1 Preface

Increasingly, mainframe computers are being used together with PCs or multiuser systems in heterogeneous environments. This configuration combines the fast processing speeds of central mainframes with the individual nature of personal computers, which are nowadays generally equipped with user-friendly graphical interfaces to facilitate working with applications.

1.1 Characteristic features of FHS-DOORS

FHS-DOORS is the graphical interface for all BS2000/OSD applications. A simple mouse click often replaces the tedious process of entering text. The number of input errors is reduced and the application is far easier to use. All applications have a uniform interface, so that users can concentrate on their work rather than on the mechanics of using the application.

FHS-DOORS supports all BS2000/OSD form systems, so existing applications need not be rewritten. The structure of the screen forms is automatically converted to match the user interface. Another option is to convert the forms to panels with the form converter and then to optimize these panels using the DOORS editor.

FHS-DOORS runs under Microsoft Windows V3.1, Windows 95 and Windows NT.

In order to work with FHS-DOORS, you will need to have a DDE-compatible emulation for BS2000 installed on your PC. This emulation is the communications basis via which forms are displayed as graphically converted panels on a PC. All communication protocols supported by the emulation are also supported by WIN-DOORS. The DOORS emulation is included in the delivery package.

1.2 Structure and contents of the FHS-DOORS documentation

FHS-DOORS is aimed at BS2000 users who want to take advantage of the benefits provided by graphical interfaces for BS2000 applications.

The documentation for FHS-DOORS comprises the following components:

– the present manual, “FHS-DOORS, Graphical Interface for BS2000/OSD Applications”

– the manual “DOORS Emulation, 9750 Basic Emulation for BS2000/OSD Connections”

– the manual “WIN-DOORS/FHS-DOORS, Optimizing Forms Using the DOORS Editor”

– online help for FHS-DOORS

– online help for the DOORS emulation

– online help for the DOORS editor

To work with an application under FHS-DOORS, you require basic knowledge of how to operate graphical interfaces. You will find an introduction to the Microsoft Windows graphical user interface and a description of how to work with it in the Microsoft Windows documentation.

FHS-DOORS, Graphical Interface for BS2000/OSD Applications

This manual contains details on the various FHS-DOORS usage models and describes the functionality provided by FHS-DOORS. It covers how to use FHS-DOORS, OLE automation, management of resource files, and installation and configuration of FHS-DOORS.

If you are using FHS-DOORS for the first time, you should first read chapter “FHS-DOORS - graphical interface for BS2000/OSD applications” on page 7ff and chapter “Using FHS-DOORS” on page 25ff.

This FHS-DOORS manual also includes descriptions of the FHS-DOORS-LC form converter (BS2000/OSD) and the graphical interface for the Event Stream Service ESS-DOORS.

This manual is intended for end users who work with host applications under FHS-DOORS and administrators who are responsible for converting forms with the FHS-DOORS-LC (BS2000/OSD) form converter and making the converted forms (panels) available on the PCs.

A detailed description of the FHS-DOORS interface can be found in the online help for FHS-DOORS.

DOORS Emulation, 9750 Basic Emulation for BS2000/OSD Connections

The DOORS emulation manual describes the functions of the DOORS emulation. It tells you how to work with the DOORS emulation, how to set up connections (connection names), how to use startup scripts to automate the process of establishing a connection and how to install and configure the DOORS emulation.

It is intended for users who want to use the DOORS emulation for connecting to a BS2000/OSD partner.

A detailed description of the DOORS emulation interface can be found in the online help for the DOORS emulation.

FHS-DOORS also works with other 9750 emulations which support the DOORS-DDE protocol (e.g. SNI's MT9750 emulation). These can be used in place of the DOORS emulation.

WIN-DOORS/FHS-DOORS, Optimizing Panels Using the DOORS Editor

The "DOORS Editor" manual describes the options available for optimizing converted forms (panels) using the DOORS editor. It tells you how to work with the DOORS editor, explains the user-specific extensions under the Dialog Builder, and describes how to install and configure the DOORS editor.

The manual is aimed solely at those responsible for optimizing panels.

A detailed description of the DOORS editor interface can be found in the online help for the DOORS editor.

README files

Details of any functional changes and additions to this manual and the online help system are included in README files named *readme.txt*. The file is located in the installation directory for the relevant products. You can view the files in an editor or print them out on a standard printer.

1.3 Changes since FHS-DOORS V3.0

- **Capture**
  The Capture mechanism allows you to define an identifier for each form. This enables FHS-DOORS to uniquely identify the form. The Capture mechanism can also be used to create subpanels for a form. If a form template is available for a form, FHS-DOORS will use it to display the panel.
- **Alphanumeric object**
  When using forms with several I/O fields, performance can be improved by combining multiple forms for a panel into a single alphanumeric object. This does not change how fields are displayed in the panel or processed by the user; however, less system resources are required to display the alphanumeric object.
- **Interfaces**
  Dynamic forms library
  A customized algorithm for forms recognition can be made available to FHS-DOORS by means of a dynamic library (DLL) for forms recognition. A skeleton program to write this DLL is supplied with the product. This program includes predefined functions that can evaluate the data of a form and process it as specified.
- **ESS-DOORS - Event Stream Service**
  ESS-DOORS provides you with a graphical interface under FHS-DOORS for the Event Stream Service (ESS) in BS2000/OSD (as of BC V3.0), and thus for BS2000 system operations as well. ESS-DOORS is intended for system operators and BS2000 users who want the convenience of using a graphical interface even with the Event Stream Service.

1.4 Notational conventions

The following conventions are used in this manual:

In syntax:

bold or proportional characters	Constants: these characters must be entered exactly as shown.
regular characters	Variables: these characters stand for other characters that you select and enter.
␣	Mandatory blank.
[ ]	Optional: arguments in square brackets are optional. The brackets themselves must not be specified.
{ | }	Braces enclose alternatives separated by the logical OR character. The braces themselves and the OR character must not be specified.

In the body of the text:

“double quotes”	Names of chapters/sections and terms requiring special emphasis.
italics	File names and terms which appear on the screen (such as functions, menus, field names or pushbuttons).
[number]	Reference to a manual listed in the “Related publications” section.
►	An action that you have to take.
Key	Indicates a key on the keyboard.
i	A note providing you with additional information.

One of the characteristic features of a graphical interface is that commands can be executed directly by selecting them from menus instead of typing them on a command line. In order to simplify the process of finding a particular command, menu titles are shown in this manual along with the menu item, which is separated by a slash. For example, *Edit/Copy* means that the *Copy* function is to be selected from the *Edit* menu. An arrow (→) in a command indicates a cascade menu.

You will find a description of the SDF syntax in section “FHS-DOORS-LC (BS2000/OSD) statements” on page 86ff.

1.5 Terms used

The terms *form* and *panel* have special meanings in this manual. In the past, the terms “mask” and “graphical mask” or “graphical form” have been used for these two terms.

A *form* is an alphanumeric form from a host application (e.g. an FHS form or a FORMPLAG form etc.).

A *panel* is a graphical form on the PC and is stored in a semantics file (sdc file). The sdc file is generated from a form with FHS-DOORS or with the form converter (BS2000/OSD).

The term *resource path* is used synonymously with *resource directory*.

2 FHS-DOORS - graphical interface for BS2000/ OSD applications

FHS-DOORS allows BS2000 to support graphics. This applies to all BS2000/OSD applications.

FHS-DOORS allows you to make use of the advantages of a graphical interface without having to rewrite your existing host applications. This means, for instance, that UTM applications will run without restrictions with FHS as of V8.0 without the need to edit your screen forms. The structure of the forms is automatically converted to graphical windows (panels).

It is possible to migrate to FHS-DOORS step by step. It is thus possible to run PCs with graphical interfaces in parallel with alphanumeric terminals.

Forms are converted to the graphical interface automatically using default values (e.g. background colors, fonts and borders). The DOORS editor then allows you to optimize the converted forms (panels). Important items can be highlighted in color and related items can be clearly grouped. Customization allows the graphical interface to be tailored exactly to your own requirements.

FHS-DOORS features:

- **A graphical interface for all BS2000/OSD form systems**
  There are no special requirements with regard to the form system used
- **System stability**
  No changes to the existing application are required
- **Automatic conversion of forms**
  No programming is required
- **Templates**
  Automatic conversion of forms using freely-definable, user-specific templates
- **Macros**
  Automation of work processes
- **Ergonomics, efficiency and individuality**
  Optimization of the panels using the DOORS editor

- **Graphics where they make sense**
  Mixed use of graphical and alphanumeric screens
- **OLE automation**
  Control of work procedures using, for instance, a Visual Basic script or a C++ program
- **Support for FHS V8.0 and V8.1 functions**
  Single-selection fields, multiple-selection fields, key forms, check options for fields, online help, subforms, menus
- **Graphical interface for OSD operators**
  Support of the ESS-DOORS application of the graphical interface of Event Stream Services as of OSD Version 3.

2.1 The new FHS-DOORS model

The new FHS-DOORS model is no longer based on a particular BS2000/OSD form system. This means that FHS-DOORS can be used with any BS2000/OSD form system. No specific software or versions are necessary in BS2000 to be able to use FHS-DOORS.

FHS-DOORS can be used with a connection to any UTM, TIAM or DCAM application or task.

When you establish a connection to a host application with FHS-DOORS, the system first checks whether the host application operates in full-screen mode (i.e. forms) or line mode. If it uses full-screen mode, the forms are displayed in the FHS-DOORS application window. If it uses line mode, output is sent to the emulation window of the emulation used to establish the connection or to the FHS-DOORS application window.

Automatic conversion of forms using templates

The forms are automatically converted to panels on the PC at runtime. Templates mean that this process can be controlled using freely-definable, user-specific templates. This means that all the forms can be linked to a "common" template, or each form can be linked to an "individual" template.

The templates are created directly with FHS-DOORS and saved in a file. Generation of the templates requires no programming since it is carried out in simple dialogs. Templates can be made available to all users once they have been saved.

