases the profile pool and writes the profile variables into the profile library. Global storage is likewise released, and the dialog complex is thus terminated. No further dialog service for that complex can then be requested.

– Multiple DMOPEN calls with the INIT operand will result in the creation of multiple dialog complexes. If such dialog complexes exist in parallel, dialog variables cannot be exchanged among them via variable pools, i.e. each dialog complex is clearly delineated from other dialog complexes and is handled independently by the dialog manager.

– If multiple dialog complexes are created with the same application flag in the PROFILE operand, there will also be multiple profile pools corresponding to the same member of the profile library. In such cases, closing the pool may cause the profile member contents created by some other dialog complex to be overwritten.

Example of a dialog application:

Dialog complex 1

```
ispci (dmcomm1,buflen,"DMOPEN INIT PROFILE(ABC)");

call subr1
   Pass dmcomm1

call dialogcomplex2

ispci (dmcomm1,buflen,"DMCLOSE");
```

subr 1

```
Transfer system section to dmcomm2
ispci (dmcomm1,buflen,"DMOPEN")
ispci (dmcomm1,buflen,"DISPLAY...")
ispci (dmcomm1,buflen,"DMCLOSE")
```

Dialog complex 2 (without profile pool)

```
ispci (dmcomm2,buflen,"DMOPEN INIT")
ispci (dmcomm2,buflen,"DISPLAY...")
ispci (dmcomm2,buflen,"DMCLOSE")
```

Example of a dialog application with parallel dialog sections:

Dialog complex 1

```
ispci (dmcomm1,buflen,"DMOPEN INIT,PROFILE(ABC)");   Opens the
                                                     dialog complex
ispci (dmcomm1,buflen,"DISPLAY...")                  DISPLAY call

dmcomm2 = dmcomm1
dmcomm3 = dmcomm1                                    Copies the DMCOMM

ispci (dmcomm2,buflen,"DMOPEN")                      Opens both dialog sections
ispci (dmcomm3,buflen,"DMOPEN")                      Creates function pools 2 and 3


call subr1                                           Alternate operations in different
   Pass dmcomm 2                                     dialog sections and pools
call subr2
   Pass dmcomm 3
 ispci (dmcomm1,buflen,"DISPLAY...")
call subr1
   Pass dmcomm 2
 ispci (dmcomm2,buflen,"DISPLAY...")
call subr2
   Pass dmcomm 3

call subr2
ispci (dmcomm1,buflen,"DMCLOSE");                    Closes the dialog complex
                                                     including the sections
```

subr1(parameter par_dmcomm)

```
ispci (par_dmcomm,buflen,"VDEFINE...")               Work with function pool 2
ispci (par_dmcomm,buflen,"DISPLAY...")
```

subr2 (parameter par_dmcomm)

```
ispci (par_dmcomm,buflen,"VCOPY.....")               Work with function pool 3
ispci (par_dmcomm,buflen,"DISPLAY...")
```

4.4 Calling dialog services

Dialog manager services are available to application programs.

The general format of calls within a program and the types and notation of parameters and operands are described in this section, but without specific details on every individual parameter or operand.

The dialog services and their parameters and operands are described in alphabetical order starting on page 125.

4.4.1 ISPCI interface routine

Dialog manager services are invoked in a program as a CALL interface via the ISPCI (3rd. parameter) or ISPCI2 (5th. parameter) interface routine. This interface is based on the international recommendations for a dialog interface (SAA CPI - systems application architecture common programming interface). The following method is used for passing parameters at this interface:

Register 1 -->	A(communication area)	
	A(length of operand area)	
	A(operand area)	
	A(length area)	1)
	A(value area)	1)

1) These parameters are only required by some services.The entry point ISPCI2 of the interface routine must be used for the call to these services. More details on the meaning of these additional parameters can be found under the description of the corresponding service. This method of passing parameters when calling subprograms is implemented, or can at least be requested, in all compilers (CALL statement in COBOL and FORTRAN, function call in C).

Storage space for the communication area and the operand area must be supplied by the application program.

The operand area is a storage area that contains the name and the operands of the dialog service. A detailed description of the operand area is provided in the sections describing the individual dialog services.

The length of the operand area in bytes must be specified as a binary signed integer of 4-byte length (word boundary). The maximum length is equal to 512 bytes. Only legal characters are permitted in the operand area itself (e.g. X’00’ would lead to an error).

Before the interface routine is called, the operand area must be filled with the name and the operands of the desired service. These specifications are given as strings, thus enabling a simple definition in the usual programming languages.

The specified length of the operand area may be greater than the number of characters entered in it if the operand area is padded with blanks to the specified length.

The following example shows a COBOL application program section that opens access to the dialog manager and then displays a format:

```
01 DMCOMM       PIC X(128) VALUE SPACE.
01 BUFLEN       PIC S9(7) COMP VALUE 512.
01 BUFFER       PIC X(512) VALUE SPACE.

MOVE „DMOPEN INIT PROFILE(ABC)“ TO BUFFER.
CALL „ISPCI“ USING DMCOMM BUFLEN BUFFER.

MOVE „DISPLAY PANEL( FORM001 )“ TO BUFFER.
CALL „ISPCI“ USING DMCOMM BUFLEN BUFFER.
```

Further examples on calling dialog services are provided under the individual descriptions of services.

4.4.2 Communication area

The communication area contains data that is used for communications between the program and dialog manager. It is a storage area of the application program.

The communication area (DMCOMM) is 128 bytes long and has the following structure:

Offset	Length	Name	Meaning
0	8	DMRC	Return code
0	2	DMSC2	Subcode2 (reserved)
2	2	DMSC1	Subcode1 (error class)
4	4	DMMC	Main code (name value)
8	8	DMMSGID	Message key
16	8	DMFLAG	Flag for special services
16	1	DMERR	Flag for error control
24	104	DMSYS	System section

The components of the communication area have the following significance:

DMRC
The return code provides information on whether the called service was successfully or incorrectly executed. Return information is saved by the dialog manager in the field DMRC and in the sub-fields DMMC and DMSC1, respectively. Field DMSC2 must be treated as reserved. SC2 is only significant for the user if SC1 has the value 0. Its value is 2 in the case of a warning.

The main code is the main value of the return code.

Subcode1 indicates the error class to which the main value belongs.

The following error classes exist (decimal value of subcode1):

0 successful execution (including warnings)

1 parameter error (syntax error or no meaningful values)

32 internal error of the dialog manager

64 other errors

The possible values of the main code of error class 0 are given in the description of dialog services. All return codes for unsuccessful execution of a service are included in the appendix.

DMMSGID
DMMSGID can contain a message key after execution of a service. The associated message explains the cause of the error for a given error code. If the return code is zero, DMMSGID contains blanks.
If error control by the dialog manager has been set and an error leading to the output of the error mask occurs, the message text will already be displayed in the mask.

DMFLAG
The content of this field is used as follows:

DMERR	Switch to turn on/turn off error control
Y	Error control for errors
W	Error control for warnings and for errors
N (or blanks)	No error control

Recommended setting: Y

DMSYS
Contains the system section of the communication area. This information must not be destroyed by application programs.

Use of the communication area

The communication area must be specified in a program as a parameter when calling the interface routine ISPCI. It can be defined in programming languages as shown for some of the typical programming languages in the examples below. Analogous methods may be used in other languages if available.

Communication area in COBOL:

```
DATA DIVISION
WORKING-STORAGE SECTION
01 DMCOMM    IS COMP.
  02 DMRC.
    03 DMSC2 PIC S9(4).
    03 DMSC1 PIC S9(4).
    03 DMMC PIC S9(7).
  02 DMMSGID PIC X(8).
  02 DMFLAG.
    03 DMERR PIC X(1).
    03 FILLER PIC X(7).
  02 DMSYS PIC X(104).
```

Communication area in C:

```
typedef struct
    {
    short int         sc2;
    short int         sc1;
    long  int          mc;
    char               msgid[8];
    char               dmerr;
    char               filler[7];
    char               sys[104];
    } DMCOMM_T;
DMCOMM_T dmcomm;
```

Communication area in FORTRAN:

```
C  Definition as INTEGER*2 Field with 64 Elements
   INTEGER*2     DMCOMM(64)
   INTEGER*2     DMSC2,DMSC1
   INTEGER*4     DMMC
   EQUIVALENCE   (DMCOMM(1),DMSC2),(DMCOMM(2),DMSC1)
   EQUIVALENCE   (DMCOMM(3),DMMC)
   CHARACTER*8   DMMSGID
   EQUIVALENCE   (DMCOMM(5),DMMSGID)
   CHARACTER*8   DMFLAG
   CHARACTER*1   DMERR
   EQUIVALENCE   (DMCOMM(9),DMFLAG),(DMCOMM(9),DMERR)
   CHARACTER*104 DMSYS
   EQUIVALENCE   (DMCOMM(13),DMSYS)
```

4.4.3 Error handling by the dialog manager

Dialog services place a return code in the communication area. This code contains information on the successful or unsuccessful execution of the dialog service.

The return code must be evaluated in the application program on each return from a dialog service. This evaluation is simplified by the subdivision of return codes into classification codes and description codes.

The dialog manager offers application programmers the option of displaying information on errors (i.e. an error report) in a mask in the event of an error.

The following conditions must be satisfied for the dialog manager to display an error mask:

1. The DMERR switch (in DMCOMM) must be set to Y or W (recommended setting: Y).
2. Classification code SC1 must have a non-zero value (error) or SC1=0 and SC2=2 (Warning).
3. The error situation must not prevent operability of the dialog manager.

An error description is output as a mask in the main window (full-screen). The mask contains at least the following information:

– the operand area supplied by the user, following variable substitution by the dialog manager
– return code
– a description of the error

The description is contained in the message associated with the message code in the communication area. Message codes (also called message IDs) are listed in the appendix starting on page 234.

After the error is displayed, the application program may abort execution of the application or continue with it if meaningful. If an error code > 200000 occurs, no error mask can be shown.

The system command ABORT can be entered in the command input field of the error mask. This causes the main code in DMCOMM to be set to 399999. The application program should respond to this code as required.

4.4.4 Structure of the operand area

The operand area must contain the name of the dialog service. It may include both positional and keyword operands.

Positional operand

A positional operand is required for some variable services. It contains a list of names enclosed within parentheses. One or more blanks, a comma, or blanks and commas may be used as separators in the list. If the name list consists of only one name, the parentheses may be omitted.

Examples:

The operand areas specified by the following pairs have identical contents:

```
VDELETE (V111,V222,V333)
VDELETE ( V111 V222 V333 )
```

```
VDELETE ABC
VDELETE (ABC)
```

Keyword operands

A keyword operand consists of a string to identify the operand and possibly a value. The value is entered in parentheses after the keyword. (The parentheses may be entered directly after the keyword or be separated from it by blanks.) If the value in parentheses consists of only blanks, the operand is treated as unspecified.

Some operands allow a list of values to be specified. The values in this list are entered without additional parentheses and may also be separated by blanks, a comma, or a comma enclosed within blanks. Keyword operands may be specified in any order.

4.4.5 Variable substitution in the operand area and in messages

If a string in the operand area or in the text of a message begins with the character “&”, the string following that character is seen as the name of a dialog variable. The dialog manager then assumes that the value of this dialog variable should be substituted at this position in the operand area.

The following syntax applies to variable substitutions:

```
&(varname) or &(varname#index)
```

“varname” is the name of a dialog variable (or the name of an SDF-P application). The dialog variable may be of any type that is convertible to a string. The substituted value is always a string.

The parentheses may be omitted if the name of the dialog variable does not contain a period and no index is specified. In this case, the end of the name is identified by any character that does not conform to the rules for naming dialog variables or by a period. The period has a special significance in such cases: it is used only as an end marker and is dropped (compatible with earlier variable substitutions in FHS V8.0).

A single “&” followed by a blank is interpreted as the character “&” ; no substitution occurs.

A duplicated “&” (&&) becomes a “&”.

No recursive substitution occurs.

Examples:

```
'DISPLAY PANEL( ABC )'
```

A call to the display service with this operand area produces the output of format ABC.

```
'DISPLAY PANEL (&ABC)' 'DISPLAY PANEL(&(A.B))'
```

A call to the display service with this operand area outputs the format whose name is contained in dialog variable ABC or A.B.

```
'DISPLAY PANEL(&NFL#5)'
```

A call to the display service with this operand area outputs the format whose name is contained in dialog variable NFL in the 5th element of the dialog variable array.

```
'DISPLAY PANEL(&(DEF)#5)'
```

A call to the display service with this operand area outputs the format whose name is contained in the 5th array element of an array named by the value of dialog variable DEF.

The following are also valid specifications:
&ABC&DEF NO&ABC.D &ABC;TEXT &(a.b).LIST

Value substitutions are not possible in the operand area of a DMOPEN service, since it is precisely this dialog service that assigns a variable pool to the application program.

The entire operand area can be defined by a variable name.

The inserted strings are not checked for further value substitutions. The total length of the operand area after variable substitution must not exceed 2048 characters.

4.5 Description of dialog services

Each dialog service is described below with a brief description of the service, the parameters of the interface routine, and the structure and operands of the operand area. In addition, the function of the service is explained in detail, and examples on its use are provided.

If one dialog service has an effect on a following dialog service, the affected dialog service may be assumed to be in the same dialog section (same communication area) unless explicitly specified otherwise.

Dialog services have no effect on one another, but for a few restrictions (e.g. SETP; see below). When two DISPLAY calls of different dialog sections follow one another, no differential output is generated even if the same format is displayed.

Notes on usage of the term "field-name":

The name of a dialog variable to be displayed in a mask field is assigned to that field using IFG. In order to name a field in a mask, the corresponding name of the dialog variable is used as the field name. A field name that is followed by a "#" and an index designates a specific list record for a list field, since it addresses an element of an array of dialog variables.

4.5.1 ADDPOP - Define the position of a dialog box

ADDPOP instructs the dialog manager to display all following formats in a box. The box initialized by the ADDPOP call remains in effect until a REMPOP call or a further ADPOP call occurs. The effect of an ADDPOP call does not appear on the screen until the next DISPLAY call.

The size of the dialog box is determined by the size of the mask. If the size exceeds the space available at the desired position, the dialog manager will look for some other space, depending on whether or not the POPLOC operand was specified (see also the section “Creating and removing explicit dialog boxes”) on page 16.

The REMPOP service can be used to remove a dialog box. The box is actually removed at the next DISPLAY call.

An ADDPOP call may be followed by a new ADPOP call only after a DISPLAY call with the format name.

Parameters of the interface routine (ISPCI):

– communication area
– length of the operand area
– operand area

Description of parameters:

The parameters must be specified as described in the section “ISPCI interface routine” on page 116.

Structure of the operand area:

```
ADDPOP    [POPLOC{field-name }]
                 {           }
                 {*CENTRTAL  }

          [ROW(row-num)]

          [COLUMN(col-num)]
```

Operands:

ADDPOP

Name of the service

POPLOC(field-name)

POPLOC defines the reference point for field-related positioning of the dialog box.

“field-name” must be the name of a field of the current mask. The current mask is the last output format. The reference point is the first character of the specified field.

If the field name refers to a list record, an index to define a specific list record may be specified. If no index is specified in such cases, the topmost line in the current list area is assumed.

The value of the TOPINDEX variable at the time of the ADDPOP call is used to define the topmost line in the list area (top index). In the following DISPLAY call to output a mask in a dialog box, the underlying list area is positioned in accordance with the TOPINDEX specification (see also “List processing, TOPINDEX variable”).

If several fields of the same name exist, the first field is used.

*CENTRAL specifies that the box is to be output centered in the middle of the screen. The ROW and COLUMN operands are ignored in this case.

If the POPLOC operand is omitted, the upper left corner (initial point) of the current mask is taken as the reference point.

If the POPLOC operand is specified without a field name, a box may be output on an “empty” screen, i.e. no full-screen format is required.

ROW(row-num)

Specifies the offset of the dialog box in the downward (positive value) or upward (negative value) direction in relation to the reference point.

If the ROW operand is omitted, a downward shift of 2 lines occurs (default offset).

COLUMN(col-num)

Specifies the offset of the dialog box to the right (positive value) or left (negative value) of the reference point.

If the COLUMN operand is omitted, a shift of 2 columns to the right occurs (default offset).

Example:

The following COBOL statements output the format FORM002 in a dialog box that overlays format FORM001:

```
01 BUFLEN      PIC S9(7) COMP VALUE 512.
01 BUFFER      PIC X(512) VALUE SPACE.

MOVE „DISPLAY PANEL( FORM001 )“ TO BUFFER.
CALL „ISPCI“ USING DMCOMM BUFLEN BUFFER.

MOVE „ADDPOP“ TO BUFFER.
CALL „ISPCI“ USING DMCOMM BUFLEN BUFFER.

MOVE „DISPLAY PANEL( FORM002 )“ TO BUFFER.
CALL „ISPCI“ USING DMCOMM BUFLEN BUFFER.
```

4.5.2 ATTR - Define dynamic attributes for mask fields

The field attributes of mask fields (e.g. brightness, color, etc.) are generally static and are set when defining a format using IFG. These attributes are stored in the format definition. In some situations, it may, however, be necessary for the application program to change the field attributes for specific fields.

The ATTR service enables dynamic modification of field attributes for mask fields. The effect of an ATTR call does not appear on the screen until the next DISPLAY service is executed and is only valid for that one call.

The ATTR service may be called more than once before a DISPLAY call. The individual ATTR calls are then concatenated, i.e. all defined attribute assignments are collected and take effect at the next DISPLAY call. The dynamic field attributes are given precedence over the statically defined attributes.

The DISPLAY service deletes all dynamic attribute assignments before returning to the application.

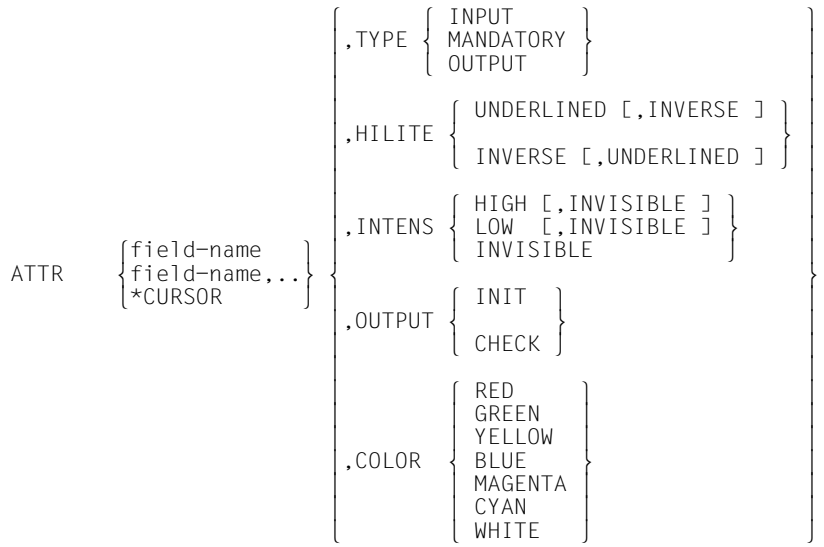
Parameters of the interface routine (ISPCI):

– communication area
– length of the operand area
– operand area

Description of parameters:

The parameters must be specified as described in the section “ISPCI interface routine” on page 116.

Structure of the operand area:



Operands:

ATTR

Name of the service

field-name

Specifies the name of a masked field for which dynamic field attributes are to be changed. A list of names (enclosed within parentheses) may be specified if the same attributes apply to all.

If the field name refers to a list record, an index to define a specific list record may also be specified. If the index is omitted in such cases, the topmost line in the list area is assumed (see “List processing, TOPINDEX variable”).

It is also possible to specify an asterisk (*) as an index for list fields. This causes all indices of the specified field (column of a list) to be assigned the same attributes. The asterisk cannot be specified in combination with TYPE(MANDATORY).

*CURSOR

The field attributes of the field in which the cursor appears at the next DISPLAY call are to be changed. If it is determined at the next DISPLAY call that a dynamic attribute has also been requested for the field name of the field at which the cursor was positioned, the specifications are combined. If the specifications are the same, the *CURSOR entry is given precedence.

TYPE( )

Sets the field type:

INPUT
The corresponding mask field is set as an unprotected input field without mandatory input. The field can be changed via the keyboard.

MANDATORY
Defines the mask field as mandatory, i.e. the field must be modified by the user at the terminal.

OUTPUT
The mask field is defined as an unprotected and unmarkable output field. The contents of the field cannot be changed from the keyboard.

HILITE()

Defines the type of highlighting used for the mask field:

UNDERLINED
The field contents are shown underlined.

INVERSE
The field contents are displayed in inverse video.

Note:
If the highlighting of fields has already been set in the format definition, it is not possible to undo the setting using dynamic attributes. Fields to be dynamically highlighted should therefore not be highlighted when defining the format.

INTENS()

Specifies the intensity of the field:

HIGH
The field is displayed with high intensity.

LOW
The field is displayed with low intensity.

INVISIBLE
The field is invisible and cannot be printed.

OUTPUT()

Defines the usage of defaults for mask fields.

Without the OUTPUT specification, default values of the mask are used if it is determined in the DISPLAY call that the dialog variable assigned to the mask field does not exist or that its value has a relevant length of zero. On returning from the DISPLAY service, the corresponding dialog variables will contain the default values of each respective mask field. If a dialog variable does not exist, it is implicitly created as a variable of type CHAR.
Default settings for list fields are ignored.

INIT
When the mask is displayed, the default value that was set when defining the format is output for the specified field regardless of the contents of the dialog variable connected with that mask field. If no value for the field was defined in the format, the field is shown with fill characters.

CHECK
For existing dialog variables that have values with a relevant length greater than zero, the default value is used only if the content of the dialog variable connected with the mask field has the value zero.

Without the OUTPUT operand, the default setting of a mask field is always used whenever the dialog variable connected with the mask field does not exist or has a relevant length of zero.

COLOR()

Specifies the color for the masked field. This specification is ignored for devices that cannot display colors.

RED

GREEN

YELLOW

BLUE

MAGENTA

CYAN

WHITE

Example:

The following COBOL statements change the field attributes for some mask fields temporarily.

```
01 BUFLEN      PIC S9(7) COMP VALUE 512.
01 BUFFER      PIC X(512) VALUE SPACE.

MOVE „ATTR FELD1,TYPE(MANDATORY),HILITE(UNDERLINED)“ INTO BUFFER.
CALL „ISPCI“ USING DMCOMM BUFLEN BUFFER.

MOVE „ATTR (FELD2,FELD3),TYPE(OUTPUT)“ INTO BUFFER.
CALL „ISPCI“ USING DMCOMM BUFLEN BUFFER.
```

4.5.3 CONTROL - Set operating modes

The CONTROL service is used to define specific processing options.

Parameters of the interface routine (ISPCI):

– communication area
– length of the operand area
– operand area

Description of parameters:

The parameters must be specified as described in the section “ISPCI interface routine” on page 116.

Structure of the operand area:

```
                  { REFRESH                  }
                  | SAVE                     |
                  | RESTORE                  |
CONTROL DISPLAY   { CCSN( ccsname/*EXTEND ) }
                  | NOCCSN                   |
                  { ACK                      }
```

Operands:

CONTROL

Name of the service

DISPLAY SAVE

Causes this information on the currently displayed screen to be saved. This enables a nested display of masks in the main window.

The SAVE call must follow the DISPLAY call for which the screen is to be saved. The display is restored by CONTROL DISPLAY RESTORE.

SAVE and RESTORE must be called in pairs and with the same communication area that was used for the DISPLAY call.

DISPLAY RESTORE

Restores the screen to its existing status at the time of a previous CONTROL DISPLAY SAVE. Processing can then continue as was possible before the SAVE.

DISPLAY REFRESH

Causes the screen to be fully refreshed at the next DISPLAY call (no update output). P keys and character sets are reloaded as required. This specification is only applicable to the next display call in the dialog complex independent of the dialog section (communication area).

DISPLAY ACK

Causes the characters“*** ” to be displayed as an input prompt in line mode before the output of a mask at the next DISPLAY call. It is only after the requested input is made that a fully refreshed screen (no update output) is displayed. This enables the application programmer to prevent messages that were not output by the dialog manager from being overwritten before the operator can take note of them.

P keys and character sets are not reloaded (cf. REFRESH).

Causes the screen to be fully refreshed at the next DISPLAY call (no differential output). P keys and character sets are reloaded as required. This specification is only applicable to the next display call in the dialog complex independent of the dialog section (communication area).

DISPLAY CCSN(ccsname)

Causes the specified ccsname to be used as the CCSNAME (Coded Character Set Name) for all subsequent DISPLAY calls. *EXTEND specifies that the currently valid CCSN for the terminal is to be used. If a CCSNAME for a format was set when defining the format and if that name differs from the specified name, a warning is issued by the DISPLAY call. The CCSNAME of the CCSN operand is given precedence.

This specification is only valid for the current dialog section and remains in effect until a new CONTROL call with CCSN or NOCCSN and the same communication area occurs.

A check to determine if the CCSNAME is known or the XHCS subsystem is loaded does not occur until the next DISPLAY call.

DISPLAY NOCCSN

Cancels the definition of a CCSN. This entry is related to the communication area and remains in effect until a new CONTROL call with CCSN or NOCCSN and the same communication area occurs.