Gradual optimization

After the panels have been converted automatically, each of them can be further optimized. This optimization process can be carried out at runtime (online) or offline. FHS-DOORS includes an object-oriented editor (DOORS editor) for this optimization process. The optimized panels are stored in files (sdc files) and can then be made available to all users. Optimization requires no programming since it is carried out in simple dialogs

Filling in fields automatically using macros

Macros allow you to automate individual steps, e.g. by automatically filling in certain fields and sending the panel to the host application. Macros thus relieve users of tiresome, repetitive tasks. FHS-DOORS provides an easy way of recording macros.

Macros can be started using shortcuts or menus or automatically as soon as the panel is displayed.

Controlling execution with OLE automation

Instead of controlling the host application from panels, entire work processes can be automated with a Visual Basic script or a C++ program using the OLE objects provided by FHS-DOORS.

It is thus possible, for instance, to collect data from various forms, from different host applications and from different hosts and show this data in a single Excel chart. The various forms called are no longer displayed. Instead of the underlying host application, the user then only sees the interface of the Visual Basic script or the C++ programs which control a particular process.

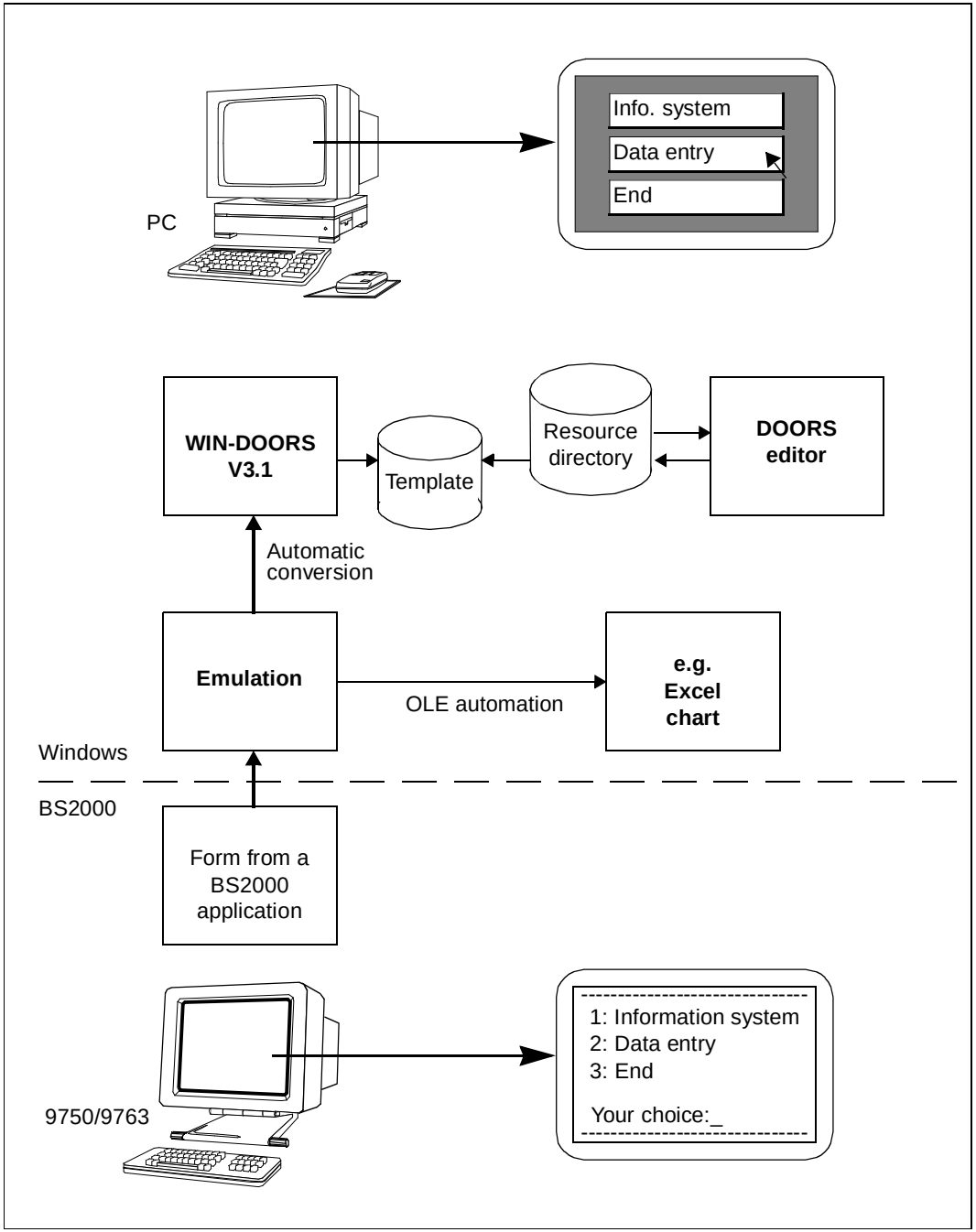


Figure 1: New FHS-DOORS model

2.2 The extended FHS-DOORS model

FHS-DOORS uses the FHS V8.x dialog extension (FHS-DE) to provide a simple graphical interface on the PC. This involves transferring functions such as the checking of input fields or the help system to the PC wherever possible.

When FHS-DE is used, the form converter in BS2000 generates user-friendly panels for FHS-DOORS on the basis of the information contained in the forms library. The FHS fields are converted to graphical objects: single-selection fields are replaced by option buttons, and multiple-selection fields by check boxes. The size and position of the objects are defined by the row/column location of the fields in the FHS form. KEY forms are replaced by pushbuttons at the bottom of the panel or by menus.

New or extended FHS-DOORS model?

This section is intended to help you decide whether you should use the “new” or “extended” FHS-DOORS model.

The extended FHS-DOORS functions can be recommended:

– if you work with FHS applications or FORMPLAG applications

– and if FHS as of V7.1 or FORMPLAG as of V2.4C is installed on BS2000 and if UTM as of V3.2 is installed.

The advantages are as follows:

if FHS as of V7.1 is used

– the form converter (BS2000/OSD) can be used to generate the panels (sdc files) from the BS2000 forms

– it is not necessary to select an identifier for recognizing forms (capture)

– subforms can be specially optimized

and if FHS as of V8.0 and UTM as of V3.3 are also used

– panels can be distributed automatically (downloading)

– FHS dialog extensions (FHS-DE) are supported directly.

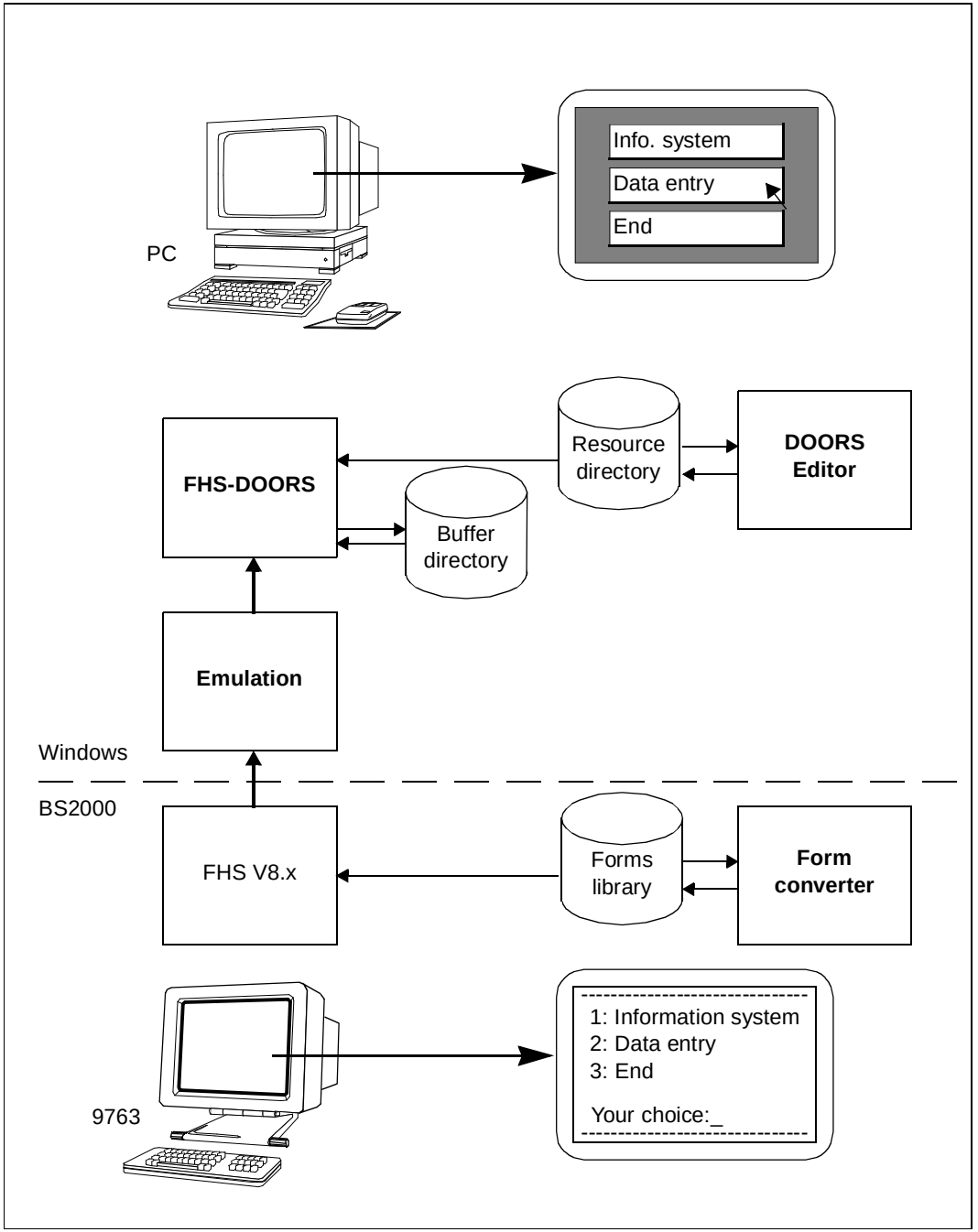


Figure 2: Extended FHS-DOORS model

Requirements for the extended model

Before you can use the extended FHS-DOORS model, you must first check whether your host application is suitable. The following requirements must be met:

– The forms for the host application must have been created with the Interactive Form Generator (IFG).