Example:

The following COBOL statements are used to set specific options:

```
01 BUFLEN      PIC S9(7) COMP VALUE 25.
01 BUFFER      PIC X(25) VALUE SPACE.

MOVE „CONTROL DISPLAY SAVE“ INTO BUFFER.
CALL „ISPCI“ USING DMCOMM BUFLEN BUFFER.

MOVE „CONTROL DISPLAY RESTORE“ INTO BUFFER.
CALL „ISPCI“ USING DMCOMM BUFLEN BUFFER.
```

4.5.4 DISPLAY - Display a format and/or a message

The DISPLAY service is used to display a format and/or a message. The format definition is, however, only read from the format library if the PANEL operand has been specified. Once a format is loaded, it remains loaded for the entire dialog complex.

Several different options are available to the terminal user after the display. For example, the user can change field contents and thus the values of dialog variables, initiate the output of help panels, or request actions via the menu bar. It is only when all the user actions and all the validation checks and processes specified in the format definition are completed that the display service is terminated.

For information on how default values of a mask field are processed, see the ATTR service (OUTPUT-Operand).

The following table explains the execution sequence of the DISPLAY service in combination with the PANEL or MSG operands.

PANEL	**MSG**	**Execution sequence**
yes	no	Reads the format definition and the dialog variables Displays the format
yes	yes	Reads the format definition and the dialog variables Reads the message Displays the format with the message in the message area or in a message box
no	yes	Reads the message Repeats the last displayed screen without re-reading the dialog variables (including control variables) and overlays a message in the message area or in a message box
no	no	Repeats the last displayed screen without re-reading the dialog variables (including control variables)

The DISPLAY service facilities can be exited

– by pressing the ENTER key, provided the cursor is not in the menu bar and no system command is in the command area,
– by entering a CANCEL or EXIT command, which may also be assigned to function keys,
– by an application command that is assigned to a function key.

Parameters of the interface routine (ISPCI):

– communication area
– length of the operand area
– operand area

Description of parameters:

The parameters must be specified as described in the section “ISPCI interface routine” on page 116.

Structure of the operand area:

```
DISPLAY    [ PANEL( format-name ) ]

                                    { field-name }
           [  MSG ( msgid ) [ MSGLOC{ *CENTRAL   } ]]
                                    { $lll#ccc   }

           [  CURSOR( field-name ) [ CSRPOS( pos ) ]]

           [  LOCK ] [ ALARM ] [ HARDCOPY ]

           [  MANDATORY ] [ NOAUTOTAB ]

           [  NODISPLAY ]
```

Operands:

DISPLAY

Name of the service

PANEL(format-name)

Name of the format to be displayed. If this operand is omitted, the last format specified in the PANEL operand is reactivated without reading the dialog variables again, i.e. the previously displayed values are shown.

MSG(msgid)

Message key of a message to be shown together with the format.

MSGLOC(field-name)

Name of a mask field to be used for positioning a message box (see also “Output of messages”). If the field name refers to a field of a list area, an index to define a specific list record may also be specified (see also “List processing”).

The field must be contained in the format that is specified in the PANEL operand or, if no PANEL operand is specified, in the format that was displayed by the last DISPLAY call with the PANEL operand.

If multiple fields of the specified name exist in the format, the first field is used.

A message box is output 2 characters to the right and 2 lines below the field specified in the MSGLOC operand. If the space below the defined field is insufficient for the message to be output, the dialog manager will search for another position using a specific algorithm (see also “Output of messages”).

If *CENTRAL is specified as the field name, a message box is output in the center of the screen.

The $lll#ccc specification allows absolute positioning of the message box.

If the MSGLOC operand is omitted, the message box is placed with reference to the cursor position.

CURSOR(field-name)

Name of the field in which the cursor is to be placed.

If multiple fields of the specified name exist in the format, the first field is used.

The specified name can only be the name of an input field or of a markable text field.

If the CURSOR operand is omitted, the cursor is placed in the first input field of the format.

If the field name refers to a field of a list area, an index to define a specific list record may be specified, e.g. CURSOR(IN-FELD#3) (see also “List processing”). If no index is specified in such cases, the topmost line in the list area is assumed.

An absolute cursor position in the form $lll#ccc may be specified instead of the field name, where “lll“ is the absolute line position on the screen and “ccc” the absolute column specification.

CSRPOS(pos)

Position of the cursor in the field defined by the CURSOR operand. The first character in the field is at position 1.

The CSRPOS operand is only significant if the value in the CURSOR operand is the name of an input field.

If CSRPOS is omitted or has an illegal value (less than 1 or greater than the field length), a value of 1 is assumed.

LOCK

If this operand is specified, control is returned to the application program immediately after the format is displayed. This function can be typically used to display a logo or a “Please wait” message.

ALARM

This operand, if specified, issues an acoustic signal when the format is displayed.

HARDCOPY

The screen contents are automatically output on an existing hardcopy device.

MANDATORY

All mask fields with the static attribute “mandatory input” are reset. This is useful in cases where the input requirement has been satisfied after a DISPLAY call (and is therefore no longer in effect) but needs to be reactivated for all fields when the same format is re-displayed. This specification has no effect for list fields.

NOAUTOTAB

The cursor can be moved to protected fields of the screen by using arrow keys. If NOAUTOTAB is not specified, the cursor can only be moved to unprotected and markable fields of the screen.

NODISPLAY

The screen is created internally exactly as if NODISPLAY were omitted; however, it is not transferred to the terminal. The DISPLAY call is terminated as if the ENTER key were pressed. If the command area was preset with an FHS system command, the command is executed, but errors, if any, are ignored.

Example:

The following COBOL statements display the format FORM001 and the message MSGI123:

```
01 BUFLEN      PIC S9(7) COMP VALUE 512.
01 BUFFER      PIC X(512) VALUE SPACE.

MOVE „DISPLAY PANEL(FORM001) MSG(MSGI123) CURSOR(FELD1)“ TO BUFFER.
CALL „ISPCI“ USING DMCOMM BUFLEN BUFFER.
```

4.5.5 DMCLOSE - Terminate use of dialog services

The DMCLOSE service terminates access to dialog services initialized by DMOPEN.

The DMCLOSE call releases the function pool for which a flag is entered in the system section of the communication area. This communication area is passed to the dialog manager with the DMCLOSE call. If a profile pool was opened by the associated DMOPEN call (DMOPEN INIT PROFILE(...)), and if NOSAVE was not explicitly specified, the profile variables are written back to the profile library, and all storage areas of the dialog manager for the dialog complex and the profile pool are released. No further dialog service, except for a new DMOPEN, may then be executed.

Parameters of the interface routine (ISPCI):

– communication area
– length of the operand area
– operand area

Description of parameters:

The parameters must be specified as described in the section “ISPCI interface routine” on page 116.

Structure of the operand area:

```
DMCLOSE [ NOSAVE ]
```

Operands:

DMCLOSE
Name of the service

NOSAVE
This entry may only be specified for a final DMCLOSE that terminates the dialog complex (analogous to DMOPEN INIT); it is otherwise ignored. NOSAVE ensures that the profile pool is not written back to the profile library.

Example:

The following COBOL statements terminate the dialog that has the same communication area as specified in the COBOL variable DMCOMM:

```
01 BUFLEN      PIC S9(7) COMP VALUE 512.
01 BUFFER      PIC X(512) VALUE SPACE.

MOVE „DMCLOSE“ TO BUFFER.
CALL „ISPCI“ USING DMCOMM BUFLEN BUFFER.
```

4.5.6 DMOPEN - Begin use of dialog services

The first dialog service that is called by an application program must be the DMOPEN service with the INIT operand. This DMOPEN call opens a dialog complex (see “Opening and closing a dialog application”).

A DMOPEN call always opens a new function pool. The information on the function pool is placed in the system section of the communication area that is passed to the dialog manager with the DMOPEN call.

DMOPEN can also be used to open a profile pool for a dialog complex. In order to do this, a personal profile library must be assigned to each terminal user by means of the file link name IDHPROF before using the dialog manager. The profile library member “member-name” contains the dialog variables of the application profile. These variables are read into the profile pool. If no such profile library member is present, processing begins with an empty profile pool. Subsequently, when a DMCLOSE service associated with the DMOPEN is executed for a dialog complex, the dialog variables of the profile pool can be written back to the existing library member (or a new library member can be created).

A DMOPEN without INIT opens a dialog section. All dialog sections of the dialog complex are implicitly closed by the DMCLOSE associated with the DMOPEN with INIT.

Every application program should call the DMCLOSE service before releasing the storage space used for the communication area.

Parameters of the interface routine (ISPCI):

– communication area
– length of the operand area
– operand area

Description of parameters:

The parameters must be specified as described in the section “ISPCI interface routine” on page 116.

If the INIT operand is not specified, the system section of the communication area must be supplied initialized.

No variable substitutions are permitted in the operand area.

Structure of the operand area:

```
DMOPEN [ INIT [ PROFILE(member-name) ]]
```

Operands:

DMOPEN

Name of the service

INIT

Opens a dialog complex. This entry must be given with the first DMOPEN.

PROFILE(member-name)

Opens a profile pool for a dialog complex in addition to the function pool. “member-name” specifies the library member of the profile library in which the profile variables can be found. The naming conventions for the profile library correspond to that of SDF-P.

If the PROFILE operand is omitted, an empty nameless profile pool is opened. This profile pool is not saved on executing DMCLOSE.

The operands must be specified in the given order.

Example:

The following COBOL statements initiate usage of the dialog manager for a dialog complex:

```
01 BUFLEN      PIC S9(7) COMP VALUE 512.
01 BUFFER      PIC X(512) VALUE SPACE.

MOVE „DMOPEN INIT PROFILE(ABC)“ TO BUFFER.
CALL „ISPCI“ USING DMCOMM BUFLEN BUFFER.
```

4.5.7 REMPOP - Remove definition of a dialog box.

The REMPOP service is used to remove the definition of the dialog box initialized by the last ADDPOP service or to remove all definitions.

This service has no effect on the current screen content. The screen is not changed until the next DISPLAY call. If the unaltered underlying mask is to be displayed, the DISPLAY service must be called without the PANEL operand.

Parameters of the interface routine (ISPCI):

– communication area
– length of the operand area
– operand area

Description of parameters:

The parameters must be specified as described in the section “ISPCI interface routine” on page 116.

Structure of the operand area:

```
REMPOP [ ALL/*ALL ]
```

Operands:

REMPOP

Name of the service

ALL or *ALL

The definitions of all dialog boxes are removed.

4.5.8 VCOPY - Copy value of a dialog variable into application program

The VCOPY service reads the value or values of one or more dialog variables and copies them sequentially to the specified target storage area. The length of the values to be written is defined by the corresponding length specification in the length area of the call. The target area is a storage area defined in the application program.

The dialog variables are located by searching the function and profile pools in order, i.e. the profile pool is searched only if no variable is found in the function pool. Special system variables are always determined on a current basis.

Special system variables are always assigned current values or copied from a system pool.

The copied value is stored in accordance with the data type specified in the FORMAT operand. If the dialog variable to be read has a data type that differs from the type specified for the target field, data conversion occurs.

The space specified by the length field is padded with fill characters if the value of the dialog variable is less than the defined length. If the value is longer, it is truncated.

Parameters of the interface routine (ISPCI):

– communication area
– length of the operand area
– operand area
– length area
– target area

Description of parameters:

The parameters must be specified as described in the section “ISPCI interface routine” on page 116.

Length area

The length area is a storage area of the application program consisting of one or more fields. Each field occupies a full word (4 bytes, word boundary) and must contain a binary signed integer.

This area must contain a value for each name of a dialog variable in the “name-list” operand and for each fill area. The n-th specification in the name list must match the n-th value in the length area.

The length entries in the length area specify the maximum length of the storage space in the “target area” that is available for the value to be copied. The target area is thus structured by means of the length specifications in the length area.

Following the VCOPY call, each respective field in the length area will contain, depending on the specified data type of the values in the target area for types CHAR and BINSTR, the length of the copied dialog variable without insignificant trailing blanks (relevant length). If the value of the dialog variable to be copied consists of only insignificant blanks (see operand NOBSCAN of the VDEFINE service) or has a length of 0 as a result of the end identifier for type BINSTR, then the relevant length is 0. For other data types, the corresponding length of field remains unchanged.

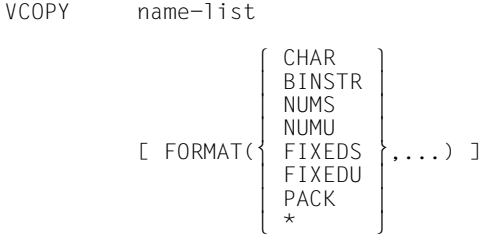
If the dialog variable exists in neither the function pool nor the profile pool, the corresponding length field will contain the value 0 after the VCOPY call, and the associated target area for the value will be padded with fill characters. In addition, a return code is set as a warning.

If an * is specified as a name in the name list, a fill area of the specified length is defined.

Target area

The target area is a storage area of the application program in which the data to be copied is to be stored. The target area is subdivided into fields for each value in accordance with the maximum length specifications in the length area. The values are stored in the specified format and padded to the maximum length of the field with fill characters if required.

Structure of the operand area:



Operands:

VCOPY

Name of the service

name-list

One or more names of dialog variables whose values are to be copied to the target area.

If a name is followed by a “#” character and an index, the value of the dialog variable of the corresponding dialog variable array element is copied.

A * in the name list defines a fill area. A * may also be entered at the same position in the list of data types. The length of the fill area is defined by a corresponding length specification in the length area.

FORMAT(data-type)

Specifies the data type for the copied value in the target area (see also VDEFINE). If the data types of the source and target do not match, a type conversion, if possible, is performed.

“data-type” may be given as a single data type specification or as a list of data types. This list must contain exactly the same number of elements as the names entered in the name list.

If only a single value is specified, that data type applies to all dialog variables.

If the FORMAT operand is permitted, data type CHAR is assumed for all values.

An * can be specified if a fill area was defined at that position in the name list.

The permitted data types for type conversions are shown in the table below:

Target field type	**Type of dialog variable to be copied (source)**						
	CHAR	**BINSTR**	**NUMS**	**NUMU**	**FIXEDS**	**FIXEDU**	**PACK**
CHAR	+	+	3)	3)	3)	3)	3)
BINSTR	+	+	3)	3)	3)	3)	3)
NUMS	1)	1)	+	+	+	+	+
NUMU	2)	2)	4)	+	4)	+	4)
FIXEDS	1)	1)	+	+	+	+	+
FIXEDU	2)	2)	4)	+	4)	+	4)
PACK	1)	1)	+	+	+	+	+

The following applies:

\+ Data conversion possible

\- Data conversion not possible

1. The value of the source field must have the following format:

   [+/-]9[9...][b...] left-justified

2. The value of the source field must have the following format:

   [+]9[9...][b...] left-justified

3. The value of the target field has the following format:

   [-]9[9...][b...] left-justified

4. Sign errors possible

   Meanings:

   9 Digits 0 - 9

   b Blanks (or NULL for BINSTR)

Examples:

The following COBOL statements supply the program with the dialog variables ST1, ST2 and the system variable SYS-CCS-NAME. The dialog variables have the following values of type CHAR:

```
ST1          : does not exist
ST2          : Db              b = blanks
SYS-CCS-NAME : EDF041bb
```

```
01 BUFLEN      PIC S9(7) COMP VALUE 100.
01 BUFFER      PIX X(100) VALUE SPACE.
```

```
01 LENBER.
   02 VLEN        PIC S9(7) COMP OCCURS 3 TIMES.
01 TARGET.
   02 DCCS-NAME   PIC X(8).
   02 VST.
      03 VST1     PIC X(6).
      03 VST2     PIC X(5).
```

```
MOVE  8 TO VLEN(1).
MOVE  6 TO VLEN(2).
MOVE  5 TO VLEN(3).
```

```
MOVE „VCOPY (SYS-CCS-NAME ST1 ST2)“ TO BUFFER.
CALL „ISPCI2“ USING DMCOMM BUFLEN BUFFER LENBER TARGET.
```

Following execution of the VCOPY service, the values are available in the COBOL variables VST1, VST2 and DCCS-NAME. These values are as follows:

```
DCCS-NAME: EDF041bb
VST1     : bbbbbb
VST2     : Dbbbb
VLEN(1)  : 6
VLEN(2)  : 0
VLEN(3)  : 1
```

4.5.9 VDEFINE - Define explicit dialog variables

The VDEFINE service is used to define an explicit dialog variable and to assign it storage space provided by the application program.

After an explicit dialog variable has been defined, the function pool contains its name, data type, repetition factor, and the address of its value.

A single VDEFINE call may be used to define a list of dialog variables. Each dialog variable in this list may have a separate data type and a separate length, but it is also possible to define all variables with the same data type and the same length.

The storage space for all variables to be defined by calling the VDEFINE service must be contiguous. Individual sections of the storage area can also be defined as fill areas.

The DIM operand can be used to define an array of dialog variables.

Parameters of the interface routine (ISPCI2):

– communication area
– length of the operand area
– operand area
– length area
– value area

Description of parameters:

The parameters must be specified as described in the section “ISPCI interface routine” on page 116.

Length area

Is a storage area of the application program with one or more fields. Each field consists of a full word (4 bytes, word boundary) and must contain a binary signed integer.

If LIST is specified in the OPTION operand, this area must contain a value for each name in the “name-list” operand. The n-th name in the name list corresponds to the n-th value in the length area.

If LIST is not specified in the OPTION operand, only one field is required in this area. The value then applies to all dialog variables in the name list.

The values in the length area determine the lengths of the storage areas in the value area. In other words, the value area is structured by means of the length area.

If an * character is specified as a name in the name list, a fill area of the specified length is defined.

Value area

The value area is a storage area of the application program. It contains the values of the specified dialog variables.

The application program must ensure that the values of the dialog variables are located in this storage area in the specified order and in the specified format. The length of the n-th value in the value area is defined by the content of the n-th field in the length area.

Some programming languages offer “structures” as a language element for this purpose. In COBOL, the COBOL program variables to be declared as explicit dialog variables using a VDEFINE service must be defined in succession. It is important to ensure that the values are stored contiguously in memory.

Structure of the operand area:

```
VDEFINE   name-list

                    ⎧ CHAR   ⎫
                    ⎪ BINSTR ⎪
                    ⎪ NUMS   ⎪
                    ⎪ NUMU   ⎪
          [ FORMAT(⎨ FIXEDS  ⎬,...) ]
                    ⎪ FIXEDU ⎪
                    ⎪ PACK   ⎪
                    ⎩ *      ⎭

          [ DIM ( number ) ]

                    ⎧ COPY   ⎫
          [ OPTION(⎨ NOBSCAN ⎬,...) ]
                    ⎩ LIST   ⎭
```

Operands:

VDEFINE

Name of the service

name-list

The names specified in this list define explicit dialog variables, i.e. associate a variable name with storage space of the program.

No index entries are permitted.

An * in the name list defines a fill area. An * may also be entered at the same position in the list of data types. The length of the fill area is determined by the corresponding value in the length area.

FORMAT(data-type)

Specifies the representation of the value in the value area. “data-type” may be a single data type specification or a list of data types. This list must contain exactly the same number of elements as the names entered in the name list.

If only a single value is specified, all defined dialog variables have the same data type.

An * may be specified if a fill area was defined at that position in the name list.

The permitted data types are dealt with individually below.

DIM(number)

The DIM operand (repetition factor) indicates that the storage layout defined by the variable name, the data types, and the values in the length area occurs more than once. In other words, an array of dialog variables is defined (see also “Types of dialog variables”). If “namelist” consists of multiple names, an array of structures is defined. “number” specifies the number of array elements. If “name-list” consists of only a single name, the array is made up of elementary dialog variables.

Every dialog variable specified in the name list has “number” elements. These elements can be addressed in some dialog services as indexed dialog variables.

The COPY entry in the OPTION operand is ignored if DIM is specified.

The value of “number” can be from 1- 32767.

OPTION(option)

A combination of the following values may be specified as an “option”:

COPY
Obtains values for the explicit dialog variables defined by this VDEFINE service by copying the values of identically-named implicit dialog variables in the function pool or the values of identically-named dialog variables that are present in the profile pool or the values of identically-named system variables of the system pool (see appendix). If required, data conversions are performed when copying values. The function pool is searched first, followed by the system pool and the profile pool in that order. Implicit dialog variables are deleted from the function pool after their values are copied. This prevents naming conflicts. If the DIM operand is specified, COPY is ignored.

NOBSCAN
This value is only significant for the data types CHAR and BINSTR.

If an explicit dialog variable is defined with the attribute NOBSCAN, trailing blanks are treated as significant: when these variables are read, the length defined by VDEFINE is returned as the data length for CHAR, and a length corresponding to the C function strlen is returned for BINSTR. If NOBSCAN is not defined, trailing blanks are treated as insignificant, and the data length without trailing blanks is supplied as the length. The difference can be conceptually understood in terms of fixed or variable-length data.

The following examples illustrate the difference:

```
                                Data in the   Length on reading the
                                 variable     dialog variable

VDEFINE ... CHAR   4 NOBSCAN  Abbb          4
VDEFINE ... CHAR   4          Abbb          1
VDEFINE ... CHAR   4 NOBSCAN  bbbb          4
VDEFINE ... CHAR   4          bbbb          0
VDEFINE ... BINSTR 5 NOBSCAN  Abbb0         4
VDEFINE ... BINSTR 5          Abbb0         1
VDEFINE ... BINSTR 5 NOBSCAN  bbbb0         4
VDEFINE ... BINSTR 5          bbbb0         0
VDEFINE ... BINSTR 5 NOBSCAN  A0000         1
VDEFINE ... BINSTR 5          A0000         1
VDEFINE ... BINSTR 5 NOBSCAN  00000         0
VDEFINE ... BINSTR 5          00000         0
```

This operand is relevant in connection with VCOPY and the display of dialog variables in a mask field and for variable substitution.

LIST

LIST must be specified if the dialog variables of the name list have different lengths. It allows each dialog variable of the name list to be assigned a separate length.

If LIST is not specified, all variables of the name list have the same length. If more than one length is specified in this case, the first length specification is used for all variables of the name list, and the other entries are ignored.

Fill areas can be used in the structure of dialog variables to be defined only if LIST has been specified.

Data types:

The length restrictions given in the data type definitions apply to the dialog manager. Stricter limitations may apply in the application program as a result of the programming language being used.

When an explicit dialog variable is to be displayed in a mask by using the DISPLAY service, a conversion into a string designated as an external format must occur when reading the variable. In the reverse case, when a value entered in the mask is stored in an explicit dialog variable, the string specified as an external format must be converted into the internal format of the dialog variable (see also “Usage of display services”).

Data conversions may also be required when copying or substituting values for dialog variables.

The representations of internal values for different data types are shown below:

CHAR

String; left-justified and padded with blanks.

CHAR variables can have a length of 1 - 32767 bytes.

No validation check for legal characters is performed when accessing the dialog variable.

NUMS

Signed numeric string; only the digits 0 through 9 and the sign (+/-) are permitted. The value is right-justified and padded with zeros (C’0’). The sign is placed immediately to the right of the last digit.

A NUMS variable can have a length of 2-16 bytes.

This variable type supports the previous numeric data representation in FHS addressing aids. The external representation of such variables is only possible in an arithmetic mask field with corresponding editing attributes.

NUMU

Unsigned numeric string; only the digits 0 through 9 are allowed. The value is right-justified and padded with zeros (C'0').

A NUMU variable can have a length of 1-15 bytes.

This variable type supports the previous numeric data representation in FHS addressing aids. The external representation of such variables is only possible in an arithmetic mask field with corresponding editing attributes.

FIXEDS

Signed binary integer that can occupy 2 or 4 bytes.

FIXEDU

Unsigned binary integer that can occupy 2 or 4 bytes.

BINSTR

This variable type differs from type CHAR only with respect to the fill character. The fill character used here is X'00'.

When the value of a BINSTR variable is stored, the value is entered into the space reserved for it up to the maximum length-1; the last character contains the fill character. When reading a BINSTR variable, the first X'00' character marks the end of the value.

This data type supports the type STRING in C.

PACK

Packed decimal number that can occupy 1 - 8 bytes.

The value is positive if the last half-byte contains X'C', and negative if it contains X'D'. For any other contents, the value of the variable is treated as unsigned.

This data type supports the type PACKED DECIMAL in COBOL.

Examples:

1. The following COBOL statements declare an array of dialog variables X, Y, and Z in the structure S:

```
01 LBUF           PIC S9(7)  COMP VALUE 512.
01 BUF            PIC X(512) VALUE SPACE.
01 LENFELD.
   02 VLEN        PIC S9(7)  COMP OCCURS 3 TIMES.
01 S.
   02 SE          OCCURS 3 TIMES.
      03 A        PIC X(10).
      03 B        PIC S9(7)  COMP.
      03 C        PIC X(2).

MOVE 10 TO VLEN(1).
MOVE  4 TO VLEN(2).
MOVE  2 TO VLEN(3).
MOVE 'VDEFINE (X Y Z) FORMAT(CHAR FIXEDS CHAR) DIM(3)
     OPTION(LIST)' TO BUF.
CALL „ISPCI2“ USING DMCOMM LBUF BUF LENFELD S.
```

2. The following COBOL statements declare the dialog variable TEXT, which is addressed in the COBOL program as the variable ERRORS.

```
01 LBUF           PIC S9(7)  COMP VALUE 512.
01 BUF            PIC X(512) VALUE SPACE.
01 ERROR          PIC X(80).
01 VLEN           PIC S9(7)  COMP VALUE 80.

MOVE 'VDEFINE TEXT FORMAT(CHAR)' TO BUF.
CALL „ISPCI2“ USING DMCOMM LBUF BUF VLEN ERRORS.
```

4.5.10 VDELETE - Delete dialog variables in function pool

The VDELETE service can be used by the application program to delete implicit or explicit dialog variables in the function pool.