– The application must pass the correct terminal type (DSS-FE) to the Format Handling System (FHS). This is the case, for instance, with all UTM applications which have communication partners defined in the KDCFILE terminals of type 9763 (see section “Working in BS2000” on page 115ff).

– If a program queries the terminal attributes with the TSTAT macro, and if the terminal type returned is passed to FHS, the program can be used with the extended model without changes. Programs which do not use TSTAT must first be adapted (see section “Working in BS2000” on page 115ff).

Support is also provided for FORMPLAG forms as of FORMPLAG V2.4C. For more information on the support provided for FORMPLAG forms, refer to the current release notice for the relevant FORMPLAG version.

Various procedures in the extended model

The procedures used depend on whether

– the panels for FHS-DOORS are to be generated automatically from the forms at runtime

– the panels are to be distributed automatically (downloading) or manually

– the panels are to be optimized with the DOORS editor.

The following table shows you the steps you need to take to utilize a specific procedure:

	Steps required			
Use of ...	**Convert forms**	**Optimize panels**	**Add panels to forms library**	**Transfer panels to resource directory**
Automatic conversion of forms	—	—	—	—
Automatic distribution (downloading) of panels	yes	—	(automatic)	—
Automatic distribution (downloading) of optimized panels	yes	yes	yes	—
Manual distribution of panels	yes	—	—	yes
Manual distribution of optimized panels	yes	yes	—	yes

Automatic conversion of forms

If FHS-DOORS cannot access the host application resources because they are errored or simply unavailable to FHS-DOORS, the form description is generated from the FHS form automatically in FHS-DOORS. Automatic conversion of this type saves storage space. This means that it makes sense to use the form converter under BS2000 to convert to panels only those forms of the host application which then also need to be optimized or to delete any converted panels which do not require optimization.

The automatic conversion does not generate any option buttons, check boxes or pushbuttons, and validity checking for entry fields and the help system are not transferred to the PC.

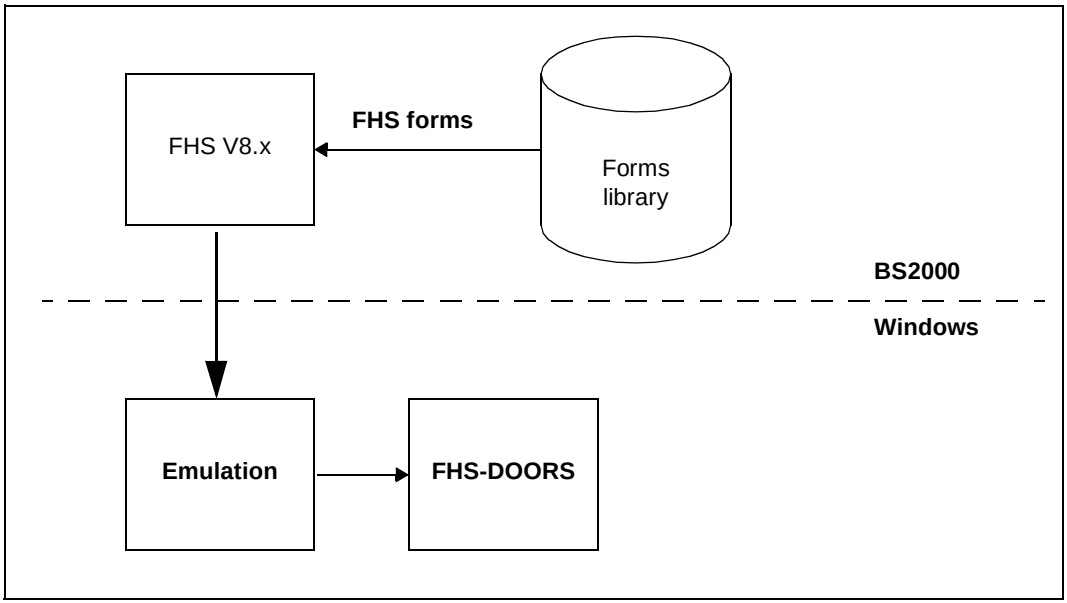


Figure 3: Automatic conversion of forms

Conversion of forms in BS2000

The form converter "FHS-DOORS-LC (BS2000/OSD)" allows you to convert FHS and FORMPLAG forms for use under FHS-DOORS. The forms to be converted must have been created with IFG for FHS and must be stored in a forms library (PLAM library) as type-R members.

The form converter reads the forms from the forms library, generates the corresponding panels (sdc files) and writes panels back to the forms library as type-S members and, if required, also writes them to a SAM file.

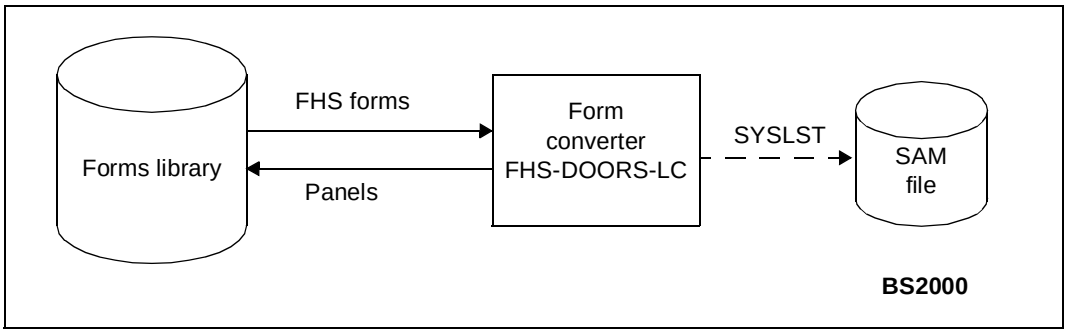


Figure 4: Conversion of forms

If FHS-DE is used, the panels generated by the form converter already include option buttons, check boxes, pushbuttons and menus. These panels will run directly under FHS-DOORS. Optimization of the panels is therefore unnecessary.

If help forms or message forms have been defined for a form, they must also be converted. If this is not done, messages to the effect that the help text or message text could not be found are issued when the user calls a help function or a message is issued.

Refer to chapter “Form conversion” on page 79ff for more details on converting forms and for a description of the FHS-DOORS-LC (BS2000/OSD) form converter.

Managing and distributing panels

After the forms have been converted to panels in BS2000 with the form converter, the panels (sdc files) must be made available on all the PCs which are to use the host application under FHS-DOORS.

Panels can be distributed:

– manually

– automatically using the download mechanism

Manual distribution of panels

If the panels are transferred manually, the sdc files must first be extracted from the forms library (with LMS or the BS2000 File Manager in DESK2000, for instance) and then stored in SAM files. These sdc SAM files are then transferred to the PCs (using File Transfer or the BS2000 File Manager, for instance), where they are stored in the resource path. If different host applications are used, the panels can be stored in different directories. All the panels for a single host application must, however, be stored in the same resource path.

If you wrote the panels directly to a SAM file via SYSLST when you converted them, you simply need to transfer this one SAM file to the PCs (you do not need to extract the panels from the forms library with LMS). However, you then need to use the *Tools/Split Resource Files* command in the DOORS editor to split the file into individual panels so that they can be used under FHS-DOORS.

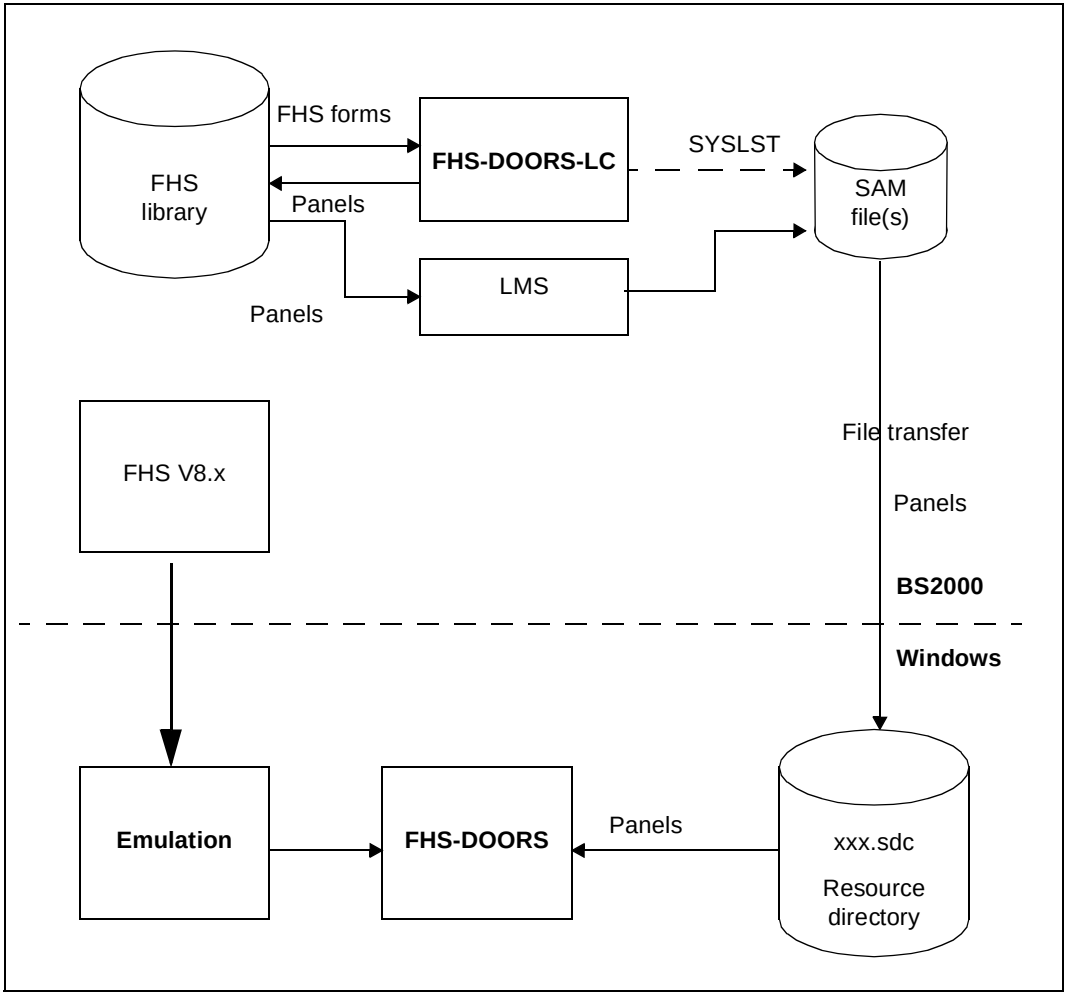


Figure 5: Transferring panels

Automatic distribution of panels (downloading)

Downloading is only possible for FHS/UTM applications and FORMPLAG applications.