Parameters of the interface routine (ISPCI):

– communication area
– length of the operand area
– operand area

Description of parameters:

The parameters must be specified as described in the section “ISPCI interface routine” on page 116.

Structure of the operand area:

```
         ⎧ name-list ⎫
VDELETE  ⎨           ⎬
         ⎩ *ALL      ⎭
```

Operands:

VDELETE

Name of the service

name-list

Lists names of dialog variables to be deleted.

If a structure of dialog variables is to be deleted, all names of the dialog variables in the structure must be specified.

If an array of dialog variables is to be removed, all names of the dialog variables of an array element (structure or elementary dialog variables must be specified, but not be indexed).

*ALL

All definitions of dialog variables in the function pool are deleted.

Example:

```
01 LBUF           PIC S9(7)  COMP VALUE 512.
01 BUF            PIC X(512) VALUE SPACE.

MOVE 'VDELETE *ALL' TO BUF.
CALL „ISPCI“ USING DMCOMM LBUF BUF.
```

4.5.11 VERASE - Delete dialog variables in profile pool

The VERASE service can be used by the application program to delete dialog variables from the profile pool.

Parameters of the interface routine (ISPCI):

– communication area
– length of the operand area
– operand area

Description of parameters:

The parameters must be specified as described in the section “ISPCI interface routine” on page 116.

Structure of the operand area:

```
        ⎧ name-list ⎫
VERASE  ⎨           ⎬
        ⎩ *ALL      ⎭
```

Operands:

VERASE

Name of the service

name-list

Specifies the names of dialog variables to be deleted from the profile pool. No index entry is allowed.

*ALL

All dialog variables in the profile pool are deleted.

4.5.12 VGET - Copy variables from profile pool to function pool.

The values of dialog variables from the profile pool or of S variables from an SDF-P variable pool are copied to the function pool.

If an identically-named dialog variable is already present in the function pool, only its value is changed. Data conversion occurs if the types of the variables do not match.

If one of the specified dialog variables does not exist in the function pool, it is created implicitly and assigned the value to be copied.

If a dialog variable or S variable to be copied does not exist, and an identically named dialog variable is present in the function pool, that variable is not changed, i.e. no implicit dialog variable is created.

Parameters of the interface routine (ISPCI):

– communication area
– length of the operand area
– operand area

Description of parameters:

The parameters must be specified as described in the section “ISPCI interface routine” on page 116.

Structure of the operand area:

```
                  ⎧ PROFILE   ⎫
VGET name-list[   ⎨ PROCEDURE ⎬ ]
                  ⎩ TASK      ⎭
```

Operands:

VGET

Name of the service

name-list

Lists the names of dialog variables (or S variables) whose values are to be copied to the function pool. No index entry is allowed.

PROFILE

The dialog variables specified in the name list are to be read from the profile pool. This entry is the default value and may be omitted.

PROCEDURE

The S variables specified in the name list are to be copied from the current SDF-P procedure pool.

TASK

The S variables specified in the name list are to be copied from the task-specific SDF-P pool.

4.5.13 VPUT - Copy variables from function pool to profile pool or to the SDF-P variable pool

The current values of dialog variables of the function pool are copied to the profile pool or to the SDF-P variable pool.

If an identically-named dialog variable already exists in the target pool, only its value is changed. A data conversion is performed if the data types of the variables do not match. If the target is the profile pool, an implicit dialog variable is created in the profile pool if no such variable exists in it. This implicit dialog variable is assigned the data type FIXEDS if the variable to be copied is of type FIXEDS or FIXEDU. Otherwise, type CHAR is assigned.

If a specified dialog variable does not exist in the function pool or in an SDF-P variable pool, a warning is issued by means of an appropriate return code.

Parameters of the interface routine (ISPCI):

– communication area
– length of the operand area
– operand area

Description of parameters:

The parameters must be specified as described in the section “ISPCI interface routine” on page 116.

Structure of the operand area:

```
                  ⎧ PROFILE   ⎫
VPUT name-list[   ⎨ PROCEDURE ⎬ ]
                  ⎩ TASK      ⎭
```

Operands:

VPUT

Name of the service

name-list

Lists the names of dialog variables whose values are to be copied. No index entry is allowed.

PROFILE

The dialog variables specified in the name list are to be copied to the profile pool. This entry is the default value and may be omitted.

PROCEDURE

The dialog variables specified in the name list are to be copied to the current SDF-P procedure pool.

TASK

The dialog variables specified in the name list are to be written to the task-specific SDF-P pool.

The following length restrictions apply:

– Profile pool: the maximum length for the data type CHAR is 16383;
– SDF-P variable pool: the maximum length is 4096.

4.5.14 VREPLACE - Replace values of dialog variables in function pool

The VREPLACE service is used to modify the values of implicit or explicit dialog variables in the function pool.

The dialog variable to be modified may be a variable that was defined earlier as an explicit or implicit dialog variable. If one of the specified dialog variables does not exist, it is created as an implicit dialog variable.

The implicit dialog variable is assigned the data type FIXEDS if the type of the new value (source value) is FIXEDS or FIXEDU. Otherwise, the implicit dialog variable is assigned type CHAR.

If one of the dialog variables to be modified has a type that defers from that of the new value, a data conversion is performed in order to store the new value. The existing dialog variable retains its data type.

An existing implicit dialog variable is removed if the new value has a length of 0.

Parameters of the interface routine (ISPCI):

– communication area
– length of the operand area
– operand area
– length area
– value area

Description of parameters:

The parameters must be specified as described in the section “ISPCI interface routine” on page 116.

Length area

Is a storage area of the application program with one or more fields. Each field consists of a full word (4 bytes, word boundary) and must contain a binary signed integer.

This area must contain a value for each name of a dialog variable specified in the “name-list” operand. The n-th name in the name list corresponds to the n-th value in the length area.

These values specify the length of values in the value area.

Value area

The value area is a storage area in the application program that contains the new values (source values) for the dialog variables. There must be one value present for each name in the name list.

The application program must ensure that the values of the dialog variables are stored contiguously in the order defined by the name list and in the specified format in disk storage area.

The length of the n-th value is defined by the contents of the n-th field of the length area.

Structure of the operand area:

```
VREPLACE  name-list

                    ⎧ CHAR   ⎫
                    ⎪ BINSTR ⎪
                    ⎪ NUMS   ⎪
                    ⎪ NUMU   ⎪
          [ FORMAT(⎨ FIXEDS  ⎬,...) ]
                    ⎪ FIXEDU ⎪
                    ⎪ PACK   ⎪
                    ⎩ *      ⎭
```

Operands:

VREPLACE

Name of the service

name-list

Lists the names of one or more dialog variables for which values in the function pool are to be modified.

If a name is followed by a “#” and an index, the value of the dialog variable of the corresponding element in an array of dialog variables is modified.

A * in the name list defines a fill area. An * character may also be entered at the same position in the list of data types. The length of the fill area is defined by the corresponding value in the length area.

FORMAT(data type)

Specifies the data type for the copied value in the target area (see also VDEFINE). If the data types of the source and target do not match, a type conversion, if possible, is performed.

“data-type” may be given as a single data type specification or as a list of data types. This list must contain exactly the same number of elements as the names entered in the name list.

If only a single value is specified, that data type applies to all dialog variables.

If the FORMAT operand is permitted, data type CHAR is assumed for all values.

An * can be specified if a fill area was defined at that position in the name list.

The permissible data types for data conversions are shown in the table below:

Source value type	**Type of dialog variable (target) to be substituted:**							**Implicit dialog variable**
	CHAR	**BINSTR**	**NUMS**	**NUMU**	**FIXEDS**	**FIXEDU**	**PACK**	
CHAR	+	+	1)	2)	1)	2)	1)	CHAR
BINSTR	+	+	1)	2)	1)	2)	1)	CHAR
NUMS	3)	3)	+	4)	+	4)	+	CHAR
NUMU	3)	3)	+	+	+	+	+	CHAR
FIXEDS	3)	3)	+	4)	+	4)	+	FIXEDS
FIXEDU	3)	3)	+	+	+	+	+	FIXEDS
PACK	3)	3)	+	4)	+	4)	+	CHAR

The following applies:

\+ Data conversion possible

\- Data conversion not possible

1. The value of the source field must have the following format:

   [+/-]9[9...][b...] left-justified

2. The value of the source field must have the following format:

   [+]9[9...][b...] left-justified

3. The value of the target field has the following format:

   [-]9[9...][b...] left-justified

4. Sign errors possible

   Meanings:

   9 Digits 0 - 9

   b Blanks (or NUL for BINSTR)

Examples:

```
01 LBUF        PIC S9(7)  COMP VALUE 512.
01 BUF         PIC X(512) VALUE SPACE.
01 VALUES      PIC X(80).
01 VLEN.
03 LV          PIC S9(7)  COMP OCCURS 4 TIMES.
MOVE '1234567890ABCDEFGHIJKL' TO VALUES.
MOVE 10 TO LV(1).
MOVE 10 TO LV(2).
MOVE 'VREPLACE (V1 V2)' TO BUF.
CALL „ISPCI2“ USING DMCOM LBUF BUF VLEN VALUES.
MOVE  5 TO LV(1).
MOVE 10 TO LV(2).
MOVE 'VREPLACE (V3 V4) FORMAT(CHAR)' TO BUF.
CALL „ISPCI2“ USING DMCOMM LBUF BUF VLEN VALUES.
```

After the VREPLACE service is executed, the dialog variables have the following values:

Values:

```
V1    1234567890
V2    ABCDEFGHIJ
V3    12345
V4    67890ABCDE
```

4.6 Assigning libraries

The following libraries are required when working with the dialog manager and must be assigned at the first call to a dialog service. The assignment remains in effect till the end of the session.

– Format library

  The format library contains all the required dialog elements and must be assigned as a primary library with the file link name IDHPLIB. Alternative libraries may be assigned via a file link name BLSLIBnn, where nn is a number from 00 to 99. The libraries are searched by number in ascending order. It is not necessary to have a continuous sequence.

– Profile library

  The profile library contains the profile variables and is user-specific. Each terminal user must be assigned his or her profile library with the file link name IDHPROF. The language-specific parts for the use of different international languages (formats, key assignments, help and messages) are located in a format library. The specific language variant required for each terminal user can then be activated by assigning the respective language-specific format library when calling FHS-DM.

4.7 Compatibility

The dialog manager of FHS can only be used for TIAM applications. It is not compatible with application programs that use FHS as before. Such applications are, however, still executable without restrictions.

The dialog manager can only be used with formats that were generated with IFG as of Version V08.1A for explicit use with the dialog manager. These formats are called DM formats.

DE formats that were generated for the dialog extension FHS Version V8.0 for use under UTM can only be processed to a restricted extent by the dialog manager. This is generally not very meaningful, since DE formats are intended for UTM applications.

DM formats are extended DE formats with the following differences:

– The assignment for field names for all I/O fields and for markable text fields is checked by IFG. This is mandatory due to the connection with dialog variables.

– DM formats have additional names for dialog variables, selection fields, list areas, and pull-down boxes. Without these specifications, the dialog manager can only process such elements to a limited extent.

– Guidance text is provided for scrolling information in help panels.

There are no differences with respect to messages and key assignment tables (key lists).

Since addressing aids have been dropped, it is no longer possible to work with global and field-related attributes as before. Other features are offered instead by the dialog manager.

Not all of the functions that are definable with IFG are processed by the dialog manager. The following options are ignored:

– Specifications for input fill characters
– Specification of undefined values
– Provision of an exit routine

4.8 SDF-P variables in FHS-TIAM programs

Access to SDF-P variables

The variable services VGET and VPUT enable an application program to read and write elementary SDF-P variables. This permits communication between an SDF-P procedure and an application program called from within that procedure. The use of SDF-P variables is described in detail in the next chapter.

5 SDF-P interface

As of FHS V8.1, FHS can also assume the role of the output server for S variable streams. In addition, applications for FHS V8.1 can be used and controlled with the aid of S procedures.

Both these functions are described below in the following two sections.

5.1 FHS as an output server

FHS is available as an output server on both the program and the command level in the SDF-P context, i.e.:

– FHS-PRIV V8.1 can be called on the command level using TRANSMIT-BY-STREAM.

– FHS-PRIV V8.1 can be called by a TU program using a TRANSVV SVC.

As a prerequisite to both options, the variable stream used by the program must be assigned to FHS in the SERVER operand of the ASSIGN-STREAM command.

When information for FHS V8.1 is received via a TRANSMIT-BY-STREAM command FHS provides display services that correspond to the DISPLAY, ADDPOP and REMPOP services which it offers in TU mode.

5.1.1 ASSIGN-STREAM

In order to assign FHS as a Server for S variable streams, FHS must be specified in the SERVER(...) assignment of the TO operand in the ASSIGN-STREAM command. You can assign your own S variable streams for FHS applications in the STREAM-NAME operand. More information on this subject can be found in the manual “SDF-P V2.0“.

Example:

```
ASSIGN-STREAM STREAM-NAME=<strutured-name 1...20>
              TO=*SERVER(SERVER-NAME=FHS,
                  SERVER-INFORMATION='FHS-LIB=format library
                                     [,P-KEY-SETTING=(P1,....,P20)]')
```

When this command is entered in SDF-P, FHS is called. The specification in SERVER-INFORMATION is then evaluated by FHS:

– FHS-LIB is the format library of FHS. The name of the FHS library must be specified as follows:

  ```
  SERVER-INFORMATION = 'FHS-LIB=$USER-ID.library-name'.
  ```

  You must specify the name of a format library here. If desired, you can also specify some other format library by using a LINK command (e.g. to link the German standard formats IDHx.... from $.SYSFHS.FHS.081.FHS-DM.D).

  The formats are searched in the library specified under SERVER-INFORMATION and then loaded into the BLSLIBxx (00 -99) being used.

– P-KEY-SETTING specifies which F key is simulated by a P key. This entry is optional; it is only shown here to illustrate the complete scope of the ASSIGN-STREAM command.

5.1.2 TRANSMIT-BY-STREAM

The transfer of S variables is controlled in S procedures with the TRANSMIT-BY-STREAM command. This command controls the transfer of variables to and from the specified server via a selected S variable stream from the client side.

The selected variable stream is specified in the STREAM-NAME operand.

The remaining operands define which variables are to be transferred to or from the server in accordance with the setting for ASSIGN-STREAM ...,TO=*VARIABLE( ). In other words, these variables specified here must be declared as structures as in the case of ASSIGN-STREAM.

Example

```
TRANSMIT-BY-STREAM  STREAM-NAME = <strutured-name 1...20>
                   ,VARIABLE-NAME = <composed-name 1..255>
                   ,RETURN-VARIABLE-NAME = *SAME / <composed-name 1..255>
                   ,CONTROL-VAR-NAME = <composed-name 1..255>
                   ,RET-CONTROL-VAR-NAME = *SAME / <composed-name 1..255>
```

If this command is entered in SDF-P and FHS is assigned as the server, FHS is called and the contents of the S variable streams are output to FHS masks.

FHS can itself generate variables that are returned to the caller. The source of such variables may be input fields (values entered by the terminal user) or FHS system parameters (e.g. cursor position,...).

The variables can be classified by content into two types:

– Data
– Control variables

The control variables for FHS are defined in the S variable SYSFHS-CONTROL (see also page 193).

The layout is supplied as a procedure in the library SYSPRC.FHS.081 in the member DISPLAY-CONTROL. The members have reserved names with fixed semantics. Variables declared with this procedure must be defined with SCOPE=VISIBLE.

SYSFHS-CONTROL includes the following 7 information categories:

– layout and version
– information on display services
– specifications for the output location
– ispecifications for the format
– information on field attributes
– information on input
– FHS return codes

After the call, FHS resets the elements of the control variables to default values, if such values have been defined.

5.2 FHS services for SDF-P applications

In order to call a variable service from an SDF-P procedure, you must insert the appropriate value in the variable structure defined in the TRANSMIT-BY-STREAM command. The possible values are listed in the following section.

For the sake of clarity, the single quotes for string values have been omitted in the following tables. Please note that values assigned to string variables must be enclosed within single quotes.

Example:

```
/MYFHS-CONTROL.SERVICE = '*ADD-POPUP'
/MYFHS-CONTROL.POP-LOCATION = 'MYFIELD'
```

When working with lists, an index may be entered at positions where a field name can be specified. The index must be replaced by an “*” (asterisk), and then specified as a value.

5.2.1 Layout and version

These variables are supplied in the standard header of SDF-P. The values of FHS are listed below. Layout version 1 is implemented in FHS V8.1.

Name	Type	Possible value
UNIT	string	'FHS'
FUNCTION	string	'DISPLAY'
VERSION	integer	1

5.2.2 Information on display services

In order to call a variable service (DISPLAY-, ADDPOP, and REMPOP) from an SDF-P procedure, you must enter the desired service in the service specification of the TRANSMIT-BY-STREAM command. The following entries are possible.

Name	Type	Default value	Possible value
SERVICE	string	*DISPLAY	*ADD-POPUP / *REMOVE-POPUP / *ADDPOP-DISPLAY / *REMPOP-DISPLAY
DIAGINFO	string	*NO	*YES / *WARNING
POP-LOCATION	string	*NONE	<field-name 1..255> / *CENTRAL
POP-LOC-IND	integer	0	<integer>
ROW	integer	2	<integer>
COLUMN	integer	2	<integer>

SERVICE:

The service to be executed by FHS:

*DISPLAY Outputs a format of the current box hierarchy level (as for TU-FHS, see page 137).

*ADD-POPUP Adds a pop-up hierarchy (as for TU-FHS, se page 126).

*REMOVE-POPUP
Removes/deletes a pop-up hierarchy (as for TU-FHS, see page 146).

*ADDPOP-DISPLAY
Combined action; adds a pop-up hierarchy and outputs a format of that hierarchy level.

*REMPOP-DISPLAY
Combined action; deletes a pop-up hierarchy and outputs a format of a lower hierarchy level.

DIAGINFO

Displays additional diagnostic information (e.g. DMERR for TU-FHS).

*NO No diagnostic information

*YES Diagnostic information for errors

*WARNING Diagnostic information for errors and warnings

POP-LOCATION:
defines the reference point of a box.

*NONE The reference point is the upper left corner of the current format.

<field name 1..255>
Name of the field whose first letter represents the starting point for the box.

*CENTRAL The center of the screen.

POP-LOC-IND:
Index entry for POP-LOCATION

If the field name refers to a field of a list area, you can enter an index to define a specific list line.

Example:

The following must be specified for POP-LOCATION:

POP-LOCATION = 'A#*'
POP-LOC-IND = 4

ROW:
Vertical offset of the box to the reference point:
positive value: under; negative value: above.

COLUMN:
Horizontal offset of the box to the reference point:
positive value: to the right; negative value: to the left.

5.2.3 Specifications for the output location

These variables define the IFG format, the message (if present), and the output location.

Name	Type	Default value	Possible values
RESOURCE	string	*SAME	<panel-name> / *ALL
MESSAGE-ID	string	*NONE	<msg-id>
MESSAGE-FIELD	string	*NONE	<field-name 1..255> / $lll#ccc / *CENTRAL
MSG-FIELD-IND	integer	0	<integer>

RESOURCE:
Name of the format.

*SAME — Redisplays the last format that was output

<panel-name> — Name of the format

*ALL — Is used by the REMPOP service to remove all formats.

MESSAGE-ID:
Identifies the message to be output/displayed in the format

MESSAGE-FIELD:
Name of the field for which the message is to be output/displayed.

*NONE — The reference point is the top left corner of the current format.

<field-name 1..255> — Name of the field whose first letter represents the starting point for the box.

$lll#ccc — Coordinates of the field (lll = line, ccc = column)

*CENTRAL — Central point of the screen.

MSG-FIELD-IND
Index entry for MESSAGE-FIELD.Specifications for the format

These variables define the output attributes of a format.

Name	Type	Default value	Possible values
CURSOR-OUTPUT-INDEX	integer	0	<integer>
CURSOR-OUTPUT	string	*NONE	<field-name 1..255> / $lll#ccc
CURSOR-OUTPUT-POS	integer	0	<integer>
LOCK	string	*NO	*YES
ALARM	string	*NO	*YES

Name	Type	Default value	Possible values
HARDCOPY	string	*NO	*YES
AUTOTAB	string	*YES	*NO
MANDATORY	string	*NO	*YES
REFRESH	string	*NO	*YES
ACK	string	*NO	*YES

CURSOR-OUTPUT-INDEX:
Index entry for CURSOR-OUTPUT.

If the field name references a field of a list area, you can include an index to specify a particular list line.

CURSOR-OUTPUT:
Name of the field in which the cursor is to be placed on output:

*NONE The cursor is placed in the first field.

<field-name 1..255>
Name of the field.

$lll#ccc Coordinates of the field (lll = line, ccc = column).

CURSOR-OUTPUT-POS:
Position of the cursor within the field defined by the CURSOR operand. The first character in the field is at position 1.

The CSRPOS operand is only significant if the value in the CURSOR operand is the name of an input field.

If CSRPOS is omitted or has an illegal value (less than 1 or greater than the field length), a value of 1 is assumed.

LOCK:

*YES: If this operand is specified, control is returned to the application program immediately after the format is displayed. This function can be typically used to display a logo or a “Please wait” message.

ALARM:

*YES: If this operand is specified, an acoustic signal (beep) is used at the terminal to indicate that the format is displayed.

HARDCOPY:

*YES: If this value is specified, the contents of the screen are automatically output to an available hardcopy device.

AUTOTAB:

*YES: The cursor skips automatically from field to field.

*NO: The cursor can be positioned on protected fields of the format by using the arrow keys.

MANDATORY:

*YES: Fields defined with the attribute “Mandatory input” retain that attribute when the same format is output again.

REFRESH:

*YES: The screen is refreshed (no differential output)

ACK:

Before the format is output, “ACK” is displayed as an input prompt in line mode, and the system waits for an acknowledgment from the user.

5.2.4 Information on field attributes (TU ATTR)

These variables enable dynamic modification of some field attributes.

Name		Type	Default value	Possible values
ATTR		list	<list of structure>	
	FIELD	string	*CURSOR	<field-name 1..255> / (field-name1,....,field-name8)
	FIELD-IND		0	<integer> / (<integer1>,....,<integer8>)
	TYPE	string	*UNCHANGED	*INPUT / *OUTPUT / *MANDATORY
	HILITE	string	*UNCHANGED	*UNDERLINE / *INVERSE / *INV-UND
	INTENSITY	string	*UNCHANGED	*HIGH / *LOW / *INVISIBLE / *HIGH-INV / *LOW-INV
	OUTPUT	string	*UNCHANGED	*INIT / *CHECK
	COLOR	string	*UNCHANGED	*RED / *GREEN / *YELLOW / *BLUE / *MAGENTA / *CYAN / *WHITE

ATTR: List of fields to be modified during a DISPLAY service. The attributes are contained in the structure that follows.

FIELD:

Name of the field to be modified.

*CURSOR The field for which attributes are to be modified is the field in which the cursor will be displayed at the next DISPLAY call. If it is determined at the next DISPLAY call that a dynamic attribute has also been requested for the field name of the field on which the cursor is positioned, the specifications are combined. If the entries are the same, the *CURSOR entry is given precedence.

<field-name 1..255> / (field-name1,....,field-name8)
Specifies the name of the field to be modified.

FIELD-IND:

Index entry for the field specified in FIELD.

If the field name references a field of a list area, you can include an index to specify a particular list line.

TYPE:

Specifies the type of attribute to be set for the field.

*INPUT
The corresponding mask field becomes an unprotected input field without the “mandatory” attribute. The field can be modified via the keyboard.

*MANDATORY
Input for the mask field is mandatory, i.e. the field must be modified by the user at the terminal.

*OUTPUT
The mask field becomes a unmarkable output field. Its contents cannot be modified via the keyboard.

HILITE:

Defines how mask fields are to be highlighted.

*UNDERLINED
The field contents are shown underlined.

*INVERSE
The field contents are displayed in reverse video.

*INV-UND
The field contents are displayed underlined and in reverse video.

Note:
If a method of highlighting fields was defined in the format definition itself, dynamic attributes cannot be used to cancel the existing definition. Fields to be dynamically highlighted should therefore be excluded from highlighting specifications in the format definition.

INTENSITY:

Specifies the intensity level (brightness) of the field.

*HIGH
The field is shown with high intensity.

*LOW
The field is shown with normal intensity.

*INVISIBLE
The field contents are invisible and cannot be printed.

*HIGH-INV
The background is displayed with high intensity.

*LOW-INV
The background is displayed with normal intensity.

OUTPUT:

Defines the use of default values for mask fields.

Without the OUTPUT specification, default values for the mask are always used if the dialog variable assigned to a mask field does not exist or has a relevant length of 0 on executing the DISPLAY call. On return from the DISPLAY service, the corresponding dialog variables contain the default value for each respective mask field. If a dialog variable does not exist, a variable of type CHAR is created implicitly.
Specifications for defaults are ignored for list fields.

*INIT
The dialog variables are initialized, i.e. the default value that was set for the specified field on defining the format is output when displaying the mask regardless of the current contents of the dialog variable associated with that mask. If there is no default value in the format for the variable, the field is displayed with fill characters.

*CHECK
For an existing dialog variable that has a value with a relevant length greater than zero, the default value is only used if the content of the dialog variable associated with the mask field has the value zero.

Without the OUTPUT operand, defaults for mask fields are always used if the dialog variables associated with the masked fields do not exist or have a relevant length of zero.

COLOR:

Specifies the color for the mask field. This entry is ignored for devices on which no color can be represented.

*RED

*GREEN

*YELLOW

*BLUE

*MAGENTA

*CYAN

*WHITE

5.2.5 Information on input

These variables define the information returned by FHS after formatting. There are no default values here, since these values are returned by FHS.

Name	Type	Return value
COMMAND	string	<string 1..255>
FHS-VERSION	string	<string 1..30>
CURSOR-INPUT	string	<string 1..255>
CURSOR-INPUT-INDEX	integer	<integer>
CURSOR-INPUT-POS	integer	<integer>

COMMAND:
Scrolling or application command to which the application must respond.

FHS-VERSION:
Version of the dialog manager.

CURSOR-INPUT:
Name of the field in which the cursor is located. If the cursor is in an unknown field or between fields, “$LLL#CCC” is returned, where l stands for line and c for column.

CURSOR-INPUT-INDEX:
Index of the cursor in a list.

CURSOR-INPUT-POS:
Position of the cursor in the field defined by CURSOR-POSITION.

5.2.6 FHS return codes

These variables are output in a standard header of SDF-P. FHS uses them to return output information.

Name	Type	Return code
SUBCODE2	integer	<integer>
SUBCODE1	integer	<integer>
MAINCODE	string	<name 1..7>

5.3 Naming conventions for list processing

Conventions SDF-P variables in the model line (list processing)

Name with no index

Example: ABC

ABC: is the current list

SDF-P variable definition:

```
/ declare-variable name = ABC (type=*string),-
/ multiple-elements = *list(limit=100)
```

Structured name with no index

Example: A.B-C

B-C: is the current list

SDF-P variable definition:

```
/ declare-variable name = A (type = *structure(*by-syscmd))
/ begin-structure
/ declare-element name = B-C (type=*string),-
/ multiple-elements = *list(limit=100)
/ end-structure
```

Structured name with constant index

Example: A#3.B-C

A: list or array of structures which themselves contain a list (index=3)

B-C: current list to be displayed

SDF-P variable definition:

```
/ declare-variable name = A (type = *structure(*by-syscmd)),-
/ multiple-elements = *array(upper-bound = 10)
/ begin-structure
/ declare-element name = B-C (type=*string),-
/ multiple-elements = *list(limit=100)
/ end-structure
```

Structured name with variable index

Example: A#*.B-C

A: is the current list

B-C: is scalar

SDF-P variable definition:

```
/ declare-variable name = A (type = *structure(*by-syscmd)),-
/ multiple-elements = *list (limit = *NONE)
/ begin-structure
/ declare-element name = B-C (type=*string)
/ end-structure
```

Structured name with variable and constant index

Example: A#*.B-C#3

A: current list; list of structures from which one element is to be displayed

B-C: list (constant index)

SDF-P variable definition:

```
/ declare-variable name = A (type = *structure(*by-syscmd)),-
/ multiple-elements = *list (limit = *NONE)
/ begin-structure
/ declare-element name = B-C (type=*string),-
/ multiple-elements = *list (limit=100)
/ end-structure
```

5.4 Controlling FHS applications using S procedures

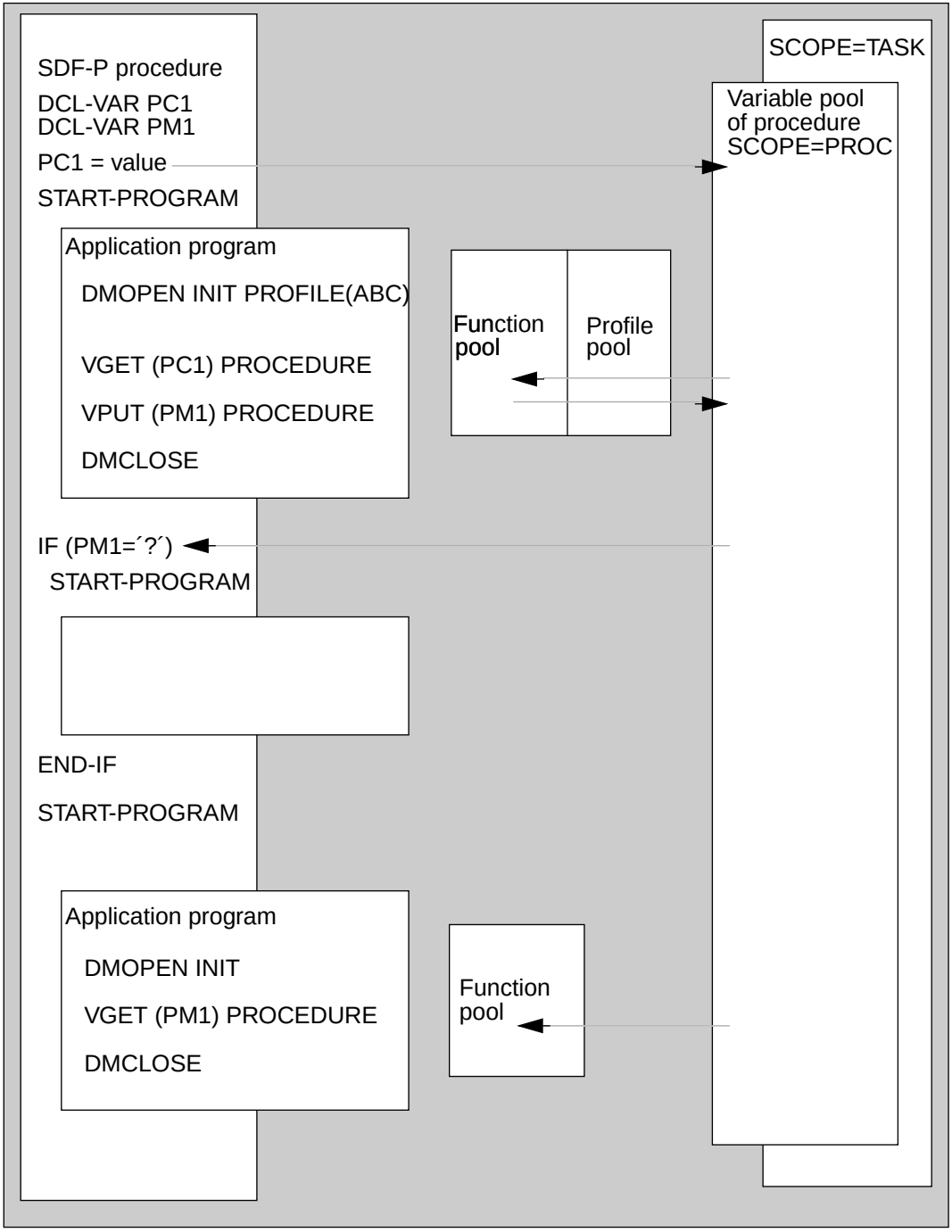
5.4.1 Outputting S variables with FHS 8.1

The commands ASSIGN-STREAM and TRANSMIT-BY-STREAM can be used by S procedures to start a dialog with the terminal user by means of FHS format fields that are filled with FHS dialog variables.

5.4.2 Outputting and generating S variables in FHS-TIAM programs

The variable services VGET and VPUT in FHS V8.1 enable an application program to read and write elementary S variables. This makes communication possible between an S procedure and an application program that is called by it.

The following figure shows the exchange between an application program and an S procedure.


SDF-P procedure
DCL-VAR PC1
DCL-VAR PM1
PC1 = value
START-PROGRAM
SCOPE=TASK
Variable pool
of procedure
SCOPE=PROC
Application program
DMOPEN INIT PROFILE(ABC)
VGET (PC1) PROCEDURE
VPUT (PM1) PROCEDURE
DMCLOSE
Function
pool
Profile
pool
IF (PM1=´?´)
START-PROGRAM
END-IF
START-PROGRAM
Application program
DMOPEN INIT
VGET (PM1) PROCEDURE
DMCLOSE
Function
pool

5.4.3 Controlling FHS applications in nested S procedures

Assignments of S variable streams are stacked in nested S procedures exactly as for system files (SYSDTA,SYSOUT,..). You should therefore use the operand setting SYSTEM-FILE-CONTEXT=STD or OWN in the SET-PROCEDURE-OPTIONS command if you want your string statements to also be processed in a stack.

In TPR mode, FHS saves its display environment in accordance with the S variable stream assignment in the TRANSMIT-BY-STREAM command. A context specific to the variable stream is initialized by FHS at the time of assignment and automatically used thereafter for every FHS operation with the same variable stream. This occurs until the assignment is terminated (e.g. if *DUMMY or another server is assigned to the same variable stream) explicitly or implicitly at the end of the procedure (as defined by the SYSTEM-FILE-CONTEXT assignment).

This mechanism allows nested procedures to be coded independently without overlapping display instructions in different FHS contexts.

To ensure the independence of the calling procedure, the variable SYSFHS-CONTROL.REFRESH must be set to the value *YES* to enable the output of a control panel after the call to this procedure.

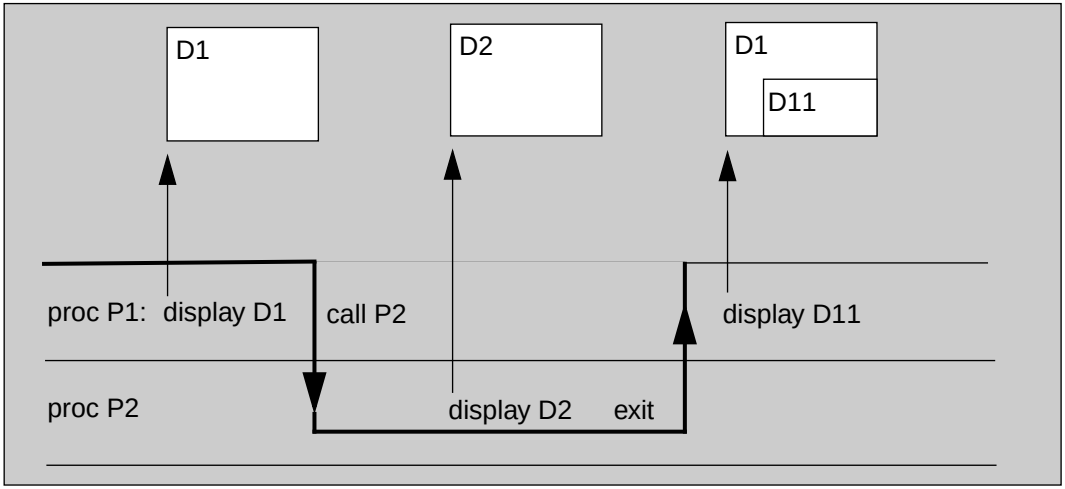
Example

In procedure P1, FHS is assigned to variable stream S1, and control panel D1 is displayed. P1 calls procedure P2. In procedure P2, FHS is likewise assigned to the variable stream S2, and control panel D2 is displayed. Consequently, D1 is completely overwritten by D2.

```
Proc P1 : assign   S1 -----> FHS
          displays D1
          calls    P2
          pop-up   D11


Proc P2 : assign   S2 -----> FHS
          displays D2
          exit
```

On termination of P2, FHS returns to the display environment of S1. In order to do this, the pop-up menu on the current screen (e.g. control panel D11) is implicitly refreshed (with or without a branch) on the current control panel of P1 (D1).



Control panel D2 is deleted, and control panel D1 is restored at the next display of procedure P1.

Note:

Variables to be displayed in a control panel must be visible in the current S procedure (see the section on “Scope of variables” in the chapter “Using variables in S procedures”).

5.5 SYSFHS-CONTROL - Structure for layout and initialization

The following S procedure is supplied with FHS. It defines the layout of the control variables and initializes them.

```
/set-procedure-options caller=include
/begin-parameter-declaration
/   declare-parameter -
/   „------------ std param ---------------------------------*“-
/               (PREFIX       (type=string,init=’SYSFHS-’) -
/               ,INCLUDE-FORM (type=string,init=’LAYOUT’) „/initialize“ -
/               ,VARIABLE-NAME(type=string,init=’’) -
/   „------------ include specific param -------------------*“-
/  „ action variables „ -
/               ,SERVICE      (type=string,init=’*DISPLAY’) -
/               ,DIAGINFO     (type=string,init=’*NO’) -
/               ,POP-LOCATION (type=string,init=’*NONE’) -
/               ,POP-LOC-IND  (type=integer,init=0) -
/               ,ROW          (type=integer,init=2) -
/               ,COLUMN       (type=integer,init=2) -
/  „ resource variables „ -
/               ,RESOURCE      (type=string,init=’*SAME’) -
/               ,MESSAGE-ID    (type=string,init=’*NONE’) -
/               ,MESSAGE-FIELD (type=string,init=’*NONE’) -
/               ,MSG-FIELD-IND (type=integer,init=0) -
/  „ panel variables „ -
/               ,CURSOR-OUTPUT-INDEX (type=integer,init=0) -
/               ,CURSOR-OUTPUT       (type=string,init=’*NONE’)  -
/               ,CURSOR-OUTPUT-POS   (type=integer,init=0) -
/               ,LOCK                (type=string,init=’*NO’) -
/               ,ALARM               (type=string,init=’*NO’) -
/               ,HARDCOPY            (type=string,init=’*NO’) -
/               ,AUTOTAB             (type=string,init=’*YES’)  -
/               ,MANDATORY           (type=string,init=’*NO’)  -
/               ,REFRESH             (type=string,init=’*NO’)  -
/               ,ACK                 (type=string,init=’*NO’)  -
/  „ field attributes „ -
/               ,ATTR-LIST    (type=integer,init=0) -
/                „ number of list elements to reset       „-
/               ,FIELD     (type=string,init=’*CURSOR’) -
/               ,FIELD-IND (type=string,init=’0’) -
/               ,TYPE      (type=string,init=’*UNCHANGED’) -
/               ,HILITE    (type=string,init=’*UNCHANGED’) -
/               ,INTENSITY (type=string,init=’*UNCHANGED’) -
/               ,COLOR     (type=string,init=’*UNCHANGED’) -
/               ,OUTPUT    (type=string,init=’*UNCHANGED’) -
```

```
/   „ input information „ -
/                ,COMMAND               (type=string,init='') -
/                ,FHS-VERSION           (type=string,init='') -
/                ,CURSOR-INPUT          (type=string,init='') -
/                ,CURSOR-INPUT-INDEX    (type=integer,init=0) -
/                ,CURSOR-INPUT-POS      (type=integer,init=0) -
/                )
/end-parameter-declaration
/
/if (upper-case(INCLUDE-FORM) == 'LAYOUT')
/
/begin-structure ATTR,scope=proc
/   declare-element -
/                    (FIELD     (type=string) -
/                    ,FIELD-IND (type=string) -
/                    ,TYPE      (type=string) -
/                    ,HILITE    (type=string) -
/                    ,INTENSITY (type=string) -
/                    ,COLOR     (type=string) -
/                    ,OUTPUT    (type=string) -
/                    )
/end-structure//begin-structure name=&PREFIX.LAYOUT,scope=proc
/  declare-element STD-HEADER(type=structure(&PREFIX.FHDR))
/  declare-element -
/  „ action variables „ -
/                 (SERVICE      (type=string) -
/                 ,DIAGINFO     (type=string) -
/                 ,POP-LOCATION (type=string) -
/                 ,POP-LOC-IND  (type=integer) -
/                 ,ROW          (type=integer) -
/                 ,COLUMN       (type=integer) -
/  „ resource variables „ -
/                 ,RESOURCE         (type=string) -
/                 ,MESSAGE-ID       (type=string) -
/                 ,MESSAGE-FIELD    (type=string) -
/                 ,MSG-FIELD-IND    (type=integer) -
/  „ panel variables „ -
/                 ,CURSOR-OUTPUT-INDEX (type=integer) -
/                 ,CURSOR-OUTPUT       (type=string)  -
/                 ,CURSOR-OUTPUT-POS   (type=integer) -
/                 ,LOCK                (type=string)  -
/                 ,ALARM               (type=string)  -
/                 ,HARDCOPY            (type=string)  -
/                 ,AUTOTAB             (type=string)  -
/                 ,MANDATORY           (type=string)  -
/                 ,REFRESH             (type=string)  -
/                 ,ACK                 (type=string)  -
/                 )
```

```
/  „ field attributes „
/  declare-element ATTR            (type=struc(attr))-
/                 ,mult-elem=list
/
/   „ input information „ -
/   declare-element -
/                  (COMMAND               (type=string) -
/                  ,FHS-VERSION           (type=string) -
/                  ,CURSOR-INPUT          (type=string) -
/                  ,CURSOR-INPUT-INDEX    (type=integer) -
/                  ,CURSOR-INPUT-POS      (type=integer) -
/                  )
/end-structure
/
/else-if (upper-case(INCLUDE-FORM) == 'INITIALIZE')
/
/   if (VARIABLE-NAME == '')
/      write-text '%  mandatory parameter variable-name missing.'
/      raise-error
/   end-if
/  declare-variable PARAM(type=string)
/  include-procedure *lib-elem(lib=$.SYSPRC.SDF-P-BASYS.020,el=FHDR) -
/   „------------ std param ----------------------------------*“-
/               ,(INCLUDE-FORM='INITIALIZE' -
/                ,VARIABLE-NAME='&VARIABLE-NAME..STD-HEADER' -
/   „------------ include specific param --------------------*“-
/                ,UNIT ='FHS',  „fhs unit name“ -
/                ,FUNCTION ='DISPLAY', „fhs fc for display?“ -
/                ,VERSION = 1 „control variable layout version“-
/                )
/
/   for PARAM = -
/                  ('SERVICE'             -
/                  ,'DIAGINFO'            -
/                  ,'POP-LOCATION'        -
/                  ,'POP-LOC-IND'         -
/                  ,'ROW'                 -
/                  ,'COLUMN'              -
/                  ,'RESOURCE'            -
/                  ,'MESSAGE-ID'          -
/                  ,'MESSAGE-FIELD'       -
/                  ,'MSG-FIELD-IND'       -
/                  ,'CURSOR-OUTPUT-INDEX' -
/                  ,'CURSOR-OUTPUT'       -
/                  ,'CURSOR-OUTPUT-POS'   -
/                  ,'LOCK'                -
/                  ,'ALARM'               -
/                  ,'HARDCOPY'            -
```

```
/                  ,'AUTOTAB'             -
/                  ,'MANDATORY'           -
/                  ,'REFRESH'             -
/                  ,'ACK'                 -
/                  ,'COMMAND'             -
/                  ,'FHS-VERSION'         -
/                  ,'CURSOR-INPUT'        -
/                  ,'CURSOR-INPUT-INDEX'  -
/                  ,'CURSOR-INPUT-POS'    -
/                  )
/         &VARIABLE-NAME..&PARAM = &PARAM
/   end-for
/
/   if (ATTR-LIST > 0)
/      I = 1
/      while ( I <= ATTR-LIST )
/         for PARAM = -
/                     ('FIELD'     -
/                     ,'FIELD-IND' -
/                     ,'TYPE'      -
/                     ,'HILITE'    -
/                     ,'INTENSITY' -
/                     ,'COLOR'     -
/                     ,'OUTPUT'    -
/                     )
/            &VARIABLE-NAME..ATTR#I.&PARAM = &PARAM
/         end-for
/         I=I+1
/      end-while
/   end-if
/
/else
/  write-text '%  form=&INCLUDE-FORM not supported; include aborts'
/  raise-error
/end-if
/EXIT-PROCEDURE
```

Note:

The standard header is linked in this procedure in an include. This is an S procedure that is supplied with SDF-P and is responsible for function identifications and return code and information (see TRANSMIT-BY-STREAM in the chapter on “SDF-P commands” for details).

Include for the standard header

This standard header is supplied with SDF-P. It is called in the procedure SYSFHS (with /INCLUDE-PROCEDURE).
The procedure is stored in the member FHDR in the library SYSPRC.SDF-P-BASYS.020.

This standard header can be defined as the first element of the structure of control variables.

```
/SET-PROCEDURE-OPTIONS &* CALLER=INCLUDE
/“NOT SUPPORTED BY SDF-P-BASYS“
/BEGIN-PARAMETER-DECLARATION
/    /DECLARE-PARAM -
/    “----------------------- STD PARAM ------------------------------------*“-
/                   (PREFIX                    (INITIAL-VALUE=´SYSSDP´) -
/                   ,INCLUDE-FORM    (INITIAL-VALUE=´LAYOUT´) “INITIALIZE“ -
/                   ,VARIABLE-NAME (INITIAL-VALUE=´´)  -
/    “----------------------- INCLUDE SPSECIFIC PARAM ----------------------*“-
/                   ,UNIT                (INITIAL-VALUE=´´)  -
/                   ,FUNCTION            (INITIAL-VALUE=´´)  -
/                   ,VERSION             (INITIAL-VALUE=0)  -
/                   ,SUBCODE2            (INITIAL-VALUE=0)  -
/                   ,SUBCODE1            (INITIAL-VALUE=0)  -
/                   ,MAINCODE            (INITIAL-VALUE=´CMD0001´)  -
/                    )
/END-PARAMETER-DECLARATION
/
/IF (NOT IS-SDF-P( ))
/    EXIT-PROCEDURE ERROR=*YES(SUBCODE1=41,MAINCODE=CMD2241)
/END-IF
/
/IF (UPPPER-CASE(INCLUDE-FORM) == ´LAYOUT´)
/
/BEGIN-STRUCTURE NAME=&PREFIX.IFID-MDL,SCOPE=*PROCEDURE
/    DECLARE-ELEMENT -
/                    (UNIT             (TYPE=*STRING)  -
/                   ,FUNCTION          (TYPE=*STRING)  -
/                   ,VERSION           (TYPE=*INTEGER)  -
/                    )
/END-STRUCTURE
/BEGIN-STRUCTURE NAME=&PREFIX.RETC-MDL,SCOPE=*PROCEDURE
/    DECLARE-ELEMENT -
/                   ,SUBCODE2          (TYPE=*INTEGER)  -
/                   ,SUBCODE1          (TYPE=*INTEGER)  -
/                   ,MAINCODE          (TYPE=*STRING)  -
/)
/END-STRUCTURE
/BEGIN-STRUCTURE NAME=&PREFIX.FHDR,SCOPE=*PROCEDURE
```

```
/    DECLARE-ELEMENT INTERFACE-ID(TYPE=*STRUCTURE(&PREFIX.IFID-MDL))
/    DECLARE-ELEMENT RETURNCODE(TYPE=*STRUCTURE(&PREFIX.RETC-MDL))
/END-STRUCTURE
/
/ELSE-IF (UPPER-CASE(INCLUDE-FORM) == ´INITIALIZE´)
/
/    IF (VARIABLE-NAME == ´´)
/          WRITE-TEXT ´% OBLIGATORISCHER OPERAND VARIABLE-NAME GEHT AB.´
/          RAISE-ERROR
/    END-IF
/    DECLARE-VARIABLE PARAM(TYPE=*STRING)
/    FOR PARAM = (´UNIT´,´FUNCTION´)
/         &VARIABLE-NAME..INTERFACE-ID.&PARAM = &PARAM
/    END-FOR
/    &VARIABLE-NAME..INTERFACE-ID.VERSION=INTEGER(VERSION)
/    FOR PARAM = (´SUBCODE2´,´SUBCODE1´)
/         &VARIABLE-NAME..RETURNCODE.&PARAM = INTEGER(&PARAM)
/    END-FOR
/    &VARIABLE-NAME..RETURNCODE.MAINCODE=MAINCODE
/ELSE
/          WRITE-TEXT ´% FORM = &INCLUDE-FORM WIRD NICHT UNTERSTUETZT; -
/INCLUDE WIRD ABGEBROCHEN´
/          RAISE-ERROR
/END-IF
/EXIT-PROCEDURE
```

The server does not check whether this header exists, so the header may be omitted when specifying the control variables.

Description of parameters

UNIT (type=string)
Specifies the server that has defined the control variable. This should be the same name as the one specified with /ASSIGN-STREAM.

FUNCTION (type=string)
Specifies the function for which the server has defined the layout of the control variables. The name is defined and processed by the server.

VERSION (type=integer)
Version of the control variables. This allows earlier versions of the control variables to be compatibly supported by the server.

SUBCODE2 (type=integer)
“Subcode2” value returned by the server (in RET-CONTROL-VAR-NAME).

SUBCODE1 (type=integer)
“Subcode1” value returned by the server (in RET-CONTROL-VAR-NAME).

MAINCODE (type=string)
Message ID returned by the server (in RET-CONTROL-VAR-NAME).

The parameters SUBCODE2, SUBCODE1, and MAINCODE are subject to the same conventions that apply to the command return code, e.g. IDHA001 (see page 234).