If FHS-DOORS does not find a panel (sdc file) for a form on the PC, the panel is downloaded directly from the forms library at runtime and stored in a buffer directory (maskbuf).

A form is always downloaded if a panel is to be displayed, but no corresponding sdc file can be found in the resource path or in the buffer directory.

If necessary, older sdc files are deleted from the buffer directory during downloading to make room for new sdc files. You should not therefore store optimized panels (sdc files) in the buffer directory, since these could be overwritten during the downloading process.

When forms are converted to panels with the form converter under BS2000, the panels which are generated are written directly to the forms library. This means that the panels are immediately available for downloading.

You should also read the comments on downloading in chapter “Panel management and distribution” on page 73ff.

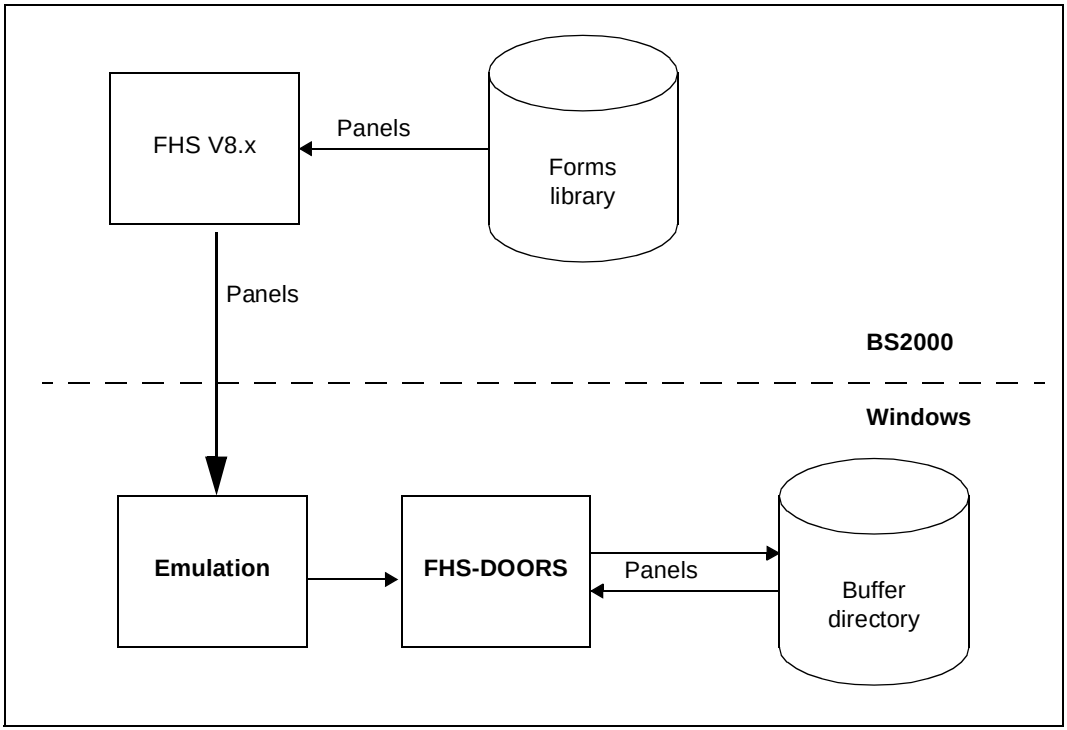


Figure 6: Automatic distribution of panels (downloading)

Optimizing panels with the DOORS editor

The DOORS editor allows you to optimize panels on the PC. You can, for instance, improve the appearance of a panel or emphasize important data. In short, you can adapt panels to suit the requirements of your organization. Refer to the manual "WIN-DOORS/FHS-DOORS, Optimizing Forms Using the DOORS Editor" [2] for a description of how to customize panels and the objects FHS-DOORS provides for this purpose.

Panels can be optimized step by step. If, for instance a host application provides 100 panels, but users generally only work with 10 of these, it may make sense to first optimize these 10 panels and to leave the remainder unchanged or only make minor changes. Panels converted with the form converter in BS2000 can be used directly under FHS DOORS without any optimization.

Testing panels

Once you have optimized panels with the DOORS editor, you should test the optimized panels to ensure that they work correctly with the host application under FHS-DOORS. You can activate debugging mode in FHS-DOORS to help you find any errors which may be present.

Automatic distribution of optimized panels (downloading)

The optimized panels can also be stored in the forms library. FHS-DOORS then loads these optimized panels directly from the forms library (see also section “Automatic distribution of panels (downloading)” on page 75ff).

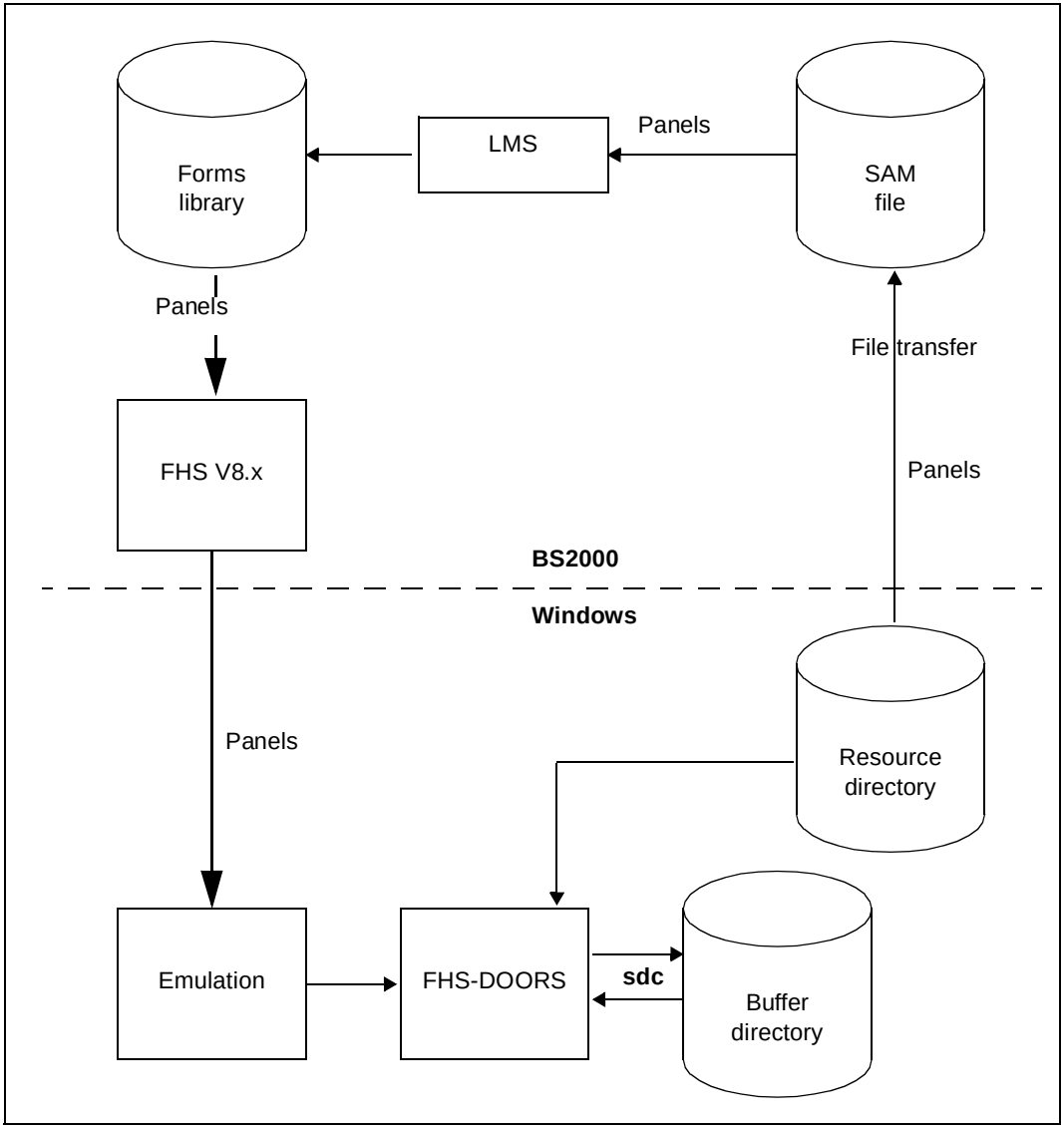


Figure 7: Automatic distribution of optimized panels (downloading)

2.3 Support for functions in FHS V8.0 and V8.1

Checking entry fields

FHS-DE provides checking options for fields. The checking options are defined when the FHS form is created.

The defined checking options are transferred to the panel (sdc file). The validity checks are then carried out by FHS-DOORS directly when the user makes an entry. If an invalid entry is made, a message is displayed in a message box. The message is either the standard FHS message or the message defined for the field with the DOORS editor.

If, in the DOORS editor, no error message was defined for a field (no error message number specified for the object), a user's entry is not checked by FHS-DOORS. The entry is passed directly to the host application. If the values passed are invalid,

– and if no error messages are defined for the FHS form, the values are passed to the host application and the errored values are flagged

– and if error messages are to be output for the FHS form, the transaction is canceled.

The DOORS editor allows you to add new field checks and modify or delete existing field checks. This can lead to a situation where a value is accepted by FHS-DOORS and passed to the host application, where it is not accepted. This leads to the transaction being canceled. To avoid this, you should always make sure that valid values only are passed to the host application when you are optimizing the semantics.

Outputting messages

FHS-DE allows messages to be output which provide information on specific events and thus simplify the dialog with the host application. The messages are defined in massage forms and can be addressed via an identifier (the message key).

The message identifier is passed separately to FHS-DOORS, which extracts the message with that identifier from the resource file and displays it in a message box. The message box can be moved to any position on the screen or closed again without the need for interaction with the host application. Thus all the fields of the form can be accessed at all times.

If message forms have been defined for a form, they must be converted along with the form. Otherwise, "Message not found" will be displayed when attempts are subsequently made to access the messages.

Help

FHS-DE allows an extensive help system to be generated for a host application.

Under FHS-DOORS, the help system runs directly on the PC. The help system does not involve communication between the PC and the host application unless a help panel has to be downloaded to the PC.

To call up help on an object you have to position the focus on the object and invoke the associated context-sensitive help function. You do this either by selecting a “Help on context” menu item or by pressing a “Help on context” key (e.g. F1). FHS-DOORS looks for the help for that object and displays it on the screen.

The object help forms are defined either in the forms library or with the DOORS editor. Help can be defined for:

– simple objects, such as entry fields or option buttons

– complex objects, such as an option button group

– windows

– messages

If help forms have been defined for a form, they must be converted along with the form. Otherwise, “Help not found” will be displayed when attempts are subsequently made to access the help text.

Dialog boxes (pop-up windows)

FHS-DE allows you to insert intermediate dialogs by superimposing a dialog box (pop-up window) on the form. These intermediate dialogs can be multilevel interactions, with multiple superimposed pop-up windows appearing on the screen.

FHS-DOORS supports pop-up windows in exactly the same way as FHS-DE.

Menus

FHS (FHS-DE) supports menus as of V8.1. If no panel exists for a form with menus, the menus are shown under FHS-DOORS in the same way as with FHS-DE. If, however, a panel has been created, the menus are displayed as pop-up menus (Windows menus) in the FHS-DOORS application window.

FHS commands

Some commands are processed directly by FHS and not by the host application itself. Under FHS-DOORS, these commands can be processed directly by FHS-DOORS without interaction with the host application.

HELP	Help on context. FHS-DOORS displays the help form for the selected object. If necessary, the panel is downloaded to the PC.
EXTHELP	Help on window. FHS-DOORS displays the help form for the window. If necessary, the panel is downloaded to the PC.
KEYSHELP	Help on keys. FHS-DOORS displays the help form for the KEY format. If necessary, the panel is downloaded to the PC.
HELPHELP	Help on help. FHS-DOORS displays the FHS-DOORS help panel for Help on Help.
HARDCOPY	HARDCOPY is supported by the emulation, not directly by FHS-DOORS.
SETP	SETP is not supported in the current version of FHS-DOORS.
CANCEL	The CANCEL command closes an implicit box. In FHS-DOORS, the message box or dialog box is closed by a function of the graphical interface (such as clicking on the *OK* or *Cancel* pushbutton).
EXIT	The EXIT command closes all implicit boxes. If EXIT is defined in the KEY format, an EXIT pushbutton is generated when the forms are converted. You can also specify that EXIT should be implemented as a menu item rather than as a pushbutton.

3 Using FHS-DOORS

3.1 Sessions and connections

FHS-DOORS uses so-called sessions for all connections to the host applications. A session contains certain parameters for the session which can be set with the *Setup* menu as well as the parameters indicating the host application and host computer to which a connection is to be established, the emulation to be used and the panels to be used (resource path). All these parameters can be configured as required for each session. The parameters can be stored in a “parameter file” so that the session does not need to be reconfigured every time you use it.

FHS-DOORS allows you to:

– open new sessions

– open saved sessions

– edit saved sessions

– close open sessions

– save open sessions

FHS-DOORS distinguishes three different statuses:

– “no session open”

– “session open, no connection established”

– “session open, connection established”

Generally, you will want to open a session and establish the connection immediately. Do this by clicking on *Start* in the relevant dialog boxes. FHS-DOORS then switches to the status “session open, connection established”.

If, however, you first want to edit/modify a saved session, you must open the session, but you must not establish the connection. Click on *OK* to open the session. FHS-DOORS then switches to the status “session open, no connection established”.

3.1.1 Parameter files for sessions

You can save the parameters of a session in parameter files in order to open that session at a later point in time or directly on starting FHS-DOORS (see “Starting a session directly” on page 40).

Parameters that are applicable to all sessions are stored in the *fhsd.ini* file in the Windows directory, while those that are only applicable to a special session are stored in the *.drs* file in the FHS-DOORS directory. The parameters in these two files can be edited to some extent either with menu commands or with an editor.

The parameter file for a session is subdivided into individual sections under various headers. Each of the parameters under a specific section is listed in a separate line together with its corresponding value. The following sections are included in the *fhsd.ini* file:

Section header	**Meaning**
misc	Contains parameters that affect FHS-DOORS settings
WindowState	The parameters in this section are managed by FHS-DOORS. They control the windows displayed by FHS-DOORS.
Recent File List	Contains the last four opened sessions; these parameters are managed by FHS-DOORS

Table 1: Structure of the *fhsd.ini* file

The following sections can be found in the *.drs* file:

Section header	**Meaning**
connection	Contains the connection parameters
misc	Contains parameters that determine how panels are displayed
Custom	Parameters for earlier DOORS versions
font	Parameters that define the font
debug	Contains just one parameter for debugging
display attributes	Parameters for converting BS2000 display attributes
alternate colors	Parameters for converting BS2000 display attributes with alternative colors
secondary windows	Contains parameters that determine the behavior of secondary windows (subpanels)
UsrDlls	Contains parameters for the dynamic library (DLL) that is read by FHS-DOORS when converting commands and special keys for buttons

Table 2: Structure of the *.drs* file

Section header	Meaning
Help	Parameters to call external Windows Help files
SpecialKeysSyntax	Parameters to convert special keys
popup	Contains parameters that can be used to control pop-up windows from “natural” applications

Table 2: Structure of the *.drs* file

If you want to edit or modify the parameters of a saved session before starting it, you will need to open the session, but without establishing the connection. To open a session, select the *OK* button. FHS-DOORS will then switch to the status “Session open; no connection established”.

The number of sessions that can be set up simultaneously depends on the used emulation, but only one connection can exist for each instance of FHS-DOORS.

A connection can be established only if:

– an emulation has already been started or

– the emulation is specified in the emulation parameter on opening a session directly and thus started automatically with it.

When FHS-DOORS is started (without specifying a parameter file), the initial FHS-DOORS status shows “no session opened”, i.e. only the primary window is displayed.

3.1.2 Parameters for all sessions (fhsd.ini)

The following parameters in the *misc* section of the *fhsd.ini* file can be modified by means of menu commands or a dialog box:

Parameter	Menu command/Dialog box
AutoTab	*Setup/Options->Miscellaneous*
DiagnosticsAlwaysOnTop	*Setup/Options->Miscellaneous*
DisableAutosend	*Tools/Macro*
Overwrite	*Setup/Options->Miscellaneous*
Statusbar	*Setup/Statusbar*
Toolbar	*Setup/Toolbar*

Table 3: Parameters of the *fhsd.ini* file that can be set via menu commands or a dialog box

A description of these commands can be found in the online help for FHS-DOORS. The two parameters listed below are also in the *misc* section of the *fhsd.ini* file, but can only be set with an editor such as Microsoft Notepad:

`OverwriteWhenSending={0|1}`

Defines whether the Insert mode for the keyboard is to be turned off when sending data to the host application.
The default value is 0.

`Trimodal={0|1}`

Defines whether or not FHS-DOORS uses a keyboard driver for trimodal keyboards.
The default value is 0.

Example

```
[Font]
FaceName=FHS DOORS
Style=Regular
Size=-13
Clip=2
Out=1
Quality=1
Family=49

[Debug]
Diagnosis=0
Semantics=0
; When this flag is set, a trace of the I/Os is taken in
; the file c:\windows\fhsddump.log
Dump=0

[Display Attributes]
Normal=0
Underlined=4
Bold=8
Flashing=17

[Alternate Colors]
Background=12632256
Foreground=255

[Misc]
AutoTab=1
Overwrite=1
StatusBar=1
3DLook=1
ShowFullMenus=0
Language=1
```

```
;
;Trimodal - When this flag is set, pressing the DEL key on the numeric keypad
; while holding the CTRL key down outputs a comma.
Trimodal=1
;
; CallMapScriptOnce - Diabuild „map“ script are called when a window is first
; displayed, but also when the BS2000 updates the screen. The following flag
; can be used to turn off this behavior.
CallMapScriptOnce=0
;
; Optimized810Protocol - This flag should be set when using some BS2000
; formatters (like FORMPLAG).
Optimized810Protocol=1
;
; ShowEmptyProtectedBoxes - Normally, an edit box, standing for a protected
; field which is either empty or holding only spaces or underscores, is
; automatically hidden. When this flag is set, these boxes stay visible.
ShowEmptyProtectedBoxes=0
ConstantLineHeight=0
ProcessLineMode=1
Mode=2107
ToolBar=1
DiagnosticAlwaysOnTop=1

[WindowState]
MainLeft=181
MainTop=36
MainWidth=638
Help=11 0 1 -1 -1 -1 -1 100 100 300 300

[Recent File List]
File1=C:\FHSDOORS\TUTORIAL\TUTORIAL.DRS
File2=C:\FHSDOORS\FHSD\PE.DRS
```

3.1.3 Parameters for different sessions (.drs file)

The following parameters in the *misc* section of the *.drs* file can be modified via menu commands or a dialog box:

Section header	Parameter	Menu command/Dialog box
connection	BufferAutoMngt	*Setup/Session->Options*
	BufferSize	*Setup/Session->Options*
	ConnectionId	*Setup/Session* or *Session/New*
	EmulationName	
	EmulationCmdLine	
	PromptBeforeDelete	*Setup/Session->Options*
	ResourcePath	*Setup/Session* or *Session/New*
misc	Alignment	*Setup/Options->Graphical Tuning*
	AutoResize	
	CharacterOverhead	
	ConstantLineHeight	
	HookDllPath	*Tools/Capture*
misc	LinesSpacing	*Setup/Options->Graphical Tuning*
	ObjectsSpacing	
	Optimized810Protocol	
	Overlapping	
	ProcessLineMode	*Setup/Options->Miscellaneous*
	RemoveTrailingSpaces	*Setup/Options->Graphical Tuning*
	3DLook	*Setup/Options->Graphical Tuning*
	ShwowEmptyProtectedBoxes	*Setup/Options->Graphical Tuning*
Custom	LookForWildLSPs	*Setup/Options->Miscellaneous*
font	Clip	*Setup/Font*
	FaceName	
	Family	
	Out	
	Quality	
	Size	

Table 4: Parameter of the *.drs* file that can be set via menu commands or dialog boxes

Section header	Parameter	Menu command/Dialog box
font	Style	*Setup/Font*
debug	Diagnosis	*Setup/Options->Miscellaneous*
display attributes	Bold	*Setup/Options->Display Attributes*
	Flashing	
	Normal	
	Underline	
alternate colors	Background	
	Foreground	

Table 4: Parameter of the *.drs* file that can be set via menu commands or dialog boxes

A description of these commands can be found in the online help for FHS-DOORS.