These control variables should be returned by the server in order to enable more precise error handling than is possible with the return codes of the TRANSMIT-BY-STREAM or TRANSVV commands.

5.6 Differences between FHS-DM and FHS-PRIV

P key assignments

In contrast to FHS-TU, the value of P-KEY setting cannot be stored in a user file.

The system variables KEY-LIST and KEY-AEREA do not exist.

CCS name and 8-bit mode

The CCS name cannot be specified; the system variable CCSNAME does not exist.

Users who wish to work in 8-bit mode must enter the command “MODIFY-TERMINAL-OPTION“” before the assignment.

SYS command

The SYS command is rejected in TPR mode.

6 Sample programs

The following is a COBOL example that uses FHS-DM as a user interface.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. COB7.
*****************************************************************
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
    TERMINAL IS VIDEO.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 DMCOMM.
 02 DMRC IS COMP.
  03 DMSC2 PIC S9(4).
  03 DMSC1 PIC S9(4).
  03 DMMC PIC S9(7).
 02 DMMSGID PIC X(8).
 02 DMFLAG.
  03   DMERR                  PIC X(1).
  03   FILLER                 PIC X(7).
 02   DMSYS                    PIC X(104).
77   BUFFER PIC X(512).
77   LEN    PIC S9(5) COMP VALUE 512.
77   VAR1   PIC X(6).
77   VAR1LEN PIC S9(5) COMP.
*
************************************ PROGRAM START
*
PROCEDURE DIVISION.
*****************************************************************
CONTR SECTION.
CONTR-0.
PERFORM MAIN.
CONTR-9.
DISPLAY „PROGRAM COB7 TERMINATED“ UPON VIDEO.
STOP RUN.
*****************************************************************
```

```
MAIN SECTION.

MAIN-0.
*
       MOVE „ DMOPEN INIT PROFILE(DATA1)“ TO BUFFER.
       CALL „ISPCI“ USING DMCOMM LEN BUFFER.
       PERFORM CHECK-FHS-RC.
*
       MOVE „ VDEFINE VAR1 FORMAT(CHAR)“ TO BUFFER.
       MOVE 6 TO VAR1LEN.
       CALL „ISPCI2“ USING DMCOMM LEN BUFFER VAR1LEN VAR1.
       PERFORM CHECK-FHS-RC.
*
       MOVE „ VGET VAR1 PROFILE    „ TO BUFFER.
       CALL „ISPCI“ USING DMCOMM LEN BUFFER.
       PERFORM CHECK-FHS-RC.
*      .
*      .
*      .
*      vdefine of other variables
*      .
*      .
*      .
*
*  See Explanation (1)
       MOVE „ DISPLAY PANEL(V81) „ TO BUFFER.
       CALL „ISPCI“ USING DMCOMM LEN BUFFER.
       PERFORM CHECK-FHS-RC.
*
       MOVE „ ADDPOP „ TO BUFFER.
       CALL „ISPCI“ USING DMCOMM LEN BUFFER.
       PERFORM CHECK-FHS-RC.

*  See Explanation (2)
       MOVE „ DISPLAY PANEL(BOXV81) „ TO BUFFER.
       CALL „ISPCI“ USING DMCOMM LEN BUFFER.
       PERFORM CHECK-FHS-RC.
*  See Explanation (3)
       MOVE „ DISPLAY MSG(IDHT017   ) „ TO BUFFER.
       CALL „ISPCI“ USING DMCOMM LEN BUFFER.
       PERFORM CHECK-FHS-RC.
*
       MOVE „ VPUT VAR1 PROFILE       „ TO BUFFER.
       CALL „ISPCI“ USING DMCOMM LEN BUFFER.
       PERFORM CHECK-FHS-RC.
*
```

```
 MOVE „ DMCLOSE „ TO BUFFER.
       CALL „ISPCI“ USING DMCOMM LEN BUFFER.
       PERFORM CHECK-FHS-RC.
       EXIT.
*
*****************************************************************
 CHECK-FHS-RC SECTION.
 CHECK-0.
       IF DMSC1 IN DMRC IN DMCOMM NOT = 0
          THEN DISPLAY „RETURNCODE:“ DMSC2 „/“
               DMSC1 „/“ DMMC „      „
               DMMSGID UPON VIDEO.
       EXIT.
*****************************************************************
```

A corresponding sample program in ASSEMBLER

```
         MACRO
         FILLBUF &STMT
         MVI   BUFFER,' '
         MVC   BUFFER+1(L'BUFFER-1),BUFFER
         MVC   BUFFER(L'&STMT),&STMT
         MEND
*
ASM7     CSECT
ASM7     AMODE ANY
ASM7     RMODE ANY
         USING *,10
         BALR  10,0
         BCTR  10,0
         BCTR  10,0
R1       EQU   1
R2       EQU   2
R7       EQU   7
R14      EQU   14
R15      EQU   15
*
         LA    R1,PL
*
         L     R15,ISPCI@
         FILLBUF STMT1
         BASR  R14,R15
         BAL   R7,CHECKRC
*
         L     R15,ISPCI2@
         LA    R2,VAR1
         ST    R2,OPER2@
         LA    R2,VAR1L
         ST    R2,OPER2L@
         FILLBUF STMT2
         BASR  R14,R15
         BAL   R7,CHECKRC
*
         L     R15,ISPCI@
         FILLBUF STMT3
         BASR  R14,R15
         BAL   R7,CHECKRC
*        .
*        .
*        .
*        vdefine of other variables
```

```
*        .
*        .
*        .
*
*   See Explanation (1)
         FILLBUF STMT4
         BASR  R14,R15
         BAL   R7,CHECKRC
*
         FILLBUF STMT5
         BASR  R14,R15
         BAL   R7,CHECKRC
*   See Explanation (2)
         FILLBUF STMT6
         BASR  R14,R15
         BAL   R7,CHECKRC
*   See Explanation (3)
         FILLBUF STMT7
         BASR  R14,R15
         BAL   R7,CHECKRC
*
         FILLBUF STMT8
         BASR  R14,R15
         BAL   R7,CHECKRC
*
         FILLBUF STMT9
         BASR  R14,R15
         BAL   R7,CHECKRC*
         TERM
*-------------------------------------------------------
CHECKRC  DS    0H
         CLC   DMSC1,=X'0000'
         BE    OK
         TERMD
OK       BR    R7
*-------------------------------------------------------
*
STMT1    DC    C'DMOPEN INIT PROFILE (DATA1)
STMT2    DC    C'VDEFINE VAR1 FORMAT(CHAR)
STMT3    DC    C'VGET VAR1 PROFILE
*        .
*        .
*        .
*        other „vdefine“ statements
*        .
*        .
*        .
STMT4    DC    C'DISPLAY PANEL(V81)
```

```
STMT5    DC    C'ADDPOP
STMT6    DC    C'DISPLAY PANEL(BOXV81)
STMT7    DC    C'DISPLAY MSG(IDHT017)
STMT8    DC    C'VPUT VAR1 PROFILE
STMT9    DC    C'DMCLOSE
*
PL       DS    5A
         ORG   PL
DMCOMM@  DC    A(DMCOMM)
BUFLEN@  DC    A(BUFLEN)
BUFFER@  DC    A(BUFFER)
OPER2L@  DC    A(0)
OPER2@   DC    A(0)
         ORG
*
DMCOMM   DS    0F
DMSC2    DS    H
DMSC1    DS    H
DMMC     DS    F
DMMSGID  DS    CL8
DMERR    DS    C
RESERVD1 DS    XL7
DMSYS    DS    XL104
*
VAR1     DS    CL6
VAR1L    DC    A(L'VAR1)
BUFLEN   DC    A(L'BUFFER)
BUFFER   DC    CL256' '
*
ISPCI@   DC    V(ISPCI)
ISPCI2@  DC    V(ISPCI2)
*
         END
```

6.1 Calling FHS-DM as a subsystem

If FHS-DM is loaded as a subsystem, it can be called from any main program that calls unresolved ISPCI and ISPCI2 calls.

Make sure that these external references are dynamically resolved at runtime by the dynamic binder loader DBL!

If the FHS-DM V8.1 subsystem was loaded at an address within or above the 16MB address space, the application of the user must be created with AMODE=ANY and started with /START-PROGRAM xxx, PROGRAM-MODE=ANY

If FHS-DM V8.1 was loaded as a subsystem exclusively below the 16 MB address space, the PROGRAM-MODE parameter need not be specified in the /START-PROGRAM command.

If FHS has already been linked with the application statically, it will not be possible to call the FHS subsystem (see also the next section).

Examples of link procedures

A. Main program in COBOL

```
           /ASSIGN-SYSLST COB7.LNKLST
           /ASSIGN-SYSDTA *SYSCMD
           /START-BINDER
           //START-LLM-CREATION INTER-NAME=TESTCOB
           //BEGIN-SUB-LLM APPLIC
(1)        //INCLUDE-MODULE LIB=COB.LIB,E=COB7,TYPE=R
           //END-SUB-LLM
           //BEGIN-SUB-LLM RT
(2)        //RESOLVE-BY-AUTOLINK $.SYSLNK.CRTE,TYPE=R,SCOPE=*WHOLE-LLM
           //END-SUB-LLM
           //SAVE-LLM LIBRARY=COB.LIB,ELEM=TESTCOB,MAP=YES
           //END
           /ASSIGN-SYSLST *PRIMARY
           /ASSIGN-SYSDTA *PRIMARY
```

B. Main program in ASSEMBLER

```
           /ASSIGN-SYSLST ASM7.LNKLST
           /ASSIGN-SYSDTA *SYSCMD
           /START-BINDER
           //START-LLM-CREATION INTER-NAME=TESTASM
           //BEGIN-SUB-LLM APPLIC
(1)        //INCLUDE-MODULE LIB=ASM.MODLIB,E=ASM7,TYPE=R
           //END-SUB-LLM
           //SAVE-LLM LIBRARY=ASM.MODLIB,ELEM=TESTASM,MAP=YES
           //END
           /ASSIGN-SYSLST *PRIMARY
           /ASSIGN-SYSDTA *PRIMARY
```

(1) Your application program

(2) The CRTE runtime system for COBOL

This program produces two unresolved external references at ISPCI and ISPCI2. These external references are resolved at runtime with the FHS-DM subsystem.

You can start these programs with the following command:

```
/START-PROG *M(library,element,RUN-MODE=ADV)
```

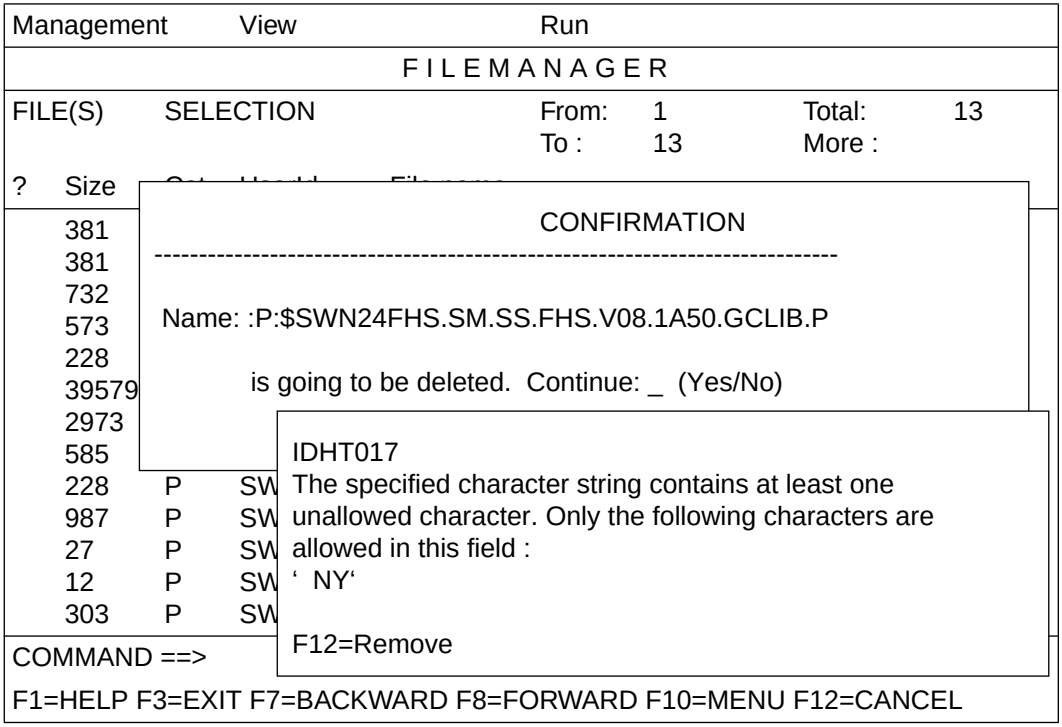
Explanation (1)

```
Management          View                    Run
                    F I L E M A N A G E R
FILE(S)             SELECTION               From:   1       Total:  13
                                            To :    13      More :
?   Size    Cat.    UserId      File name
    381     P       SWN24FHS SM.SS.FHS.V08.0A70.GCLIB.P
    381     P       SWN24FHS SM.SS.FHS.V08.0A70.GCLIB.UR
    732     P       SWN24FHS SM.SS.FHS.V08.0A70.SRC
    573     P       SWN24FHS SM.SS.FHS.V08.1A50.GCLIB.P
    228     P       SWN24FHS SM.SS.FHS.V08.1A50.GCLIB.UR
    39579   P       SWN24FHS SM.SS.FHS.V08.1A70.LIST
    2973    P       SWN24FHS SM.SS.FHS.V08.1A70.SRC
    585     P       SWN24FHS SM.SS.FHS.V08.1A80.GCLIB.P
    228     P       SWN24FHS SM.SS.FHS.V08.1A80.GCLIB.UR
    987     P       SWN24FHS SM.SS.FHS.V08.1A80.LIB
    27      P       SWN24FHS SM.SS.FHS.V08.1A80.LIST
    12      P       SWN24FHS SM.SS.FHS.V08.1A80.SRC
    303     P       SWN24FHS SM.SS.PROSOS-TU.V07.0A00.LIB
COMMAND ==>
F1=HELP F3=EXIT F7=BACKWARD F8=FORWARD F10=MENU F12=CANCEL
```

`Explanation` (2)

```
Management          View                          Run
                              F I L E M A N A G E R
FILE(S)        SELECTION                          From:     1          Total:        13
                                                  To :      13         More :
?    Size      Cat    UserId         File name
     381      ┌──────────────────────────────────────────────────────────────────────┐
     381      │                         CONFIRMATION                                 │
     732      │ ---------------------------------------------------------------------│
     573      │                                                                      │
     228      │ Name: :P:$SWN24FHS.SM.SS.FHS.V08.1A50.GCLIB.P                        │
     39579    │                                                                      │
     2973     │         is going to be deleted.  Continue: _  (Yes/No)               │
     585      │                                                                      │
     228      └──────────────────────────────────────────────────────────────────────┘
     987       P      SWN24FHS SM.SS.FHS.V08.1A80.LIB
     27        P      SWN24FHS SM.SS.FHS.V08.1A80.LIST
     12        P      SWN24FHS SM.SS.FHS.V08.1A80.SRC
     303       P      SWN24FHS SM.SS.PROSOS-TU.V07.0A00.LIB
COMMAND ==>
F1=HELP F3=EXIT F7=BACKWARD F8=FORWARD F10=MENU F12=CANCEL
```

Explanation (3)



6.2 Sample procedures for FHS in SDF-P

The following example shows the declaration and initialization of the variable MYVAR (without the ATTR-LIST element):

```
/include-procedure *lib-elem(lib=$.SYSPRC.SDF-P-BASYS.020,el=FHDR) -
/                 ,(PREFIX='SYSFHS-')
/include-procedure *lib-elem(lib=$.SYSPRC.FHS.081,el=SYSFHS-CONTROL)
/declare-variable  MYVAR (type = *struct(SYSFHS-LAYOUT))
/include-procedure *lib-elem(lib=$.SYSPRC.FHS.081,el=SYSFHS-CONTROL) , -
/                  (include-form='INITIALIZE',variable-name='MYVAR', -
/                  attr-list = 2 )
/
/show-variable MYVAR
```

The variable is then generated and initialized as follows:

```
MYVAR.STD-HEADER.INTERFACE-ID.UNIT = FHS
MYVAR.STD-HEADER.INTERFACE-ID.FUNCTION = DISPLAY
MYVAR.STD-HEADER.INTERFACE-ID.VERSION = 1
MYVAR.STD-HEADER.RETURNCODE.SUBCODE2 = 0
MYVAR.STD-HEADER.RETURNCODE.SUBCODE1 = 0
MYVAR.STD-HEADER.RETURNCODE.MAINCODE = CMD0001
MYVAR.SERVICE = *DISPLAY
MYVAR.DIAGINFO = *NO
MYVAR.POP-LOCATION = *NONE
MYVAR.POP-LOC-IND = 0
MYVAR.ROW = 2
MYVAR.COLUMN = 2
MYVAR.RESOURCE = *SAME
MYVAR.MESSAGE-ID = *NONE
MYVAR.MESSAGE-FIELD = *NONE
MYVAR.MSG-FIELD-IND = 0
MYVAR.CURSOR-OUTPUT-INDEX = 0
MYVAR.CURSOR-OUTPUT = *NONE
MYVAR.CURSOR-OUTPUT-POS = 0
MYVAR.LOCK = *NO
MYVAR.ALARM = *NO
MYVAR.HARDCOPY = *NO
MYVAR.AUTOTAB = *YES
MYVAR.MANDATORY = *NO
MYVAR.REFRESH = *NO
MYVAR.ACK = *NO
MYVAR.ATTR(*LIST).FIELD = *CURSOR
MYVAR.ATTR(*LIST).FIELD-IND = 0
MYVAR.ATTR(*LIST).TYPE = *UNCHANGED
MYVAR.ATTR(*LIST).HILITE = *UNCHANGED
MYVAR.ATTR(*LIST).INTENSITY = *UNCHANGED
```

```
MYVAR.ATTR(*LIST).COLOR = *UNCHANGED
MYVAR.ATTR(*LIST).OUTPUT = *UNCHANGED
MYVAR.ATTR(*LIST).FIELD = *CURSOR
MYVAR.ATTR(*LIST).FIELD-IND = 0
MYVAR.ATTR(*LIST).TYPE = *UNCHANGED
MYVAR.ATTR(*LIST).HILITE = *UNCHANGED
MYVAR.ATTR(*LIST).INTENSITY = *UNCHANGED
MYVAR.ATTR(*LIST).COLOR = *UNCHANGED
MYVAR.ATTR(*LIST).OUTPUT = *UNCHANGED
MYVAR.COMMAND =
MYVAR.FHS-VERSION =
MYVAR.CURSOR-INPUT =
MYVAR.CURSOR-INPUT-INDEX =
MYVAR.CURSOR-INPUT-POS =
```

6.2.1 Examples of working with FHS using S procedures

The next example shows how an FHS V8.1 application (that uses “FHS dialog variables”) can be controlled by means of an S procedure. The S procedure controls the fields of the IFG masks that are used by the FHS application. The “FHS formats” are simulated by means of “variable containers”. The following formats are used:

SCREEN1:

```
                                File Selection

  Select the file name (*=wildcard)

                  a*




CMD:==>
F1=Help  F3=Exit  F12=Cancel
```

SCREEN2:

```
Action
                          File Management System
                File list                          Lines  7  to  8  of  8  More:
 ?    Size     Cat  UserId     File name
   3           X    XXXXXX    AX
   3           X    XXXXXX    AXX
**************************************End of list*****************************************




CMD:==>
F1=Help  F3=Exit  F12=Cancel
```

```
/“main“ set-procedure-options
/
/include-procedure *lib-elem(lib=$.SYSPRC.SDF-P-BASYS.020,el=FHDR) -
/                 ,(PREFIX=’SYSFHS-’)
/include-procedure *lib-elem(lib=$.SYSPRC.FHS.081,el=SYSFHS-CONTROL)
/declare-variable  SYSPINFO (type = *struct(SYSFHS-LAYOUT))
/include-procedure *lib-elem(lib=$.SYSPRC.FHS.081,el=SYSFHS-CONTROL) , -
/                  (include-form=’INITIALIZE’,variable-name=’SYSPINFO’)
/
/declare-variable SCREEN1(type=*structure(*by-syscmd))
/begin-structure
/  declare-element FILE-NAME
/end-structure
/
/declare-variable SCREEN2(type=*structure(*by-syscmd))
/begin-structure
/  declare-element name = FILELIST (type = *structure(*dynamic)) -
/               , multiple-elements = *list
/  declare-element name = SDFPLIST-MODINDEX (type = integer) -
/               , multiple-elements = *list‘
/  declare-element SDFPLIST-TOPINDEX
/  declare-element ACTION-CHOICE
/  declare-element ACTION-NUMBER
/  declare-element ACTION
/end-structure
/
/NUMBER-OF-ATTEMPT = 1
/
/ „Initialize MODINDEX List for 30 elements „
/LOOP: while (NUMBER-OF-ATTEMPT <= 30)
/       SCREEN2.SDFPLIST-MODINDEX#&(NUMBER-OF-ATTEMPT) = 0
/       NUMBER-OF-ATTEMPT = NUMBER-OF-ATTEMPT + 1
/end-while LOOP
/
/assign-stream -
/              stream-name = PRESENTATION,-
/              to          = *SERVER(FHS,-
/              server-info = ’FHS-LIB = MAPLIB’)
/
/SYSPINFO.RESOURCE = ’SDFPFILE’
/SYSPINFO.SERVICE  = ’*DISPLAY’
/SYSPINFO.REFRESH  = ’*YES’
/SYSPINFO.ACK      = ’*YES’
/
/transmit-by-stream variable-name = SCREEN1, -
/                   stream-name = PRESENTATION,-
/                   control-var-name = SYSPINFO
/
```

```
/if (SYSPINFO.STD-HEADER.RETURNCODE.MAINCODE NE 'IDH0000')
/show-variable SYSPINFO.STD-HEADER
/  end-if
/
/if  ((SYSPINFO.STD-HEADER.RETURNCODE.MAINCODE EQ 'IDH0004')     -
/    OR (SYSPINFO.STD-HEADER.RETURNCODE.MAINCODE EQ 'IDH0008'))
/   exit-procedure
/  end-if
/
/execute-cmd cmd=(SHOW-FILE-ATTRIBUTES f-name=&(SCREEN1.FILE-NAME) -
/             ,INFO=NAME-AND-SPACE) -
/             , text-output         =   *NONE    -
/             , structure-output    =   SCREEN2.FILELIST
/
/repeat
/
/   SYSPINFO.RESOURCE = 'SDFPLIST'
/   SYSPINFO.REFRESH  = '*YES'
/   SYSPINFO.ACK      = '*YES'
/   SYSPINFO.SERVICE = '*DISPLAY'
/   SYSPINFO.ACK     = '*YES'
/   SCREEN2.SDFPLIST-TOPINDEX = 2
/
/   transmit-by-stream variable-name = SCREEN2, -
/                        stream-name = PRESENTATION,-
/                        control-var-name = SYSPINFO
/
/   if (SYSPINFO.STD-HEADER.RETURNCODE.MAINCODE NE 'IDH0000')
/         show-variable SYSPINFO.STD-HEADER
/      end-if
/
/until ((SYSPINFO.STD-HEADER.RETURNCODE.MAINCODE EQ 'IDH0004')     -
/    OR (SYSPINFO.STD-HEADER.RETURNCODE.MAINCODE EQ 'IDH0008'))
/
/assign-stream  stream-name = PRESENTATION, to = *dummy
/
/exit-procedure
```

Example of the use of field attributes:

```
/“attt“ set-procedure-options
/
/include-procedure *lib-elem(lib=$.SYSPRC.SDF-P-BASYS.020,el=FHDR) -
/                 ,(PREFIX=’SYSFHS-’)
/include-procedure *lib-elem(lib=$.SYSPRC.FHS.081,el=SYSFHS-CONTROL)
/declare-variable  SYSPINFO (type = *struct(SYSFHS-LAYOUT))
/include-procedure *lib-elem(lib=$.SYSPRC.FHS.081,el=SYSFHS-CONTROL) , -
/                  (include-form=’INITIALIZE’,variable-name=’SYSPINFO’,-
/                   attr-list=3)
/
/declare-variable SCREEN1(type=*structure(*by-syscmd))
/begin-structure
/  declare-element INPUT-FLD
/end-structure
/
/
/assign-stream -
/              stream-name = PRESENTATION,-
/              to          = *SERVER(FHS,-
/              server-info = ’FHS-LIB = MAPLIB’)
/
/repeat
/
/SYSPINFO.RESOURCE = ’VO3     ’
/SYSPINFO.SERVICE = ’*DISPLAY’
/SYSPINFO.ATTR#1.FIELD = ’(SYS-DATE,SYS-DAY)’
/SYSPINFO.ATTR#1.FIELD-IND = ’O’
/SYSPINFO.ATTR#1.HILITE    = ’*UNDERLINE’
/SYSPINFO.ATTR#2.FIELD = ’SYS-TIME’
/SYSPINFO.ATTR#2.INTENSITY = ’*INVISIBLE’
/
/transmit-by-stream variable-name = SCREEN1, -
/                    stream-name = PRESENTATION,-
/                    control-var-name = SYSPINFO
/
/if (SYSPINFO.STD-HEADER.RETURNCODE.MAINCODE NE ’IDHOOOO’)
/show-variable SYSPINFO.STD-HEADER
/  end-if
/
/if (SYSPINFO.COMMAND = ’usercmd’)
/     include-procedure *l(lib=userlib,el=userelem)
/  end-if
/
```

```
/until ((SYSPINFO.STD-HEADER.RETURNCODE.MAINCODE EQ 'IDH0004')     -
/    OR (SYSPINFO.STD-HEADER.RETURNCODE.MAINCODE EQ 'IDH0008'))
/
/
/assign-stream  stream-name = PRESENTATION, to = *dummy
/
/exit-procedure
```

6.3 Application example: creating a graphics-based library manager

This section contains a detailed application example to demonstrate how the interaction between S procedures, S variable streams, and FHS can be used to create a graphics-based library manager.

The individual S procedures for this purpose are maintained (internally) in the library LIBRARY-MANAGER.PL as members of type J under the names RUN, SCREEN01, and SCREEN02 , respectively. RUN is the controlling S procedure that is called by CALL-PROCEDURE, while SCREEN01 and SCREEN02, by contrast, handle the two default screen displays under its control.

A listing of each of these three procedures is provided below, followed by some applications to demonstrate how the FHS-supported library manager can be used.

Procedure: RUN

```
/SET-PROCEDURE-OPTIONS CALLER=CALL
/
/"-------------------------------------------------------------------------"
/"FIRST EXTRACT LIBRARY NAME FROM WHICH THIS PROCEDURE IS CALLED"
/"-------------------------------------------------------------------------"
/DECLARE-VARIABLE SYSOUT(TYPE=*STRING),MULTIPLE-ELEMENTS=*LIST
/ASSIGN-SYSOUT TO=*VARIABLE(SYSOUT)
/SHOW-SYSTEM-FILE-ASSIGNMENT SYSTEM-FILE=*SYSCMD
/ASSIGN-SYSOUT *PRIMARY
/
/LIB-ELEM-POS = INDEX(SYSOUT#2,'*LIB-ELEM')
/LIBRARY-NAME = SUBSTRING(SYSOUT#2,LIB-ELEM-POS + LENGTH('*LIB-ELEM('))
/COMMA = INDEX(LIBRARY-NAME,',')
/LIBRARY-NAME = SUBSTRING(LIBRARY-NAME,1,COMMA - 1)
/
/"-------------------------------------------------------------"
/"INITIALIZE FHS CONTROL VARIABLES                             "
/"-------------------------------------------------------------"
/WRITE-TEXT 'LIBRARY MANAGER V1.0 - LOADING'
/INCLUDE-PROCEDURE *LIBRARY-ELEMENT(LIBRARY = $.SYSPRC.SDF-P-BASYS.020 -
/                                  ,ELEMENT = FHDR) -
/                 ,PROCEDURE-PARAMETERS = (PREFIX = 'SYSFHS-')
/INCLUDE-PROCEDURE *LIBRARY-ELEMENT(LIBRARY = $.SYSPRC.FHS.081 -
/                                  ,ELEMENT = SYSFHS-CONTROL)
/DECLARE-VARIABLE  SYSPINFO (TYPE = *STRUCTURE(SYSFHS-LAYOUT))
/DECLARE-VARIABLE  SYSPINFO-SAVE (TYPE = *STRUCTURE(SYSFHS-LAYOUT))
/INCLUDE-PROCEDURE *LIBRARY-ELEMENT(LIBRARY = $.SYSPRC.FHS.081 -
/                                  ,ELEMENT = SYSFHS-CONTROL) -
/                 ,PROCEDURE-PARAMETERS = (INCLUDE-FORM='INITIALIZE' -
```

```
/                                          ,VARIABLE-NAME='SYSPINFO')
/
/"-------------------------------------------------------------"
/"ASSIGN THE STREAM TO FHS                                     "
/"-------------------------------------------------------------"
/ASSIGN-STREAM STREAM-NAME = PRESENTATION -
/              ,TO          = *SERVER(FHS -
/              ,SERVER-INFO = 'FHS-LIB = &LIBRARY-NAME.')
/
/"-------------------------------------------------------------"
/"START LMS V3.0                                               "
/"-------------------------------------------------------------"
/ASSIGN-SYSOUT TO=*DUMMY
/START-LMS
/HOLD-PROGRAM
/ASSIGN-SYSOUT TO=*PRIMARY
/
/"-----------------------------------------------------------------------"
/"SET TIMEOUT TO 0 WHEN SWITCHING FROM LINE MODE TO FULL SCREEN    "
/"-----------------------------------------------------------------------"
/MODIFY-TERMINAL-OPTIONS OVERFLOW-CONTROL = *TIME(TIMEOUT = 0)
/
/"-------------------------------------------------------------"
/"CALL MAIN PROCEDURE (SCREEN01)                               "
/"-------------------------------------------------------------"
/INCLUDE-PROCEDURE *LIBRARY-ELEMENT(LIBRARY = &LIBRARY-NAME. -
/                                  ,ELEMENT = SCREEN01)
/
/"-------------------------------------------------------------"
/"STOP  LMS V3.0                                               "
/"-------------------------------------------------------------"
/RESUME-PROGRAM
//END
```

Procedure: SCREEN01

```
/DECLARE-VARIABLE SCREEN01(TYPE=*STRUCTURE(*BY-SYSCMD))
/BEGIN-STRUCTURE
/  DECLARE-ELEMENT NAME = FILELIST (TYPE = *STRUCTURE(*BY-SYSCMD)) -
/                 ,MULTIPLE-ELEMENTS = *LIST
/  BEGIN-STRUCTURE
/    DECLARE-ELEMENT CHOICE
/    DECLARE-ELEMENT F-SIZE
/    DECLARE-ELEMENT CAT-ID
/    DECLARE-ELEMENT USER-ID
/    DECLARE-ELEMENT SHORT-F-NAME
/  END-STRUCTURE
/  DECLARE-ELEMENT NAME = SDFPLIST-MODINDEX (TYPE = INTEGER) -
/                 ,MULTIPLE-ELEMENTS = *LIST
/  DECLARE-ELEMENT SDFPLIST-TOPINDEX(INITIAL-VALUE = 1)
/  DECLARE-ELEMENT SDFPLIST-BOTINDEX
/  DECLARE-ELEMENT SDFPLIST-NUMROW
/  DECLARE-ELEMENT FILE-MENU
/  DECLARE-ELEMENT FILE-CHOICE
/END-STRUCTURE
/
/DECLARE-VARIABLE I(TYPE = *INTEGER)
/
/WHILE (TRUE)
/
/     "INITIALIZE MODINDEX LIST FOR 50 ELEMENTS "
/     " (FHS REQUIREMENT)                       "
/     I = 1
/     WHILE (I <= 50)
/        SCREEN01.SDFPLIST-MODINDEX#&(I) = 0
/        I = I + 1
/     END-WHILE
/
/     "GET LIBRARY NAMES"
/     EXEC-CMD CMD=(SHOW-FILE-ATTRIBUTES -
/                     SELECT=*BY-ATTRIBUTES(TYPE-OF-FILES = *PLAM-LIBRARY) -
/                    ,INFO=*NAME-AND-SPACE -
/                  ) -
/              ,STRUCTURE-OUTPUT=SCREEN01.FILELIST -
/              ,TEXT-OUTPUT=*NONE -
/              ,RETURNCODE=*VARIABLE(SUBCODE2=SUB2 -
/                                   ,SUBCODE1=SUB1 -
/                                   ,MAINCODE=MAIN)
/
/     IF (SUB1 NE 0)
/        WRITE-TEXT 'ERROR &SUB2 &SUB1 &MAIN RETURNED BY EXEC-CMD'
/        WRITE-TEXT 'LIBRARY MANAGER V1.0 ABNORMALLY TERMINATED
```

```
/        EXIT-PROCEDURE
/     END-IF
/
/     "FOLLOWING LOOP IS ONLY NECESSARY TO REP A PROBLEM BETWEEN"
/     "FHS AND VAS. CORRECTION IN VAS V02.0A85, FHS V08.1A75"
/     I = 1
/     WHILE (I <= SIZE('SCREEN01.FILELIST'))
/         SCREEN01.FILELIST#I.CHOICE = ' '
/         I = I + 1
/     END-WHILE
/
/     SYSPINFO.RESOURCE = 'SCREEN01'
/     SYSPINFO.SERVICE = '*DISPLAY'
/     SYSPINFO.REFRESH = '*YES'
/     SYSPINFO.COMMAND = ''
/     SCREEN01.SDFPLIST-NUMROW = SIZE('SCREEN01.FILELIST')
/     SCREEN01.FILE-MENU=0
/     SCREEN01.FILE-CHOICE=0
/     TRANSMIT-BY-STREAM VARIABLE-NAME = SCREEN01 -
/                       ,STREAM-NAME = PRESENTATION -
/                       ,CONTROL-VAR-NAME = SYSPINFO
/
/     IF (   (SYSPINFO.STD-HEADER.RETURNCODE.MAINCODE == 'IDH0004') -
/         OR (SYSPINFO.STD-HEADER.RETURNCODE.MAINCODE == 'IDH0008'))
/        WRITE-TEXT 'LIBRARY MANAGER V1.0 NORMALLY TERMINATED'
/        EXIT-PROCEDURE
/     END-IF
/
/     IF (SYSPINFO.STD-HEADER.RETURNCODE.MAINCODE NE 'IDH0000')
/        SUB2 = SYSPINFO.STD-HEADER.RETURNCODE.SUBCODE2
/        SUB1 = SYSPINFO.STD-HEADER.RETURNCODE.SUBCODE1
/        MAIN = SYSPINFO.STD-HEADER.RETURNCODE.MAINCODE
/        WRITE-TEXT 'ERROR &SUB2 &SUB1 &MAIN RETURNED BY FHS SERVER'
/        WRITE-TEXT 'LIBRARY MANAGER V1.0 ABNORMALLY TERMINATED'
/        EXIT-PROCEDURE
/     END-IF
/
/     IF SCREEN01.FILE-MENU NE 0
/        IF SCREEN01.FILE-CHOICE == 9
/           WRITE-TEXT 'LIBRARY MANAGER V1.0 NORMALLY TERMINATED'
/           EXIT-PROCEDURE
/        ELSE-IF SCREEN01.FILE-CHOICE == 1
/           I = 1
/           WHILE (SCREEN01.SDFPLIST-MODINDEX#I NE 0)
/              SCREEN01-CURR-INDEX = SCREEN01.SDFPLIST-MODINDEX#I
/              IF SCREEN01.FILELIST#SCREEN01-CURR-INDEX.CHOICE == '/
/                 SYSPINFO-SAVE = SYSPINFO
/                 INCLUDE-PROCEDURE -
```

```
/                    NAME=*LIBRARY-ELEMENT(&LIBRARY-NAME. -
/                                         ,SCREEN02) -
/                   ,PROCEDURE-PARAMETERS=(&(SCREEN01.FILELIST#SCREEN01-
                                             CURR-INDEX.SHORT-F-NAME))
/                IF-CMD-ERROR
/                   WRITE-TEXT 'LIBRARY MANAGER V1.0 ABNORMALLY TERMINATED'
/                   EXIT-PROCEDURE
/                ELSE
/                   SAVE-RETURNCODE
/                   IF (MAINCODE() = 'STOPOOK')
/                      WRITE-TEXT 'LIBRARY MANAGER V1.0 NORMALLY TERMINATED'
/                      EXIT-PROCEDURE
/                   END-IF
/                END-IF
/                SYSPINFO = SYSPINFO-SAVE
/             END-IF
/             I = I + 1
/          END-WHILE
/       END-IF
/    END-IF
/
/    IF (SYSPINFO.COMMAND NE '')
/       EXEC-CMD CMD=(&(SYSPINFO.COMMAND)) -
/               ,TEXT-OUTPUT=*NONE -
/               ,RETURNCODE=*VARIABLE(SUBCODE2=SUB2 -
/                                    ,SUBCODE1=SUB1 -
/                                    ,MAINCODE=MAIN)
/
/       IF (SUB1 NE 0)
/          WRITE-TEXT 'ERROR &SUB2 &SUB1 &MAIN RETURNED BY COMMAND SERVER'
/          WRITE-TEXT 'LIBRARY MANAGER V1.0 ABNORMALLY TERMINATED'
/          EXIT-PROCEDURE
/       END-IF
/    END-IF
/END-WHILE
```

Procedure: SCREEN02

```
/BEGIN-PARAMETER-DECLARATION
/  DECLARE-PARAMETER LIBRARY
/END-PARAMETER-DECLARATION
/
/DECLARE-VARIABLE SCREEN02(TYPE=*STRUCTURE(*BY-SYSCMD))
/BEGIN-STRUCTURE
/  DECLARE-ELEMENT NAME = ELEMLIST(TYPE = *STRUCTURE(*DYNAMIC)) -
/                  ,MULTIPLE-ELEMENT = *LIST
/  DECLARE-ELEMENT NAME = SDFPLIST-MODINDEX(TYPE = *INTEGER) -
/                  ,MULTIPLE-ELEMENT = *LIST
/  DECLARE-ELEMENT SDFPLIST-TOPINDEX(INITIAL-VALUE = 1)
/  DECLARE-ELEMENT SDFPLIST-BOTINDEX
/  DECLARE-ELEMENT SDFPLIST-NUMROW
/  DECLARE-ELEMENT FILE-MENU
/  DECLARE-ELEMENT FILE-CHOICE
/END-STRUCTURE
/
/DECLARE-VARIABLE SYSOUT(TYPE = *STRING), MULTIPLE-ELEMENTS = *LIST
/DECLARE-VARIABLE ERROR-ON-PRINT( TYPE = *BOOLEAN, INITIAL-VALUE = FALSE )
/DECLARE-VARIABLE I(TYPE = *INTEGER)
/
/RESUME-PROGRAM
//OPEN-LIBRARY LIBRARY = &LIBRARY.,MODE = *UPDATE
/HOLD-PROGRAM
/
/WHILE (TRUE)
/
/   "INITIALIZE MODINDEX LIST FOR 50 ELEMENTS "
/   " (FHS REQUIREMENT)                       "
/   I = 1
/   WHILE (I <= 50)
/      SCREEN02.SDFPLIST-MODINDEX#&(I) = 0
/      I = I + 1
/   END-WHILE
/
/   ASSIGN-SYSOUT TO = *VARIABLE(SYSOUT)
/   RESUME-PROGRAM
//  SHOW-ELEMENT-ATTRIBUTES -
//       ELEMENT = *LIBRARY-ELEMENT(LIBRARY = *STD -
//                                 ,ELEMENT = *ALL ( VERSION = *ALL ) -
//                                 ,TYPE = *ALL) -
//      ,INFORMATION = *MAXIMUM -
//      ,SORT = *BY-NAME -
//      ,STRUCTURE-OUTPUT = SCREEN02.ELEMLIST
/   HOLD-PROGRAM
/   ASSIGN-SYSOUT TO = *PRIMARY
```

```
/
/   IF ( STMT-SPINOFF() == 'YES' )
/      SHOW-VARIABLE SYSOUT, INFORMATION = *PARAMETERS( NAME = *NONE )
/      MAINCODE = 'LMSOERR'
/      GOTO END
/   END-IF
/
/   "FOLLOWING LOOP IS ONLY NECESSARY TO REP A PROBLEM BETWEEN"
/   "FHS AND VAS. CORRECTION IN VAS V02.0A85, FHS V08.1A75"
/   I = 1
/   WHILE (I <= SIZE('SCREEN02.ELEMLIST'))
/       SCREEN02.ELEMLIST#I.CHOICE = ' '
/       I = I + 1
/   END-WHILE
/
/   SYSPINFO.RESOURCE = 'SCREEN02'
/   SYSPINFO.SERVICE = '*DISPLAY'
/   SYSPINFO.REFRESH = '*YES'
/   SYSPINFO.COMMAND = ''
/   SCREEN02.SDFPLIST-NUMROW = SIZE('SCREEN02.ELEMLIST')
/   SCREEN02.FILE-MENU=0
/   SCREEN02.FILE-CHOICE=0
/   TRANSMIT-BY-STREAM VARIABLE-NAME = SCREEN02 -
/                     ,STREAM-NAME = PRESENTATION -
/                     ,CONTROL-VAR-NAME = SYSPINFO
/
/   IF (   (SYSPINFO.STD-HEADER.RETURNCODE.MAINCODE == 'IDH0004') -
/       OR (SYSPINFO.STD-HEADER.RETURNCODE.MAINCODE == 'IDH0008'))
/      MAINCODE = 'FHSEXIT'
/      GOTO END
/   END-IF
/
/   IF (SYSPINFO.STD-HEADER.RETURNCODE.MAINCODE NE 'IDH0000')
/      SUB2 = SYSPINFO.STD-HEADER.RETURNCODE.SUBCODE2
/      SUB1 = SYSPINFO.STD-HEADER.RETURNCODE.SUBCODE1
/      MAIN = SYSPINFO.STD-HEADER.RETURNCODE.MAINCODE
/      WRITE-TEXT 'ERROR &SUB2 &SUB1 &MAIN RETURNED BY FHS SERVER
/      MAINCODE = 'FHSOERR'
/      GOTO END
/   END-IF
/
/   IF SCREEN02.FILE-MENU NE 0
/      IF SCREEN02.FILE-CHOICE == 9
/         MAINCODE = 'FHSORET'
/         GOTO END
/      ELSE
/         I = 1
/         WHILE (SCREEN02.SDFPLIST-MODINDEX#I NE 0)
```

```
/             SCREEN02-CURR-INDEX = SCREEN02.SDFPLIST-MODINDEX#I
/             IF SCREEN02.ELEMLIST#SCREEN02-CURR-INDEX.CHOICE == '/'
/                ELEMENT = SCREEN02.ELEMLIST#SCREEN02-CURR-INDEX.ELEM
/                VERSION = SCREEN02.ELEMLIST#SCREEN02-CURR-INDEX.VERSION
/                TYPE    = SCREEN02.ELEMLIST#SCREEN02-CURR-INDEX.TYPE
/                IF SCREEN02.FILE-CHOICE == 1 "DELETE ELEMENT"
/                   ASSIGN-SYSOUT TO = *VARIABLE(SYSOUT)
/                   RESUME-PROGRAM
//                  DELETE-ELEMENT ELEMENT = *LIBRARY-ELEMENT -
//                                           ( LIBRARY = *STD -
//                                           , ELEMENT = &ELEMENT.-
//                                               ( VERSION = &VERSION. ) -
//                                           , TYPE = &TYPE. )
/                   HOLD-PROGRAM
/                   ASSIGN-SYSOUT *PRIMARY
/                ELSE-IF SCREEN02.FILE-CHOICE == 2 "EDIT ELEMENT"
/                   ASSIGN-SYSOUT TO = *VARIABLE(SYSOUT)
/                   RESUME-PROGRAM
//                  EDIT-ELEMENT ELEMENT = *LIBRARY-ELEMENT -
//                                           ( LIBRARY = *STD -
//                                           , ELEMENT = &ELEMENT. -
//                                               ( VERSION = &VERSION. ) -
//                                           , TYPE = &TYPE. )
/                   HOLD-PROGRAM
/                   ASSIGN-SYSOUT *PRIMARY
/                ELSE-IF SCREEN02.FILE-CHOICE == 3 "COPY ELEMENT"
/                   WRITE-TEXT 'FUNCTION NOT IMPLEMENTED'
/                ELSE-IF SCREEN02.FILE-CHOICE == 4 "PRINT ELEMENT"
/                   ASSIGN-SYSOUT TO = *VARIABLE(SYSOUT)
/                   PRINT-FILE *LIBRARY-ELEMENT -
/                                ( LIBRARY = &LIBRARY. -
/                                , ELEMENT = &ELEMENT. -
/                                    ( VERSION = &VERSION. ) -
/                                , TYPE = &TYPE. )
/                   IF-CMD-ERROR
/                      ERROR-ON-PRINT = TRUE
/                   END-IF
/                   ASSIGN-SYSOUT *PRIMARY
/                ELSE-IF SCREEN02.FILE-CHOICE == 5 "SELECT ELEMENT"
/                   ASSIGN-SYSOUT TO = *VARIABLE(SYSOUT)
/                   RESUME-PROGRAM
//                  EXTRACT-ELEMENT ELEMENT = *LIBRARY-ELEMENT -
//                                           ( LIBRARY = *STD -
//                                           , ELEMENT = &ELEMENT. -
//                                               ( VERSION = &VERSION. ) -
//                                           , TYPE = &TYPE. ) -
//                                 ,TO-FILE = *STD
/                   HOLD-PROGRAM
```

```
/                  ASSIGN-SYSOUT *PRIMARY
/               ELSE-IF SCREEN02.FILE-CHOICE == 6 "ADD ELEMENT"
/                  WRITE-TEXT 'FUNCTION NOT IMPLEMENTED'
/               END-IF
/
/               IF ( ERROR-ON-PRINT )
/                  SHOW-VARIABLE SYSOUT, INFORMATION = *PARAMETERS( NAME =
*NONE )
/                  MAINCODE = 'PRTOERR'
/                  GOTO END
/               END-IF
/
/               IF ( STMT-SPINOFF() == 'YES' )
/                  SHOW-VARIABLE SYSOUT, INFORMATION = *PARAMETERS( NAME =
*NONE )
/                  MAINCODE = 'LMSOERR'
/                  GOTO END
/               END-IF
/            END-IF
/            I = I + 1
/         END-WHILE
/      END-IF
/   END-IF
/
/   IF (SYSPINFO.COMMAND NE '')
/      EXEC-CMD CMD=(&(SYSPINFO.COMMAND)) -
/               ,TEXT-OUTPUT=*NONE -
/               ,RETURNCODE=*VARIABLE(SUBCODE2=SUB2 -
/                                     ,SUBCODE1=SUB1 -
/                                     ,MAINCODE=MAIN)
/
/      IF (SUB1 NE 0)
/         WRITE-TEXT 'ERROR &SUB2 &SUB1 &MAIN RETURNED BY COMMAND SERVER'
/         MAINCODE = 'CMDOERR
/         GOTO END
/      END-IF
/   END-IF
/END-WHILE
/
/
/END:
/IF (   (MAINCODE = 'CMDOERR') -
/    OR (MAINCODE = 'PRTOERR') -
/    OR (MAINCODE = 'LMSOERR') -
/    OR (MAINCODE = 'FHSOERR') -
/   )
/   EXIT-PROCEDURE ERROR = *YES(SUBCODE2 = 0 -
/                               ,SUBCODE1 = 64 -
```

```
/                              ,MAINCODE = STOPERR)
/ELSE-IF (MAINCODE = 'FHSEXIT')
/   EXIT-PROCEDURE ERROR = *YES(SUBCODE2 = 0 -
/                              ,SUBCODE1 = 0  -
/                              ,MAINCODE = STOPOOK)
/ELSE-IF (MAINCODE = 'FHSORET')
/   EXIT-PROCEDURE
/ELSE
/   WRITE-TEXT 'ERROR &(SC2()) &(SC1()) &(MC()) REPORTED'
/   EXIT-PROCEDURE ERROR = *YES(SUBCODE2 = 0 -
/                              ,SUBCODE1 = 64 -
/                              ,MAINCODE = STOPERR)
/END-IF
```

Assuming that FHS-PRIV is loaded, the library manager can be called as follows

```
CALL-PROCEDURE FROM-FILE=*LIBRARY-ELEMENT(LIBRARY-MANAGER.PL, RUN)
```

A screen display (using SCREEN01) listing the names of the libraries contained under the user ID should appear. A typical screen is shown below:

```
  File
--------------------------------------------------------------------------------
                      L I B R A R Y     M A N A G E R
--------------------------------------------------------------------------------
  FILE(S) SELECTION             From:      1       Total:      5
                                To  :      5       More :
 ?  Size  Cat. UserId   File name
 ------------------------------------------------------------------------------
   210    2OS2 QM211    ALF.ASS.PLAMLIB
   30     2OS2 QM211    ALF.LIB
   342    2OS2 QM211    LIB-MAN
   342    2OS2 QM211    LIBRARY-MANAGER.PL
   12     2OS2 QM211    SCREEN02
 ====================[ N O   M O R E   D A T A ]==================
 ------------------------------------------------------------------------------
 COMMAND ===>
 F1=HELP  F3=EXIT
--------------------------------------------------------------------------------
LTG                                                              TAST
```

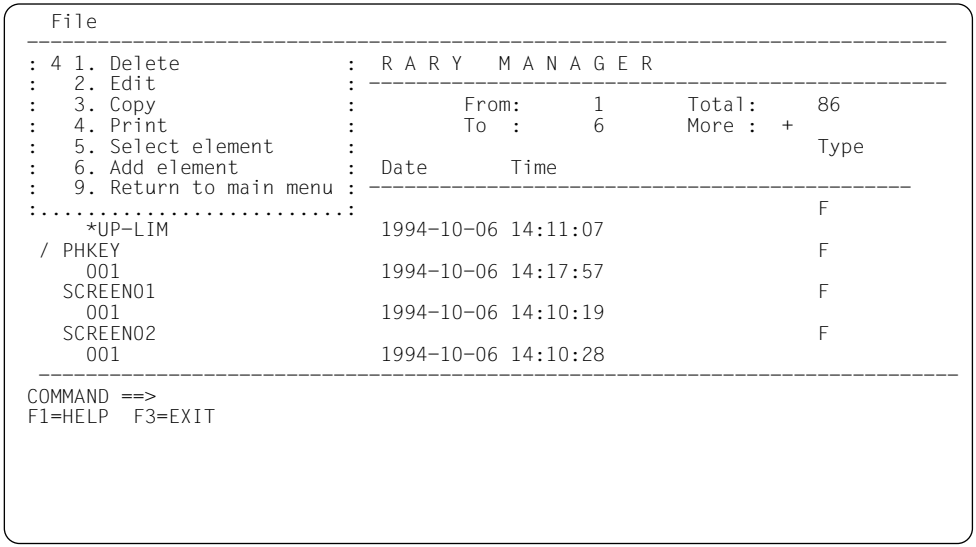
It is now possible to open one of the listed libraries in order to view the members in it. The user marks the library to be opened by entering a “/” (slash) at the start of the corresponding line, then skips to the “FILE” field (top left) with the Tab key, presses the DUE (SEND) key, and enters a “1” in the pull-down menu that appears. The following screen output illustrates the process:

```
  File
------------------------------------------------------------------------------
: 1 1. Open library         :  A R Y     M A N A G E R
:   9. Exit Library-manager : ------------------------------------------------
:.........................:    From:      1       Total:      5
                               To  :      5       More :
 ?  Size  Cat. UserId   File name
 ------------------------------------------------------------------------------
   210    2OS2 QM211    ALF.ASS.PLAMLIB
   30     2OS2 QM211    ALF.LIB
 / 342    2OS2 QM211    LIB-MAN
   342    2OS2 QM211    LIBRARY-MANAGER.PL
   12     2OS2 QM211    SCREEN02
 ====================[ N O   M O R E   D A T A  ]==================
 
 
 
 
 
------------------------------------------------------------------------------
COMMAND ===>
F1=HELP  F3=EXIT
```

If the DUE (SEND) key is pressed again, a further screen (using SCREEN02) containing the names of members of this library (along with the respective date and time of creation, type, etc.) is displayed.