The parameters described below are also in the *.drs* file, but can only be set with an editor such as Microsoft Notepad.

The [connection] section

`AutomaticSwitching={0|1}`

If this parameter is set to a value of 1, every new panel of your application that is transferred will be displayed on the PC in the foreground.
The default value is 1.

`ConnectionTitle=`string

Title of the connection shown in the connection window. This parameter is evaluated only if the used emulation has an MDI interface.If the used emulation has an MDI interface and you supply a value for this parameter as well as for *EmulationTitle*, the emulation window will not be displayed when running the application.
The default value is an empty string.

`DoorsTerminalType={9750|9755|9763}`

Defines the type of terminal on which the BS2000 application displays its output. FHS-DOORS can simulate various terminal types; however, if you are using FHS Version V8.1 or later or any version of Formplag as of Version V2.4, you must enter 9763 as the terminal type here.
The default value is the terminal type 9750.

`DataTransferRate=`integer `K`

Defines the buffer size for data transmission with the BS2000 host. The buffer size can be extended up to 28K. FHS-DOORS automatically informs the partner system which buffer size has been set.
The default value is 4K.

`EmulationTitle=`string
: Title of the emulation shown in the emulation window. This parameter is evaluated only if the used emulation has an MDI interface.
If the used emulation has an MDI interface and you supply a value for this parameter as well as for *ConnectionTitle*, the emulation window will not be displayed when running the application.
The default value is an empty string.

The [misc] section

`AddDUEToEM={0|1}`
: If this parameter is set, FHS-DOORS will also send the value of the DUE key to the host application whenever the EM key is pressed.
The default value is 0.

`CallMapScriptOnce={0|1}`
: Defines whether a Dialog Builder script is executed by BS2000 when a panel is displayed for the first time or whenever a new screen is created.
The default value is 1.

`MaxEntryBoxes=`integer
: Defines the maximum number of entry fields that may be included in a form so that the form can be automatically converted or the converted panel can still be displayed. Forms that contain more than the specified number of entry fields are displayed alphanumerically in the emulation window. You can define a maximum limit of 220 entry fields here.
The default value is 150.

`ShowMainMenu={0|1}`
: Defines whether the primary window of FHS-DOORS is to be displayed when it is started directly. If the primary window is suppressed with the value 0, the individual commands will be available in a pop-up menu that can be opened with the right mouse button.
The default value is 1.

`PlusPlusFirst={0|1}`
: Sets a higher priority for the form identifier sent by the BS2000 application than for the user data in the form.
The default value is 0.

`ReplaceNilsBySpaces={0|1}`
: Causes all null values in the form sent by the BS2000 application to be converted to spaces before being displayed in the panel.
The default value is 0.

`SwitchToEmulWhenNoPlusPlus={0|1}`
Specifies that data from the BS2000 application is to be displayed in the emulation window if the form identifier is missing.
The default value is 0.

The [custom] section

`AppAdminCoordinates=`string
Defines a string that is referenced in error messages. This string should contain the name of the system administrator and the phone number of e-mail address (if required). The string may include any special characters and blanks.
The default value is an empty string.

Example

```
AppAdminCoordinates=Hans Müller, Tel: 0049/030/456 231
```

`PrimaryResPath=`string
Defines a directory, with the absolute pathname, from which resource files of type *.rbn* are loaded before starting the application.
The default value is an empty string.

`WindowTitle=`string
This string provides the title to be displayed in the first application window (not the primary window).
The default value is an empty string.

The [secondary window] section

`AutomaticClose=`{0|1}
When data of a panel is sent to the host application, the subpanels usually remain open. This behavior can be modified with the value 1 in order to automatically close all subpanels on sending data.
The default value is 0.

`CenterOnFather={0|1}`
By default, subpanels are displayed centered on the parent panel. Setting this parameter to 0 causes FHS-DOORS to read the x and y values of the subpanel position from a resource file (*res* file) that was created on editing the panel. If a general protection fault occurs in the XTXM.DLL module when reading the resource file, you can edit the resource file with the Dialog Builder.
The default value is 1.

The [UsrDlls] section

`keys1=string`

Captures of pushbuttons and their associated actions are read from a dynamically linked library (DLL) that is assigned to forms with this input field. The specified library is used.
Starting with FHS-DOORS V3.0B, a library to support PF keys is included in the FHS-DOORS directory. This library is called *pfkeys.dll* and is used by the Natural programming language.
There is no default value.

`keys_n`=string_n

You may specify any number of additional libraries for key conversions. This can be done by appending an integer in ascending order to the keyword “keys”.
There is no default value.

The [help] section

`FileName`=string

This string contains the name of the project file for Windows help.
The default value is an empty string.

identifier1=value1

The identifier is that of the Help topic and can also be defined with the Help ID attribute in the DOORS editor. The value is the associated context number that was defined in the MAP section in the project file of the Windows help.

identifier_n=value_n

You may specify any number of identifiers with their associated context numbers in this section.

The [SpecialKeysSyntax] section

`LeftPadded={0|1}`

Defines whether the representation of keys is displayed by means of a two-digit integer. Leading blanks are replaced by zeros.
The default value is 1.

`Prefix`=string

String with the prefix for PF keys.
The default value is P.

`Suffix`=string

String with the suffix for PF keys.
The default value is an empty string.

`EMPrefix`=string
: String with the prefix for the EM key.
The default value is EM.

`EMSuffix`=string
: String with the suffix for the EM key.
The default value is an empty string.

`LAPrefix`=string
: String with the prefix for the LA key.
The default value is LA.

`LASuffix`=string
: String with the suffix for the LA key.
The default value is an empty string.

`LVDPrefix`=string
: String with the prefix for the LVD key.
The default value is LVD.

`LVDSuffix`=string
: String with the suffix for the LVD key.
The default value is an empty string.

Example

```
[SpecialKeysSyntax]
Prefix=<P
Suffix=>
LeftPadded=1
```

With this setting, the P15 key is displayed as <P15>; the P3 key as <P03>.

The [popup] section

`UsePopupRecognition={0|1}`
: Causes FHS-DOORS to recognize pop-up windows that are sent along with forms by the host application. This behavior is typically useful for applications written in Natural (see also “Dialog boxes (pop-up windows)” on page 22). The individual pop-up windows are managed in independent panels. You should generally set this parameter only if you really need it, since the complex algorithm involved has a negative impact on performance.
The default value is 0.

`HStart`=string
: This string contains the characters with which the horizontal frame of a pop-up window begins.
The default value is a colon followed by a period **:.** .

`VStart`=string
: This string contains the characters with which the vertical frame of a pop-up window begins.
The default value is a colon followed by a period **:.** .

`HMiddle`=string
: This string contains the characters with which the horizontal frame of a pop-up window is displayed.
The default value is a period **.** .

`VMiddle`=string
: This string contains the characters with which the vertical frame of a pop-up window is displayed
The default value is a period **.** .

`HEnd`=string
: This string contains the characters with which the horizontal frame of a pop-up window ends.
The default value is a colon **:** .

`VEnd`=string
: This string contains the characters with which the vertical frame of a pop-up window ends.
The default value is a colon **:** .

`HOffset1={0|1}`
: Defines the number of spaces between the left margin and data in the pop-up window.
The default value is 0.

`HOffset2={0|1}`
: Defines the number of spaces between the right margin and data in the pop-up window.
The default value is 0.

`VOffset1={0|1}`
: Defines the number of spaces between the top margin and data in the pop-up window.
The default value is 0.

`VOffset2={0|1}`
: Defines the number of spaces between the bottom margin and data in the pop-up window.
The default value is 0.

Example

```
[connection]
BufferAutoMngt=0
PromptBeforeDelete=0
BufferSize=0
EmulationName=C:\DOORS_EM\DOORS_EM.EXE
EmulationCmdLine=C:\DOORS_EM\D016ZE07.DRE
ConnectionID=
DoorsTerminalType=
ResourcePath=c:\fhsdoors\tutorial

[misc]
ShowMainMenu=1
Optimized810Protocol=1
ShowEmptyProtectedBoxes=0
RemoveTrailingSpaces=32
MaxCacheEntry=10
ProcessLineMode=1
AutoResize=1
3DLook=1
HookDllPath=

[Custom]
LookForWildLSPs=0

[font]
FaceName=FHS DOORS
Style=Regular
Size=-13
Clip=2
Out=1
Quality=1
Family=49

[debug]
Diagnosis=0
[display attributes]
Normal=0
Underlined=4
Bold=8
Flashing=17

[alternate colors]
Background=12632256
Foreground=255

[secondary windows]
AutomaticClose=0
CenterOnFather=1
```

3.2 Starting and terminating FHS-DOORS

Start FHS-DOORS by double-clicking on the *FHS-DOORS* icon in the Windows 3.11 Program Manager or via the Start menu in Windows 95.

You can also start a session directly. For details, refer to “Starting a session directly” on page 40.

After FHS-DOORS has started, the FHS-DOORS primary window appears.

From the primary window (main window), choose *Session/New*, select the resource path for the panels of the host application and specify the command used to call the emulation in the *Emulation parameters* fields. Choose *Session/Open* to open a session you saved previously with *Session/Save*.

Figure 8: FHS-DOORS primary window

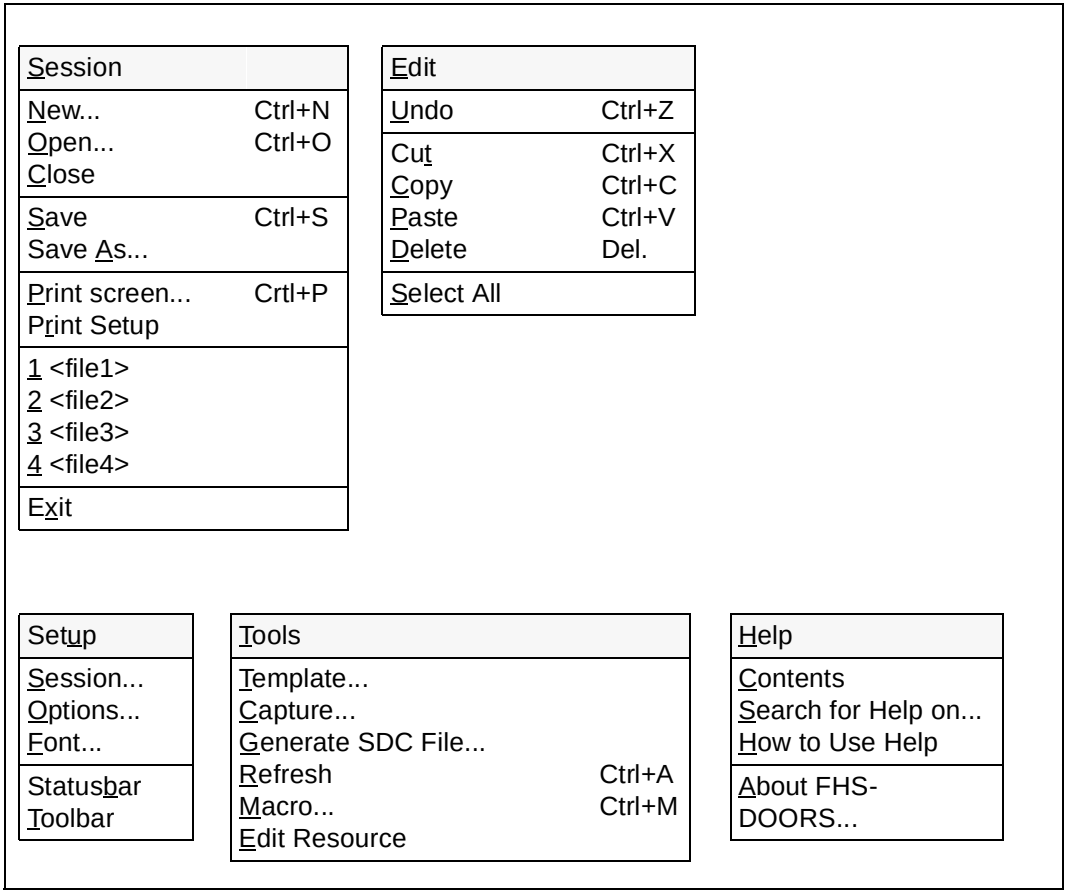


Figure 9: Menus in FHS-DOORS

Terminating FHS-DOORS

To terminate FHS-DOORS, choose *Session/Exit*.

Current connections are cleared as follows:

– by closing the current session in FHS-DOORS

– by terminating the host application on the host computer, e.g. by entering KDCOFF for UTM applications or LOGOFF for user programs.

If a session was started directly (see “Starting a session directly” on page 40), FHS-DOORS is also terminated when the session is closed.

Starting a session directly

FHS-DOORS offers an the option to start a session directly, i.e. to start FHS-DOORS, open the session, and establish a connection to the host computer.

fhsd␣parameter-file

The parameter file must exist (see also the section “Parameter files for sessions” on page 26).

When a session that was started directly is closed, FHS-DOORS is also terminated automatically.

If desired, you can create a separate icon for each session in the Windows 3.x Program Manager or the Windows 95 Start Menu and then open that session directly by simply double-clicking on the icon.

Windows 3.x

Copy the FHS-DOORS icon in the Program Manager (using the *File/Copy* command of the Program Manager) and change the program properties (using the *File/Properties* command) as follows:

Description Enter the text that is to be appear under the icon

Command Line fhsd-installation-directory**\fhsd**␣parameter-file

More information on the Program Manager can be found in the Microsoft Windows documentation.

Windows 95

Create a shortcut to the FHS-DOORS program by using the Windows 95 Explorer (with *Edit/Copy* or *Edit/Create Shortcut*). Then change the program properties in the *Properties* dialog box (with the *File/Properties* command) by entering the following via the *Shortcut* tab:

Target fhsd-installation-directory**\fhsd**␣parameter-file

More information on the Explorer can be found in the Microsoft Windows 95 documentation.

3.3 Opening sessions and establishing connections

There are two ways of opening a session and establishing a connection with FHS-DOORS:

1. Open a new session (see the *Session/New* command in the FHS-DOORS online help system)
2. Open a saved session (see the *Session/Open* command in the FHS-DOORS online help system)

For establishing connections, FHS-DOORS supports all connection methods/ transport access systems that are also supported by the emulation used. The DOORS emulation (supplied with FHS-DOORS) supports the connection methods LAN/CMX, BAM and HDLC/AFP.

3.3.1 Opening a new session

The *Session/New* command allows you to specify the resource path, the emulation parameters and mask buffer management for the new session.

In Emulation Parameters, specify the command for calling the emulation to be used to establish the connection. The command used to call the emulation depends on the emulation you are using. If the connection is to be established via the DOORS emulation, you must enter the following under *File Name*:

emul-installation-directory**\doors_em.exe** parameter-file

The parameter file must exist (see also “Creating a parameter file for the emulation” on page 111)

or

emul-installation-directory**\doors_em.exe -C** connection-name

The connection name must exist.

Opening a session and establishing a connection

Click *Start* to open the session and establish the connection.

Depending on the way in which the host application you wish to use was generated, you may first have to log on. The logon procedure with FHS-DOORS is the same as when FHS-DOORS is not used. Startup scripts allow you to automate this logon procedure if you are using the DOORS emulation for establishing the connection (See the manual “DOORS Emulation, 9750 Basic Emulation for BS2000/OSD Connections” [3].

Once the session has been opened and the connection established, you can conduct a dialog with the host application in the FHS-DOORS application window.

Parts of the dialog with the application which are not handled with forms (e.g. KDCSIGN with UTM applications or SET-LOGON-PARAMETERS and START-PROGRAM for user programs) are handled by the emulation in the emulation window.

Opening a session without establishing a connection

Click *OK* to open the session without establishing a connection.

3.3.2 Editing the current session

The *Setup/Session* command enables you to view and change the resource path, emulation parameters and mask buffer management settings for the current session.

You can therefore select different panels in the current dialog by selecting a different resource path. This is useful, for example, when optimizing panels, as you can then display and test the newly optimized panels immediately.

If you want to change the emulation parameters or the mask buffer management, you must first clear down the connection (by choosing *Setup/Session* and clicking on *Stop*). Now you can, for example, select a different emulation or change the maximum buffer size. By clicking on *Start*, you can establish the connection with the new parameters.

Settings for the current session made with *Setup/Options* (Font, Display Attributes, Auto Tab, Overwrite, etc.) can be changed at any time. For example, to switch the status bar display on or off, choose *Setup/Status Bar*.

3.3.3 Saving the current session

The *Session/Save* or *Session/Save As...* command allows you to store the parameters of the current session in a parameter file so that you can open this session again later.

3.4 Designing user interfaces with subpanels

The output that is displayed by BS2000 application to the user is based on forms. A form is a contiguous line area with a specific field structure. It is not bound to a line position within a window. A specific form may appear at various positions in different windows.

Panels are developed with FHS-DOORS on the basis of underlying forms, i.e. FHS-DOORS composes the outputs in the window from the panels of the recognized forms. The representation of a field structure that appears in multiple forms can now be defined just once as a subpanel and need not be defined individually for each panel.

3.4.1 Recognizing forms and displaying the interface

FHS-DOORS recognizes forms on the basis of output attributes, which are maintained in a special database called the Capture database. Each capture in this database links an output attribute with one or more form assignments, and each form assignment assigns a form to a specific line position in the window. The records are arranged in order.

Whenever the BS2000 application modifies the output buffer, FHS-DOORS goes through the captures in the defined order and checks whether the respective attribute applies to the current form. If a match is found, the form assignments of the record are noted for the panel setup, provided that they do not coincide with the form assignments noted from an earlier match.

FHS-DOORS composes the final output from the (sub)panels of the noted forms in the defined order. If there are output areas for which no form assignment can be found on searching the database, the subpanels for these areas are generated dynamically.

The following rules apply to the assignment of subpanels:

– A specific form may be referenced in multiple captures.

– A specific form may be referenced in single capture more than once with different line positions.

– A specific output attribute can be used in multiple captures.

– The empty attribute is a special type of output attribute that always results in a match in every form. It can also be used in multiple captures.

3.4.2 Capture database

The Capture database can be processed with the *Tools/Capture* command. Every capture has a unique name (called the Capture name). The names of the capture used by FHS-DOORS for attribute recognition are listed in the corresponding order in the *Capture* dialog box. This order can be changed by using the two arrow buttons. For more information on how to create a capture, see section “Creating a capture” on page 55.

Two status flags on the left of the Capture name indicate the status of the capture in relation to the representation of the current output. The M flag (for Match) indicates a match; the A flag (for Active) means that the from assignment of the capture are used for the output. The status flags are followed by the number of resources defined in the form assignment.

The captures in the Capture database have the line position and the resource name for a form stored in them, but not the structure of the form. A description of the structure is stored together with the information for the graphical setup of the panel in the resource file. The creation of a new resource file is described in section “Generating a resource file” on page 56.