```
  File
------------------------------------------------------------------------------
                        L I B R A R Y   M A N A G E R
------------------------------------------------------------------------------
  ELEMENT(S) SELECTION               From:      1       Total:     86
                                     To  :      6       More :  +
 ? Element                                                        Type
     Version                  Date       Time
 ------------------------------------------------------------------------
   LISTHLP                                                        F
     *UP-LIM                1994-10-06 14:11:07
   PHKEY                                                          F
     001                    1994-10-06 14:17:57
   SCREEN01                                                       F
     001                    1994-10-06 14:10:19
   SCREEN02                                                       F
     001                    1994-10-06 14:10:28
 ------------------------------------------------------------------------------
COMMAND ==>
F1=HELP  F3=EXIT
```

It is now possible to process one of the listed members. This can be done by marking the member to be processed (with a “/” at the start of the line), tabbing to the “FILE” field, pressing the DUE (SEND) key, and entering an appropriate number in the pull-down menu to specify how the member is to be processed (e.g. “4” to print).



The specified action is executed on pressing the DUE (SEND) key.

Here are some further options: the F3 key can be used to exit the library manager and to return to the initial menu. Scrolling is done by entering a “+” or “-” in the “COMMAND” line after the arrow. A Help panel can be displayed by pressing the F1 key.

7 Appendix

7.1 Overview of return codes

The primary value of the return code in DMCOMM is a decimal number with the following structure: gffnnn, where g indicates the error type, ff indicates the function group, and nnn the error number.

Error type	Meaning
0	No error
1	Error/Warning (error flag possible)
2	Error (error flag not possible)
3	Fatal error (e.g. ABORT command)

Function group	Meaning	Fourth character of the message code (x)
00	DM command (CANCEL/EXIT)	
01	General error	Z
02	Syntax check	A
03	DISPLAY service	D
04	Other services	U
05	DMOPEN , ...	B
06	SDF-P interface (FHS-PRIV)	P
07	Variable services	V
08	Reserved	O
09	DOORS	C

The messages are derived from the return code and have the following format: IDHxnnn (x and nnn are explained above).

7.2 List of messages

The messages listed below have the following format:

Message ID (decimal number)
Message text.

IDHA001 (102001)
The application has provided a buffer length greater than allowed. Please contact the developer of the application program

IDHA002 (102002)
A null byte has been detected inside the statement in the operand area or inside a substituted variable. Please contact the developer of your application program.

IDHA003 (102003)
An incorrect variable name syntax has been detected. Please contact the developer of your application program.

IDHA004 (102004)
The statement is larger than allowed after substitution of all variables (maximum 2048 characters). Please contact the developer of your application program.

IDHA005 (102005)
Incorrect statement in operand area. Please contact the developer of your application program.

IDHA006 (102006)
The statement in the operand area is empty. Please contact the developer of your application program.

IDHA007 (102007)
An operand has been specified more than once. Please contact the developer of your application program.

IDHA008 (102008)
A left parenthesis is expected but not found. Please contact the developer of your application program.

IDHA009 (102009)
A right parenthesis is expected but not found. Please contact the developer of your application program.

IDHA010 (102010)

An invalid dialog variable name syntax has been detected. Please contact the developer of your application program.

IDHA011 (102011)

The character “*” is not allowed here as index. Please contact the developer of your application program.

IDHA012 (102012)

The ROW operand has a value out of range [-43..43]. Please contact the developer of your application program.

IDHA013 (102013)

The COLUMN operand has a value out of range [-132..132]. Please contact the developer of your application program.

IDHA014 (102014)

An unknown operand has been detected. Please contact the developer of your application program.

IDHA015 (102015)

No variable name has been detected where at least one is expected. Please contact the developer of your application program.

IDHA016 (102016)

An unexpected left parenthesis has been detected. Please contact the developer of your application program.

IDHA017 (102017)

An unknown value has been detected. Please contact the developer of your application program.

IDHA018 (102018)

Two exclusive values have been detected or one value has been specified twice. Please contact the developer of your application program.

IDHA020 (102020)

The operand DISPLAY has not been detected. Please contact the developer of your application program.

IDHA021 (102021)

The Coded character set name syntax is incorrect. Please contact the developer of your application program.

IDHA022 (102022)

No operand has been detected where at least one was expected. Please contact the developer of your application program.

IDHA023 (102023)

The PANEL name syntax is incorrect. Please contact the developer of your application program.

IDHA024 (102024)

The message number syntax in the MSG operand is incorrect. Please contact the developer of your application program.

IDHA025 (102025)

The operand MSGLOC has been detected but MSG has not been specified. Please contact the developer of your application program.

IDHA026 (102026)

The syntax of the MSGLOC operand is not correct. Please contact the developer of your application program.

IDHA027 (102027)

The syntax of the CURSOR operand is not correct. Please contact the developer of your application program.

IDHA028 (102028)

The operand CSRPOS has been detected but CURSOR has not been specified. Please contact the developer of your application program.

IDHA029 (102029)

The CSRPOS operand is not a valid integer. Please contact the developer of your application program.

IDHA030 (102030)

The CSRPOS operand is out of the range [1..32767]. Please contact the developer of your application program.

IDHA031 (102031)

A profile has been specified but the INIT operand has not been detected yet. Please contact the developer of your application program.

IDHA032 (102032)

The profile name syntax is incorrect. Please contact the developer of your application program.

IDHA033 (102033)

At least one FORMAT value is incorrect. Please contact the developer of your application program.

IDHA034 (102034)

There is more than one FORMAT value but their total does not correspond to the total of variables specified. Please contact the developer of your application program.

IDHA035 (102035)

When the FORMAT operand has the value “*” (filler), the corresponding variable must also have the value “*” (filler). Please contact the developer of your application program.

IDHA036 (102036)

The value for the DIM operand is not an integer. Please contact the developer of your application program.

IDHA037 (102037)

The DIM operand value is not in the range [1..32767]. Please contact the developer of your application program.

IDHA038 (102038)

Extra characters have been found at the end of the statement. Please contact the developer of your application program.

IDHA039 (102039)

An index is not allowed in the variable names for this statement. Please contact the developer of your application program.

IDHA040 (1020400)

The “*” character is not allowed as index in the variables name for this statement. Please contact the developer of your application program.

IDHA041 (102041)

The internal table for memory management is full. Please contact the developer of your application program.

IDHA042 (102042)

The statement in the operand area requires a call via the ISPCI2 entry point. Please contact the developer of your application program.

IDHA043 (102043)

The dialog variable specified for string replacement in the statement in the operand area does not exist. Please contact the developer of your application program.

IDHA044 (102044)

The FORMAT has the value “*” (filler) but you are not currently defining a list. Please contact the developer of your application program.

IDHA045 (102045)
No FORMAT has been specified. Please contact the developer of your application program.

IDHA046 (102046)
An invalid SDF-P variable name syntax has been detected. Please contact the developer of your application program.

IDHA047 (102047)
The application has provided a buffer length that is negative or equal to 0. Please contact the developer of the application program

IDHA048 (102048)
A variable index is not allowed in an SDF-P variable name. Please contact the developer of the application program

IDHA049 (102049)
No variable substitution is allowed in the DMOPEN command. Please contact the developer of the application program

IDHD000 No error.

IDHD004 No error. The command CANCEL was specified.

IDHD008 No error. The command EXIT was specified.

IDHD021 (103021)
The DISPLAY service was specified without the PANEL operand to redisplay the previous screen. However, no previous screen that can be displayed exists.

IDHD022 (103022)
Internal error. Diagnosis: the internal field list is corrupted. Please inform your system administrator.

IDHD023 (103023)
Internal error. Diagnosis: the created screen message contains a position located after the last screen line. Please inform your system administrator.

IDHD024 (103024)
Internal error. Diagnosis: backward positioning in the internal field list. Please inform your system administrator.

IDHD025 (103025)
Internal error. Diagnosis: backward positioning in the created screen message. Please inform your system administrator.

IDHD026 (103026)
: Internal error. Diagnosis: wrong level in stack. Please inform your system administrator.

IDHD027 (103027)
: The format &SYS-PAR0 is defined as a box but no background format presently exists.

IDHD028 (103028)
: A box should be displayed, but the format &SYS-PAR0 is not defined as a box.

IDHD029 (103029)
: Internal error. Diagnosis: all implicit boxes are not closed before returning to the application. Please inform your system administrator.

IDHD030 (103030)
: Error &SYS-PAR7 when displaying a message. The error occurred during error processing.

IDHD031 (103031)
: Error in error processing. Error “&SYS-PAR7” during data preparation.

IDHD032 (103032)
: When calling the DISPLAY service with the operand “LOCK“ or “NODISPLAY”, an action or an FHS-DM message was requested because of checks. MSGID=&SYS-PAR7, PAR1=&SYS-PAR1, PAR2=&SYS-PAR2, PAR3=&SYS-PAR3.

IDHD033 (103033)
: The field name &SYS-PAR1 was specified in a previous ATTR call, but the format &SYS-PAR0 does not contain any field with this name.

IDHD034 (103034)
: Internal error. Diagnosis: wrong input in ATTR control block. Please inform your system administrator.

IDHD035 (103035)
: The internal service &SYS-PAR2 gives a return code &SYS-PAR1.

IDHD036 (103036)
: Some dynamic field attributes should be modified during the display of the format &SYS-PAR0. The attributes for the field &SYS-PAR1 are not allowed. Please inform the application developer.

IDHD037 (103037)
: Return code &SYS-PAR3 when loading the format &SYS-PAR1.

IDHD038 (103038)
Format &SYS-PAR1 cannot be found.

IDHD042 (203042)
Format &SYS-PAR0 should be displayed. Return code “&SYS-PAR1“ returned by the WRTRD call.

IDHD044 (103044)
Warning. Error when loading the character set format &SYS-PAR9. The format does not exist or is not a character set format. The character set is not loaded.

IDHD070 (103070)
Internal return code. The message IDHF186, “Syntax error in SETP command” should be displayed.

IDHD071 (103071)
Internal return code. The message IDHF184, “Syntax error in PANELID command“ should be displayed.

IDHD072 (103072)
Internal return code. The message IDHF185, “Syntax error in KEYAREA command” should be displayed.

IDHD075 (103075)
Internal return code. The message IDHF192, “Error in single choice input field“ should be displayed.

IDHD076 (103076)
Internal return code. The message IDHF094, “Error in multiple choice input field” should be displayed.

IDHD078 (103078)
Internal return code. The message IDHF198, “Index not found” should be displayed.

IDHD079 (103079)
Internal return code. The message IDHF188, “Help not available” should be displayed.

IDHD080 (103080)
Internal return code. The message IDHF191, “Help not found” should be displayed.

IDHD081 (103081)
Internal return code. The message IDHF180, “Wrong command” should be displayed.

IDHD082 (103082)

Internal return code. The message IDHF193, “Wrong cursor position in menu bar” should be displayed.

IDHD083 (103083)

Internal return code. The message IDHF194, “Wrong command in menu bar” should be displayed.

IDHD084 (103084)

Internal return code. The message IDHF187, “Wrong scroll command” should be displayed.

IDHD090 (103090)

Internal error. Diagnosis: the image stack is corrupted. There is no explicit image. Please inform your system administrator.

IDHD099 (103099)

The format &SYS-PAR1 is corrupted. Please create this format again. Diagnosis-reason: &SYS-PAR4 wrong or missing.

IDHD120 (103120)

The version of format &SYS-PAR1 cannot be processed. Please inform the application developer.

IDHD121 (103121)

The format &SYS-PAR1 is corrupted. Inconsistency between the specified and the actual number of lines in the mask. Please create the format again.

IDHD122 (103122)

During the processing of the format &SYS-PAR1, an overflow of the internal table occurred (&SYS-PAR4). A format with fewer fields should be used.

IDHD123 (103123)

The format &SYS-PAR1 is corrupted. Please create the format again. Information for diagnosis: wrong type in MDBE11.

IDHD124 (103124)

The format &SYS-PAR1 is corrupted. Please create the format again. Information for diagnosis: wrong FDB chain for pulldown menu.

IDHD125 (103125)

Warning! The format &SYS-PAR1 contains fields with no name. Entered data is ignored for these fields.

IDHD126 (103126)

In the format &SYS-PAR1 the command input field or a choice input field is a protected field (field name: &SYS-PAR2). Please correct the format.

IDHD127 (103127)

The format &SYS-PAR1 is corrupted. Information for diagnosis: the work area of the mask is too small.

IDHD128 (103128)

The format &SYS-PAR1 is corrupted. Please create the format again. Information for diagnosis: variable field in help format.

IDHD129 (103129)

The format &SYS-PAR0 is corrupted. Diagnosis: for a list, the information “number of line/column” is missing. Please create the format again.

IDHD130 (103130)

The format &SYS-PAR1 is corrupted. Diagnosis: inconsistency between MDBE7 and the MDBE chain. Please create the format again.

IDHD131 (103131)

Internal error. Diagnosis: NULL or NULL-1 as format address in stack. Please inform your system administrator.

IDHD132 (103132)

The format &SYS-PAR0 is corrupted. Please create the format again. Diagnosis: at least one line is longer than the format.

IDHD133 (103133)

An image with a box (or with a pulldown menu) must be created with more than seven character sets. (Format: &SYS-PAR0)

IDHD201 (103201)

The format &SYS-PAR0 is corrupted. Please create the format again. Diagnosis: an index for an edit rule is defined in FDB, but there is no MDBE2.

IDHD202 (103202)

The format &SYS-PAR0 is corrupted. Please create the format again. Diagnosis: requested edit- or check rule is missing in MDBE2.

IDHD203 (103203)

Internal return code for processing error in format &SYS-PAR0. The unexpected value “&SYS-PAR2” was specified in the field &SYS-PAR1. This message is normally replaced by a more explicit one.

IDHD204 (103204)

The format &SYS-PAR0 is wrong. For the field &SYS-PAR1, the comparison value used for the check of value does not satisfy the preparation rule. Value: “&SYS-PAR2”, EDIT-RC=&SYS-PAR3 (see help)!

IDHD205 (103205)

The format &SYS-PAR0 cannot be displayed because the value of the dialog variable “&SYS-PAR1” does not satisfy the requested check. Value: “&SYS-PAR2”, EDIT-RC=&SYS-PAR3 (see help). Please inform the application developer.

IDHD206 (103206)

The format &SYS-PAR0 is corrupted. Please create the format again. Diagnosis: error in MDBE2 for field &SYS-PAR1.

IDHD208 (103208)

The CCS-name &SYS-PAR3 is used in the format &SYS-PAR0 but is unknown in this system.

IDHD209 (103209)

Return code &SYS-PAR3 when calling XHCS for processing format &SYS-PAR0.

IDHD210 (103210)

The format &SYS-PAR0 is corrupted. Please create the format again. Diagnosis: error in MDBE11.

IDHD211 (103211)

XHCS not available for the output processing of format &SYS-PAR0 (RC = &SYS-PAR3).

IDHD212 (103212)

Error &SYS-PAR3 when attempting to write the value “&SYS-PAR2” in the dialog variable &SYS-PAR1.

IDHD213 (103213)

Return code &SYS-PAR3 when reading the dialog variable &SYS-PAR1 in format &SYS-PAR0.

IDHD214 (003214)

Value of a dialog variable truncated for an output field of format &SYS-PAR0.

IDHD215 (003215)

CCS name of format &SYS-PAR0 is ignored.

IDHD217 (103217)

Internal error. Diagnosis: In format &SYS-PAR0, the FDB for the field &SYS-PAR1 contains wrong information. Please inform your system administrator.

IDHD218 (103218)

The specified first line to be displayed in the list does not match the specified &SYS-PAR4. Index of first line: &SYS-PAR5 Index of &SYS-PAR4: &SYS-PAR7

IDHD219 (103219)

Internal error. Diagnosis: wrong internal image for a list (error in list header). Please inform your system administrator.

IDHD220 (103220)

Internal error. Diagnosis: wrong internal image for a list (error when scrolling). Please inform your system administrator.

IDHD221 (103221)

The format &SYS-PAR0 contains a list that was shortened since its last display (last displayed line: &SYS-PAR5, present list end: &SYS-PAR7) (See also help on message)

IDHD222 (103222)

Internal error. Diagnosis: wrong image stack level or deleted format when writing dialog variables. Please inform your system administrator.

IDHD223 (103223)

The control variable &SYS-PAR1 is negative or too large. Format: “&SYS-PAR0”.

IDHD224 (103224)

The format &SYS-PAR0 is wrong. The model line for a list does not contain any variable field. Please correct the format.

IDHD225 (103225)

Internal error. Diagnosis: the internal service IDHHLST was called with wrong parameters. Please inform your system administrator.

IDHD226 (103226)

Internal error. Diagnosis: wrong value for SCROLL lines in CPXCA. Please inform your system administrator.

IDHD227 (103227)

Internal error. Diagnosis: wrong value for SCROLL direction in CPXCA. Please inform your system administrator.

IDHD228 (103228)

Warning. Error when writing the global mark variable of the format “&SYS-PAR0”. The application did not receive information on the marked fields.

IDHD229 (103229)

Warning. The global mark variable of the format “&SYS-PAR0” is not correctly defined. The application did not receive information on the marked fields.

IDHD230 (103230)

Internal error. Diagnosis: wrong variable length (<0> or maximum) when reading the dialogue variable &SYS-PAR1. Please inform your system administrator.

IDHD231 (103231)

Wrong number of elements in dialogue variable array (internal return code = &SYS-PAR3).

IDHD232 (103232)

Warning. Value of dialogue variable &SYS-PAR1 truncated for output.

IDHD233 (103233)

Internal error. Diagnosis: attempt to write after list end.

IDHD234 (103234)

The format &SYS-PAR0 is corrupted. Diagnosis: the format contains FDBs with wrong line or column number. Please correct the format or inform your system administrator.

IDHD241 (103241)

Internal error. Diagnosis: IDHRKEY wrong function. Please inform your system administrator.

IDHD242 (103242)

Internal error. Diagnosis: IDHRKEY no keylist name specified. Please inform your system administrator.

IDHD243 (103243)

Internal error. Diagnosis: IDHRKEY wrong key number. Please inform your system administrator.

IDHD244 (103244)

Internal error. Diagnosis: IDHRKEY wrong key description. Please inform your system administrator.

IDHD246 (103246)

Internal RC. No entry for the key in the key description.

IDHD249 (103249)

The format &SYS-PAR9 was created as a partial format. Partial formats are not supported by FHS-DM. Please correct the format.

IDHD250 (103250)

The standard format &SYS-PAR9 is corrupted. Diagnosis: error because of &SYS-PAR3. Please create the format again.

IDHD251 (103251)

Internal error. Diagnosis: IDHRKEY error in error processing (&SYS-PAR9) Please inform your system administrator.

IDHD252 (103252)

The format &SYS-PAR0 cannot be displayed because the standard format for the keylist IDHKEYA cannot be loaded. Please inform the developer of the application.

IDHD253 (103253)

Internal error. Diagnosis: IDHRKEY- the box type stored in the STE is unknown. Please inform your system administrator.

IDHD271 (103271)

The format &SYS-PAR9 should be displayed. The CCSNAME &SYS-PAR4 is different from the CCSNAME of the screen image (&SYS-PAR5). Please inform the developer of the application.

IDHD272 (103272)

Internal error. Diagnosis: the load service did not return a correct module name. Please inform your system administrator.

IDHD273 (103273)

Internal error. Diagnosis: wrong parameter to load service. Please inform your system administrator.

IDHD275 (103275)

The format &SYS-PAR9 was replaced during the display by a new version (&SYS-PAR4-> &SYS-PAR5).

IDHD276 (103276)

Unknown device group in format &SYS-PAR9. Please create the format again.

IDHD277 (103277)

The format &SYS-PAR9 cannot be displayed on this terminal because it was created with a device group that does not contain this terminal.

IDHD278 (103278)

The format &SYS-PAR9 cannot be processed with the current FHS-DM. Diagnosis: wrong version in MDBE7 Please inform the developer of the application.

IDHD279 (103279)

The format &SYS-PAR9 should be used as a map description format. The format found has the type “&SYS-PAR4” (see help) and does not match the expected type. Please inform the developer of the application.

IDHD280 (103280)

The format &SYS-PAR9 should be used as a map layout description. The format found has the type “&SYS-PAR4” (see help) and does not match the expected type. Please inform the developer of the application.

IDHD281 (103281)

The format &SYS-PAR9 should be used as a map layout (DE, without help). The format found has the type “&SYS-PAR4” (see help) and does not match the expected type. Please inform the developer of the application.

IDHD282 (103282)

The format &SYS-PAR9 should be used as a help format. The format found has the type “&SYS-PAR4” (see help) and does not match the expected type. Please inform the developer of the application.

IDHD283 (103283)

The format &SYS-PAR9 should be used as a keylist. The format found has the type “&SYS-PAR4” (see help) and does not match the expected type. Please inform the developer of the application.

IDHD284 (103284)

The format &SYS-PAR9 should be used as a message member. The format found has the type “&SYS-PAR4” (see help) and does not match the expected type. Please inform the developer of the application.

IDHD285 (103285)

The format &SYS-PAR9 should be used as an ICE character set format. The format found has the type “&SYS-PAR4” (see help) and does not match the expected type. Please inform the developer of the application.

IDHD286 (103286)

The format &SYS-PAR9 should be used as a border description format. The format found has the type “&SYS-PAR4” (see help) and does not match the expected type. Please inform the developer of the application.

IDHD301 (103301)

A call to ADDPOP function with the POPLOC(<name>) operand is made but there is no background image.

IDHD302 (103302)
A field name is specified in operand &SYS-PAR2. The format &SYS-PAR0 does not contain any field with this name.

IDHD303 (103303)
The name &SYS-PAR1 in operand &SYS-PAR2 is not found in format &SYS-PAR0.

IDHD304 (103304)
A box level should be removed with a REMPOP (without ALL operand) but no box exists. Note: maybe the box has been removed because of a previous erroneous call to DISPLAY service.

IDHD305 (103305)
Internal error. Diagnosis: DMOPEN error. Please inform your system administrator.

IDHD306 (103306)
Internal error. Diagnosis: SCTCA not found (SeCTion Communication Area). Please inform your system administrator.

IDHD307 (103307)
Call to ADDPOP function in a wrong sequence (e.g. two ADDPOP in sequence or ADDPOP with POPLOC without previous DISPLAY).

IDHD308 (103308)
Call to REMPOP function in a wrong sequence (e.g. REMPOP after ADDPOP without DISPLAY in between).

IDHD309 (103309)
A call “CONTROL DISPLAY RESTORE” was made, but no saved image exists.

IDHD311 (103311)
Internal error. Diagnosis: error in cursor positioning after list processing. Please inform your system administrator.

IDHD401 (103401)
Warning. Error in message processing. A substitute message was displayed.

IDHD411 (103411)
The position of the box &SYS-PAR1 was specified by the application. The box cannot be displayed at this position.

IDHD412 (103412)
The box &SYS-PAR1 is larger than the screen format.

IDHD413 (103413)
Internal error. Diagnosis: no MSGB found when creating a message box. Please contact your system administrator.

IDHD414 (103414)
Internal error. Diagnosis: internal name table does not exist. Please contact your system administrator.

IDHD415 (103415)
Internal error. Diagnosis: name not found. Please contact your system administrator.

IDHD501 (103501)
Implementation restriction. Attempt to dynamically modify the attributes of too many fields.

IDHD502 (103502)
Implementation restriction. The length of field name whose attributes are to be modified dynamically cannot be stored in the internal name table.

IDHD800 (103800)
RESHOW internal return code.

IDHD801 (103801)
Input from terminal cannot be processed. Repeating the input was not successful. Please contact the application developer.

IDHD901 (103901)
Internal error. Diagnosis: CONTROL service called with a wrong control block. Please contact your system administrator.

IDHD916 (103916)
Internal error. Diagnosis:. Please contact your system administrator.

IDHD917 (103917)
Internal error. Diagnosis: the display service was called with a wrong opcode. Please contact your system administrator.

IDHD918 (103918)
Internal error. Diagnosis: memory not available. Please contact your system administrator.

IDHD919 (103919)
Implemention restriction. Overflow in internal stack. Please inform the application developer.

IDHD920 (103920)
Internal error. Diagnosis:. Please contact your system administrator.

IDHD921 (103921)
Internal error. Diagnosis: memory overflow in &SYS-PAR4. Please contact your system administrator.

IDHD999 (103999)
Error in error processing. Processing stops. Please inform the application developer.

IDHF000 Error in error processing

IDHF001 The message with the message number &SYS-PAR1 should be displayed. The message cannot be found in the used library element. Please inform the application developer. Continue with ENTER.

IDHF002 The message with the message number &SYS-PAR1 should be displayed. The message number is syntactically wrong. Please inform the application developer. Continue with ENTER.

IDHF003 The message with the message number &SYS-PAR1 should be displayed. The library element containing this message cannot be loaded. Please inform the application developer. Continue with ENTER.

IDHF004 The specified TOPINDEX variable (&SYS-PAR5) and the index specified in CURPOS (&SYS_PAR7) are inconsistent. Please inform the application developer. Continue with ENTER.

IDHF005 Error during explicit action in display service. Internal return code: &SYS-PAR6, MSGID: &SYS-PAR5. Please inform the application developer. Continue with ENTER.

IDHF006 Error during implicit action in display service. Internal return code: &SYS-PAR6. Please inform the application developer. Continue with ENTER.

IDHF007 Warning! Return code &SYS-PAR3 when reading the &SYS-PAR1 control variable for the choice field. The error is ignored.

IDHF010 Warning! The format &SYS-PAR0 contains fields that cannot be accessed by the application (field without name). The first field is located in position &SYS-PAR3! More information in help on message.

IDHF011 No message available.

IDHF180 Unknown command was specified in help (or message box.

IDHF182 The SYS command is not allowed in this environment (SDF-P procedure).

IDHF184 The command PANELID contains a syntax error.

IDHF185 The command KEYAREA contains a syntax error.

IDHF186 The command SETP contains a syntax error.

IDHF187 The specified scroll command is not allowed.

IDHF188 No help available.

IDHF191 The help format &SYS-PAR1 does not exist or was not created as a help format.

IDHF192 The specified value is not allowed for a choice. For this reason, no help can be found. Please clear the entry field or specify a correct value before requesting help.

IDHF193 The cursor is located in the menu bar but not in a menu title. Please position the cursor on a menu title or outside the menu bar.

IDHF194 The command that you specified by using a function key is not allowed when the cursor is located in the menu list.

IDHF198 The INDEX format &SYS-PAR1 does not exist.

IDHP001 (106001)
The control variable is not valid. Please contact the developer of the application.

IDHP002 (106002)
The control variable SERVICE is not correct. application.

IDHP003 (106003)
The control variable RESOURCE is not valid. Please contact the developer of the application.

IDHP004 (106004)
The control variable MESSAGE-ID is not valid. Please contact the developer of the application.

IDHP005 (106005)
The control variable MESSAGE-FIELD is not valid. Please contact the developer of the application.

IDHP006 (106006)
The SDF-P name in the control variable MESSAGE-FIELD is not valid. Please contact the developer of the application.

IDHP007 (106007)
The control variable MSG-FIELD-IND is not valid. Please contact the developer of the application.

IDHP008 (106008)
The control variable CURSOR-OUTPUT is not valid. Please contact the developer of the application.

IDHP009 (106009)
The SDF-P name in the control variable CURSOR-OUTPUT is not valid. Please contact the developer of the application.

IDHP010 (106010)
The control variable CURSOR-OUTPUT-INDEX is not valid. Please contact the developer of the application.

IDHP011 (106011)
The control variable CURSOR-OUTPUT-POS is not valid. Please contact the developer of the application.

IDHP012 (106012)
The control variable LOCK is not valid. Please contact the developer of the application.

IDHP013 (106013)
The control variable ALARM is not valid. Please contact the developer of the application.

IDHP014 (106014)
The control variable AUTOTAB is not valid. Please contact the developer of the application.

IDHP015 (106015)
The control variable MANDATORY is not valid. Please contact the developer of the application.

IDHP016 (106016)
The control variable HARDCOPY is not valid. Please contact the developer of the application.

IDHP017 (106017)
The control variable ROW is not valid. Please contact the developer of the application.

IDHP018 (106018)
The control variable COLUMN is not valid. Please contact the developer of the application.

IDHP019 (106019)
The control variable POP-LOCATION is not valid. Please contact the developer of the application.

IDHP020 (106020)
The SDF-P name in the control variable POP-LOCATION is not valid. Please contact the developer of the application.

IDHP021 (106021)
The control variable POP-LOC-IND is not valid. Please contact the developer of the application.

IDHP022 (106022)
The control variable ATTR.FIELD is not valid. Please contact the developer of the application.

IDHP023 (106023)
The control variable ATTR.FIELD-INDEX is not valid. Please contact the developer of the application.

IDHP024 (106024)
The control variable ATTR.TYPE is not valid. Please contact the developer of the application.

IDHP025 (106025)
The control variable ATTR.HILITE is not valid. Please contact the developer of the application.

IDHP026 (106026)
The control variable ATTR.INTENSITY is not valid. Please contact the developer of the application.

IDHP027 (106027)
The control variable ATTR.COLOR is not valid. application.

IDHP028 (106028)
The control variable ATTR.OUTPUT is not valid. Please contact the developer of the application.

IDHP029 (106029)
The control variable REFRESH is not valid. Please contact the developer of the application.

IDHP030 (106030)
The control variable ACK is not valid. Please contact the developer of the application.

IDHP031 (106031)
The FHSTABLE is not valid due to an internal error. Please contact your system administrator.

IDHP032 (106032)
The library in the server info parameter is not correct. Please modify your ASSIGN-COMMAND.

IDHP033 (106033)
The information in the server info parameter is not correct. Please modify your ASSIGN-COMMAND.

IDHP034 (106034)
The control variable ATTR is not a list. Please contact the developer of the application.

IDHT000 No error.

IDHT009 Mandatory input in this field.

IDHT010 This field contains at least one wrong character.

IDHT011 The specified value is lower than allowed. The limit values are: “&SYS-PAR3”

IDHT012 The specified value is greater than allowed. The limit values are: “&SYS-PAR3”

IDHT013 Only digits or a sign as last character are allowed. The sign is not mandatory.

IDHT014 Only the following values are allowed for this field: “&SYS-PAR3”

IDHT015 The following values are not allowed for this field: “&SYS-PAR8”

IDHT017 The specified character string contains at least one unallowed character. Only the following characters are allowed in this field: “&SYS-PAR3”

IDHT018 The specified character string contains at least one unallowed character. The following characters are not allowed in this field: “&SYS-PAR3”

IDHT020 This field does not contain enough relevant characters.

IDHT030 Error in date or time (input too short or wrong separator).

IDHT031 The specified date is outside the allowed range (before 1582-10-15 or after 2099-12-31).

IDHT032 Wrong day in date.

IDHT033 Wrong month in date.

IDHT034 Wrong year in date.

IDHT035 The separator in date or time is wrong or missing. Expected separator: “&SYS-PAR3”

IDHT037 Wrong hour in time.

IDHT038 Wrong minute in time.

IDHT039 Wrong second in time.

IDHT040 The specified value is too large. Please ensure that space for a decimal separator, for a thousand separator or for a sign is reserved in the field. Defined model: “&SYS-PAR3”

IDHT050 Too many decimal positions in arithmetical field. Defined model: “&SYS-PAR3”

IDHT060 This arithmetical field cannot contain any sign.

IDHT061 More than one sign entered in an arithmetical field.

IDHT063 Only signs have been entered in an arithmetical field.

IDHT070 This arithmetical field cannot contain any separator. Defined model: “&SYS-PAR3”

IDHT071 Error in digit grouping in an arithmetical field.

IDHT080 No decimal separator can be entered in this arithmetical field. Defined model: “&SYS-PAR3”

IDHT090 Error in arithmetical field. Defined model: “&SYS-PAR3”

IDHT092 The requested choice is presently locked.

IDHT093 The entered value is not allowed for a choice.

IDHT094 Wrong character specified for a choice.. Correct character for choice: “&SYS-PAR3”

IDHU001 (204001)
The DMCOMM is invalid. Please recall the function with a valid DMCOMM.

IDHU002 (204002)
: FHS doesn´t run with a VTSU-B version lower than 11.0. Please contact the system administrator to install this version.

IDHU003 (204003)
: FHS doesn´t run on a BS2000 version lower than 10.0. Please use another version of BS2000.

IDHU004 (204004)
: A problem occurs during initialisation of the run time system. Please contact the system administrator.

IDHU005 (204005)
: The number of DMOPEN is too high. Only 29 DMOPEN are possible. Please modify your program.

IDHU006 (204006)
: The address of the parameter list is not aligned on a full word. Please modify your program.

IDHU007 (204007)
: The address of the length operand is not aligned on a full word. Please modify your program.

IDHU008 (204008)
: The address of the value length operand is not aligned on a full word. Please modify modify your program.

IDHU009 (004009)
: FHS-DOORS is not available. There is no display. Please add the FHS-DOORS module. (IDHIO).

IDHU010 (204010)
: The DMCOMM is already closed. Please recall the function with a valid DMCOMM.

IDHV001 (007001)
: The profile element &SYS-PAR17 cannot be found in the library with IDHPROF link name. An empty profile will be created.

IDHV002 (107002)
: The profile element &SYS-PAR17 cannot be read from the library with IDHPROF link name. Check the consistency of the profile in the library.

IDHV003 (107003)
: The profile element &SYS-PAR17 cannot be written in the library with IDHPROF link name. Check if the library and the element can be updated.

IDHV011 (107011)
Internal error. ILAM initialization error &SYS-PAR12. Please contact your system administration or maintenance.

IDHV012 (107012)
The library with the IDHPROF link name cannot be found. Check the link name and the library.

IDHV031 (107031)
The variable &SYS-PAR13 is already defined as an explicit variable. It cannot be defined again. Please contact the developer of the application program.

IDHV032 (107032)
The variable &SYS-PAR13 has an invalid length for its format. Please contact the developer of the application program.

IDHV101 (107101)
Conversion error for the variable &SYS-PAR13. Check the variable definition if possible. System diagnosis: internal conversion error from the type &SYS-PAR14 into the type &SYS-PAR15, value &sys-par16 of length &sys-par17 to length &sys-par18.

IDHV102 (107102)
The variable &SYS-PAR13 is an implicit variable. It cannot be an array. Please contact the developer of the application program.

IDHV103 (107103)
The explicit variable &SYS-PAR13 is not an array. It cannot be read with the specified index. Please contact the developer of the application program.

IDHV104 (107104) Internal error. Error during VAS processing. Vas return code &SYS-PAR12 on calling the VAS &SYS-PAR11 function. Please contact your system administration or maintenance.

IDHV106 (107106)
The system variable &SYS-PAR13 cannot be updated.

IDHV107 (107107)
Internal error. Invalid pool specification for variable access. Please contact your system administration or maintenance.

IDHV111 (107111)
Read error for the variable &SYS-PAR13 which is read with an invalid VAS user type. Please contact your system administration or maintenance.

IDHV112 (107112)

Read error for the variable &SYS-PAR13 which is read in an invalid VAS variable type. Please contact your system administration or maintenance.

IDHV113 (107113)

The function cannot be processed as the variable &SYS-PAR13 does not exist. Please contact the developer of the application program.

IDHV114 (107114)

The function cannot be processed as the S variable &SYS-PAR13 does not exist. Please contact the developer of the application program.

IDHV115 (107115)

Read error for the S variable &SYS-PAR13 which is read in an unsupported S variable type (not string or integer). Please contact the developer of the application program.

IDHV116 (107116)

Internal error. Invalid container. Only PROFILE, TASK or PROCEDURE are supported. Please contact your system administration or maintenance.

IDHV117 (007117)

The dialog variable &SYS-PAR13 does not exist. Please contact the developer of the application program.

IDHV118 (007118)

The S variable &SYS-PAR13 does not exist. Please contact the developer of the application program.

IDHV120 (007120)

The variable &SYS-PAR13 has a dimension of &SYS-PAR16. Access is not possible with an index greater than this value. Please contact the developer of the application program.

IDHV121 (007121)

Truncation during conversion of the variable &SYS-PAR13 from the &SYS-PAR14 type into the &SYS-PAR15 type, value &SYS-PAR16 of length &SYS-PAR17 to length &SYS-PAR18. Please contact the developer of your application program.

IDHV126 (107126)

The S variable &SYS-PAR13 cannot be put. PUTVAR error &SYS-PAR15. Please contact the developer of the application program.

IDHV127 (107127)

The S variable &SYS-PAR13 cannot be got. GETVAR error &SYS-PAR15. Please contact the developer of the application program.

IDHV131 (107131)

The variable &SYS-PAR13 can be displayed in a list only if the variable is already generated as a list by SDF-P. Please contact the developer of the application.

IDHV132 (107132)

The variable &SYS-PAR13 can be displayed as a simple variable only if the variable is a simple S variable string or integer. Please contact the developer of the application.

IDHV133 (107133)

Internal error. The variable &SYS-PAR13 is read with a variable type that is not supported. Please contact your system administration or maintenance.

IDHV134 (107134)

The extension of a list of S variables is not supported by this version of FHS. The concerned variable is &SYS-PAR13. Please contact the developer of the application.

IDHZ001 (101001)

The format &SYS-PAR18 cannot be loaded. Check the format in the library hierarchy. (Return code &SYS-PAR15 from BIND or $PBBND1)

IDHZ002 (001002)

The format &SYS-PAR18 cannot be unloaded. Check the format in the library hierarchy. (Return code &SYS-PAR15 from UNBIND or $PBUNL1)

IDHZ003 (001003)

Internal warning. The format &SYS-PAR18 cannot be unloaded. It cannot be found in the internal table.

IDHZ004 (101004)

Internal error. Invalid action requested from the loader service.

IDHZ005 (101005)

You have reached the maximum number of BLS contexts used simultaneously as one BLS context is associated with each FHS complex (“DMOPEN INIT”). Please contact the developer of the application program.

IDHZ006 (101006)

Internal error. Error detected by the loader service. Return code &SYS-PAR15 from ($)VSVI1 for the format &SYS-PAR18.

IDHZ007 (101007)

The Format &SYS-PAR18 has been loaded correctly, but in the structure of the format an error has been detected. Please check the format.

IDHZ011 (201011)

Additional memory cannot be requested. Return code &SYS-PAR18 from system call &SYS-PAR15. You have already requested all your available memory space. Check your addressing space size with your system administrator.

IDHZ012 (201012)

Additional memory cannot be requested. Return code &SYS-PAR18 from system call &SYS-PAR15.

IDHZ013 (101013)

Internal error. No such memory has been requested with this id. Consequently, it cannot be released.

IDHZ014 (101014)

Internal error. The memory manager has been called for an unknown action.

IDHZ015 (101015)

Internal error. Error by error processing of memory manager. Return code &SYS-PAR18 from system call &SYS-PAR15.

IDHZ016 (001016)

Error during release memory processing.

7.3 Overview: system variables of the dialog manager

All system variables are in string format. They are contained in different pools. The following codes are used in the tables below:

F - Function pool
P - Profile pool
S - System pool (read-only).
C - Current value determined
I - Internal diagnostic information

General system variables

Name	Pool	Type	Length	Meaning
SYS-DATE	C	CHAR	12	Current date in the ISO format YY-MM-DDiiib 1)
SYS-STDDATE	C	CHAR	14	Current date in the ISO4 format YYYY-MM-DDiiib 1)
SYS-TIME	C	CHAR	5	Current time in 24-hour format: HH:MM 1)
SYS-STDTIME	C	CHAR	8	Current time in the 24 hour format with seconds: HH:MM:SS (1)
SYS-DAY	C	CHAR	2	Day of the month (2 digits)
SYS-MONTH	C	CHAR	2	Month (2 digits)
SYS-YEAR	C	CHAR	2	Year (2 digits)
SYS-STDYEAR	C	CHAR	4	Year (2 digits)

1) Key to the characters used in the formats described above:

Y Year
M Month (in date specifications)
D Day
H Hours
M Minutes (in time specifications)
S Seconds
iii Current day of year
b Blank

System variables of the dialog manager

Name	Pool	Type	Length	Meaning
SYS-COMMAND	F	CHAR	255	Application command that can be processed by the application program.
SYS-FHS-VER-SION	S	CHAR	6	Version of the dialog managers
SYS-P-KEYS-SETTING	P	CHAR	24	Define P-KEY assignments. The value can be changed using the system command SETP
SYS-KEY-AREA	P	CHAR	1	Display mode for key assignments - 0: Key assignments not shown (Default) - 1: Key assignments shown. The value can be changed by using the system command KEYAREA
SYS-PANEL-ID	P	CHAR	1	Display mode for format name and message ID - 0: Format name and message ID not shown - 1: Format name and message key shown The value can be changed by using the system command PANELID

The profile variables are searched in the profile pool on a DMOPEN with INIT. If they are not present, a default is generated. The variables are written back on a DMCLOSE for the dialog complex. The content of these variables can only be changed by means of DM system commands.

The system variables in the following tables will have the specified values only on exiting the displayed format.

Name	Pool	Type	Length	Meaning
SYS-CURSOR-FIELD	F	CHAR	255	Name of the mask field containing the cursor or '$lll#ccc', if the cursor is on an unnamed field or in a space between fields
SYS-CURSOR-INDEX	F	FIXEDS	2/4	Index of the list line of a displayed list containing the cursor. This value only applies if SYS-CURSOR-FIELD contains the name of a list field
SYS-CURSOR-POS	F	FIXEDS	2/4	Position of the cursor in the field named in SYS-CURSOR-FIELD
SYS-CCS-NAME	F	CHAR	8	CCSNAME that was used by the DISPLAY service to display the last format

Diagnostic variables of the dialog manager

Name	Pool	Type	Length	Meaning
SYS-PARxx	I	CHAR	<80	xx = 0 - 20 0 - 9: for diagnostics in display services 10 - 20: diagnostics in variable services internal diagnostic information that is used in messages
SYS-PAR0	I	CHAR	8	Name of the current explicit format
SYS-PAR9	I	CHAR	8	Name of the active format

The following information is stored in info variables when fields are checked. These entries can be accessed in private messages, which can be defined using IFG.

SYS-PAR1 Name of the current field

SYS-PAR2 Content of the current field

SYS-PAR3 Additional information on edit errors: EDIT error number
Check for CCS name: CCS name

The names, ZPAR1, ZPAR2 and ZPAR3 have been mapped to SYS-PAR1, SYS-PAR2 and SYS-PAR3 for compatibility with FHS-UTM. These names should no longer be used.

Glossary

ADDPOP

Generation of dialog boxes.

attribute

Characteristic relating to the display, editing or checking of a format or field. An attribute is defined either during format generation using IFG (static attribute) or in the application program by way of the global and field attributes (dynamic attributes).

basic format

Format output by the application and that can be overlaid by dialog boxes.

box

Abbreviation of dialog box; see relevant entry.

character set file

File containing the character sets generated using ICE.

control block

Memory area used to store formatting parameters and acknowledgments.

DE format

Format that can use the functions of the dialog extension. The attribute "DE format", must be explicitly specified during generation with IFG.

dialog box

Square frame on the screen which contains a DE format.

dialog extension

Component of FHS which allows formats conforming to the Alpha Style Guide to be displayed on the screen. Dialog extension enables multilevel intermediate dialog, command input, extended input checks, and an application-specific help system and messages, amongst other things.

dialog variable

Dialog variables are used in data exchange between an application program, FHS-DM and dialog elements. Dialog variables perform the function of the data transfer area.

differential output

Output of a format in which only those fields that have been changed by the application program are output afresh on the data display terminal.

exclusion character

Character on the screen which indicates a locked selection of a selection field.

explicit box

Dialog box that is output by the application.

FHS-DM

Format Handling System - Dialog Manager

format

Logical data structure that describes a "form".

format application fiel

Library used to store the format definitions.

global help

Help for the objects of a DE format that consist of several components, such as single-choice or multiple-choice selection fields.

help box

Dialog box containing help information; output by FHS.

implicit box

Dialog box output by FHS, e.g. for messages or help information.

input field

Field in which data for the application program is entered by the terminal operator.

KEY format

Format containing the assignment of function keys.

modal box

Dialog box that expects an entry from the user. The underlying box or format is inactive.

modeless box

Dialog box that does not expect direct user action. The underlying box or format remains active.

output field

Field into which data is output by the application program.

REMPOP

Removes boxes; the previous background is displayed again.

screen restart

Fresh output of the most recent, completely formatted screen after an interruption.

S-variable

Variable for SDF-P.

text field

Field containing fixed text that is defined during format generation.

Related publications

FHS - Format Handling System for UTM, TIAM, DCAM (TRANSDATA)

User Guide

IFG for FHS (TRANSDATA)

User Guide

ICE (TRANSDATA) Interactive Character Editor
User Guide

FHS-DOORS (SINIX, BS2000)

User Guide

SDF-P V2.0 (BS2000/OSD)

User Guide

RPG3 (BS2000)

RPG Compiler

User Guide

TIAM (TRANSDATA, BS2000)

User Guide

BS2000/OSD-BC V1.0

Executive Macros

User Guide

TRANSVIEW-NMA (PDN)

TRANSVIEW-NMAE (PDN)

Network Management and Measurement Data Compilation in PDN (TRANSDATA, PDN)

Commands

User Guide

TRANSVIEW-NMA (PDN)

TRANSVIEW-NMAE (PDN)

Network Management and Measurement Data Compilation in PDN (TRANSDATA, PDN)

Functions and Facilities

User Guide

Network Management in BS2000 (TRANSDATA)

User Guide

BS2000/OSD-BC V1.0

User Commands (SDF Format)

User Guide

COBOL85 (BS2000)

COBOL Compiler

User Guide

TRANSIT-CD (TRANSDATA)

User Guide

9749, 9750, 9752 Data Display Terminals (TRANSDATA)

Interface for Programmers

User Guide

BS2000 Data Communication System

Technical Description

Alpha Style Guide

Guidelines for the Design of Character-oriented User Interfaces

User Guide

Ordering manuals

The manuals listed above and the corresponding order numbers can be found in the Siemens Nixdorf *List of Publications*, in which the ordering procedure is also explained. New publications are described in the *Druckschriften-Neuerscheinungen (New Publications)*.

You can arrange to have both of these sent to you regularly by having your name placed on the appropriate mailing list. Please apply to your local office, where you can also order the manuals.

Index

D

Contents

FHS V8.1A (BS2000/OSD, TRANSDATA)

Dialog Extension for TIAM and SDF-P

User Guide

Target group
Application developers
Contents
The manual describes the program interface for using the FHS dialog manager in TIAM and SDF-P applications.

Edition: December 1994

File: FHS_DM.PDF

Information on this document

On April 1, 2009, Fujitsu became the sole owner of Fujitsu Siemens Computers. This new subsidiary of Fujitsu has been renamed Fujitsu Technology Solutions.

This document from the document archive refers to a product version which was released a considerable time ago or which is no longer marketed.

Please note that all company references and copyrights in this document have been legally transferred to Fujitsu Technology Solutions.

Contact and support addresses will now be offered by Fujitsu Technology Solutions and have the format *...@ts.fujitsu.com*.

The Internet pages of Fujitsu Technology Solutions are available at *http://ts.fujitsu.com/*...
and the user documentation at *http://manuals.ts.fujitsu.com*.

Hinweise zum vorliegenden Dokument

Zum 1. April 2009 ist Fujitsu Siemens Computers in den alleinigen Besitz von Fujitsu übergegangen. Diese neue Tochtergesellschaft von Fujitsu trägt seitdem den Namen Fujitsu Technology Solutions.

Das vorliegende Dokument aus dem Dokumentenarchiv bezieht sich auf eine bereits vor längerer Zeit freigegebene oder nicht mehr im Vertrieb befindliche Produktversion.

Bitte beachten Sie, dass alle Firmenbezüge und Copyrights im vorliegenden Dokument rechtlich auf Fujitsu Technology Solutions übergegangen sind.

Kontakt- und Supportadressen werden nun von Fujitsu Technology Solutions angeboten und haben die Form *...@ts.fujitsu.com*.

Die Internetseiten von Fujitsu Technology Solutions finden Sie unter *http://de.ts.fujitsu.com/*..., und unter *http://manuals.ts.fujitsu.com* finden Sie die Benutzerdokumentation.