3.4.3 Tips on panel design

In order to complete the design of the graphical interface, the resource files are opened and processed individually in the editor. The following tips are useful in this context:

- The window menu graphical object: A window menu can be defined in every resource file. If the output is composed from the subpanels of multiple forms, the window menu for the output is built by combining the menu definitions of the individual panels.
- The alphanum graphical object: In order to have a correctly functioning panel, the field structure of the current output must usually match the corresponding structure stored in the resource. Deviations are possible only within very narrow limits, i.e. an I/O field in the resource may be matched by a pure output field in the current panel or several contiguous output fields in the resource may correspond to a single large output field in the panel.

  If an alphanumeric object is used in the panel, the situation changes, i.e. the field structure stored in the resource is not relevant for the form area of the alphanumeric object. This is because alphanumeric objects always work with the field structure that is current at runtime. A resource with an alphanumeric object thus defines the representation of an entire class of forms.

There are three typical cases in which alphanumeric objects are used:

- Screens with several (more than hundred) fields
  An alphanumeric object can be used to combine multiple fields into one graphical object. This reduces the number of graphical objects and thus the time required to set up the panel.
- Output information for the clipboard
  Under normal circumstances, only the information that can be overwritten in the screen of the BS2000 application can be marked in a panel and copied to the clipboard. In the case of an alphanumeric object, any text may be marked and copied to the clipboard, including the data in pure output fields.
- Dynamically structured screen areas
  Screen areas that frequently vary in structure, i.e. because the structure is defined by output data and only determined at runtime, can be easily incorporated in the graphical interface by means of an alphanumeric object.

The use of alphanumeric objects is a new feature in DOORS V3.1. One example of how such objects are used is the “built-in” LINEMODE resource for displaying screen outputs in line (Rollup) mode in the runtime system.

3.5 Online help for FHS-DOORS

The online help for FHS-DOORS contains a complete description of the FHS-DOORS interface.

There are a number of ways of calling the online help system:

– Choose *Help/Contents* to display a list of the topics in the online help system.

– Choose *Help/Search for Help on* to search for a particular topic or keyword.

– If you want to display help for a particular window or dialog box, press F1 or click on *Help*.

– Context-sensitive help for an object can be obtained by clicking on the icon for the “question-mark cursor” in the toolbar and then with the question-mark cursor on the object for which help is required. If no context-sensitive help for the selected object is available, help on the current window or on the current dialog box is displayed.

– The question-mark cursor cannot be clicked in a modal dialog box. If you want to request context-sensitive help for an object in this case, you will need to first select the corresponding object and then press the Ctrl + F1 key combination.

Choose *Help/How to Use Help* for more details on how to use the Microsoft Windows online help system.

4 Sample session

This chapter is intended to provide you with a quick introduction to FHS-DOORS. It is based on a single and progressive example that illustrates how you can graphically convert forms of BS2000 applications with FHS-DOORS and then edit them with the DOORS editor.

The sample session is supplied with FHS-DOORS and is stored in the *tutorial* directory during installation. When you run this session, you will be working without a connection to a host application, i.e. the behavior of the application is merely simulated. The emulation used when running the sample session is the DOORS emulation.

In order to correctly run this sample session, you will need to have a complete installation of FHS-DOORS, which includes the DOORS emulation and the example.

Escape routes in the case of errors

– If you have the feeling that you are stuck somewhere in the middle of the sample session, you can return to the previous state by pressing the *Cancel* button.

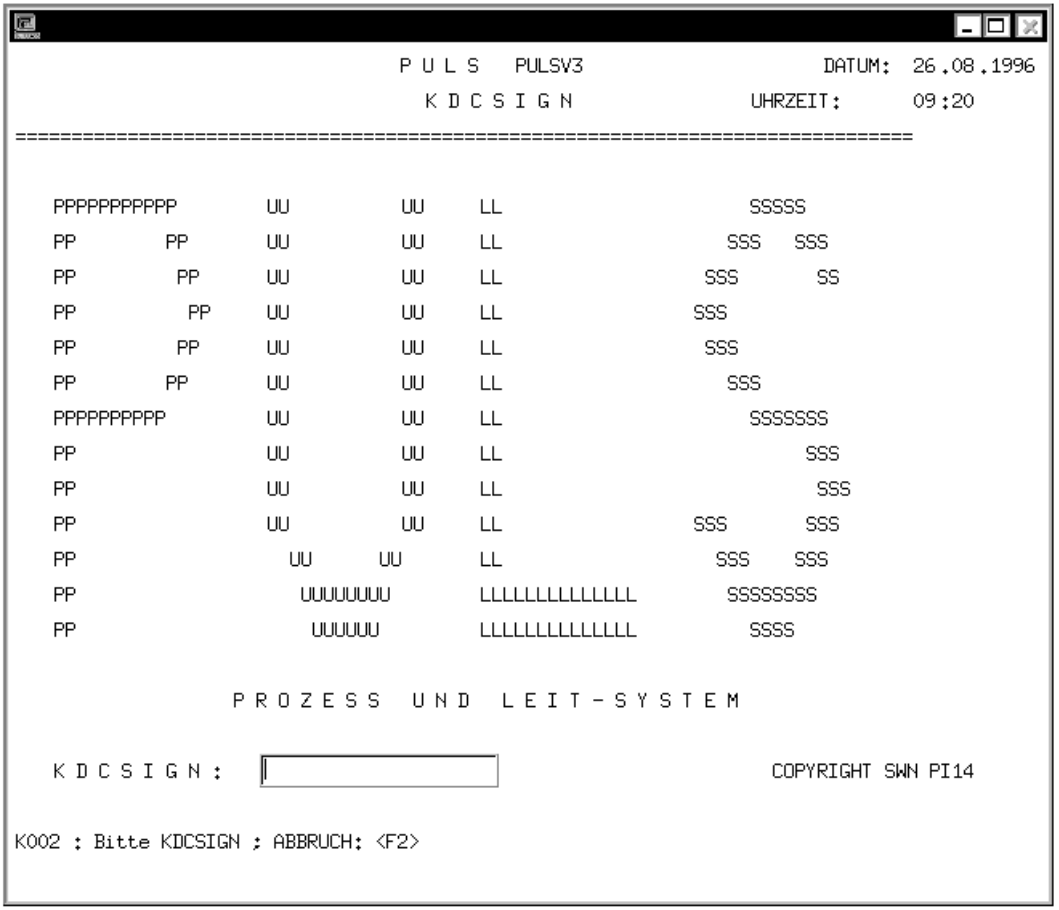
– The Alt + Tab key combination can be used to toggle between the individual (panel and emulation) windows of the sample session.

– A detailed description of the FHS-DOORS interface can be found by means of the Help function.

4.1 Starting the sample session

► Double-click on the FHS program icon in the FHS-DOORS program group or select the program icon via the Start menu of Windows 95. The primary window of FHS-DOORS will appear.

- Select the *Session/Open* command. An *Open* dialog box will be displayed. Select the *Tutorial* subdirectory in the installation directory of FHS-DOORS and then the file *Tutorial.drs*.
- Click on *OK*. This will start the sample session.



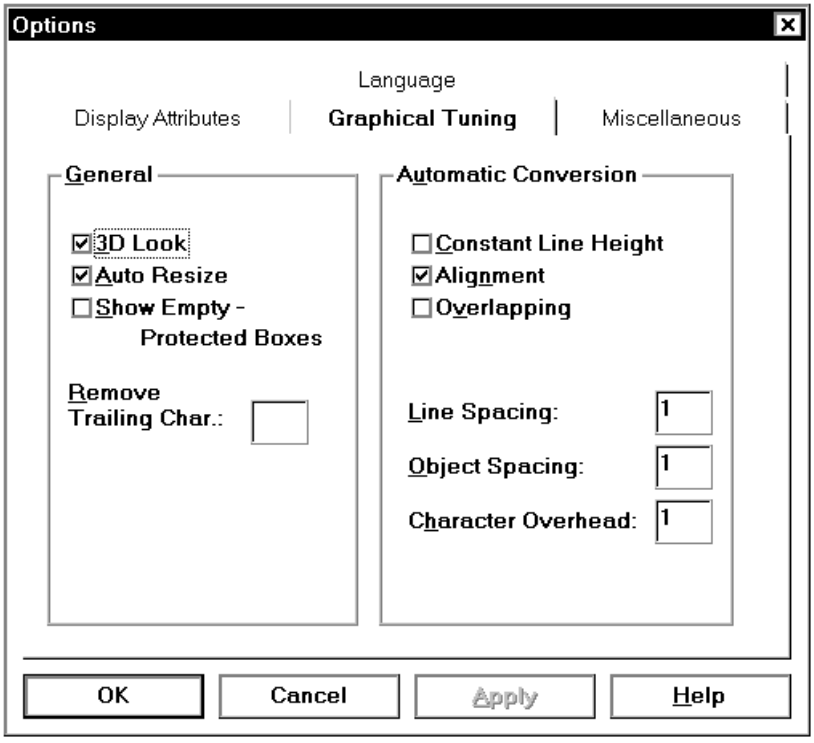
The first form of the sample session is automatically converted to a panel and displayed on the screen. This form will not have any template assigned to it, as indicated by the entry *[none]* in the FHS-DOORS toolbar.

- Use the Alt + Tab key combination to bring the DOORS emulation window to the foreground. This key combination should be used throughout the sample session whenever you need to switch to individual windows of the application.

Configuring FHS-DOORS for the sample session

In order to obtain a better display of the panel, you should first set some options:

- ► This can be done by selecting the *Setup/Options* command. The *Options* dialog box will appear on the screen.
- ► Select the *Graphical Tuning* tab by clicking on it.



- ► Deactivate the *Constant Line Height* option in the *Automatic Conversion* group and confirm it with *OK*.

You can now customize the sample session for your work environment:

- ► Select the *Setup/Session* command. The *Edit Session* dialog box will be displayed.
- ► Verify that the directory containing the sample session has been set in the *Resource Path* field.

- ► Ensure that the program file of the DOORS emulation has been specified with the correct path in the *Name* field (*C:\DOORS_EM\DOORS_EM.EXE* for a default installation) and that the parameter file (*.dre* extension) for the emulation that is entered in the *Parameter* field is *TUTORIAL.DRE*.
- ► After you have verified that these settings are correct, click *OK* to call the sample session.

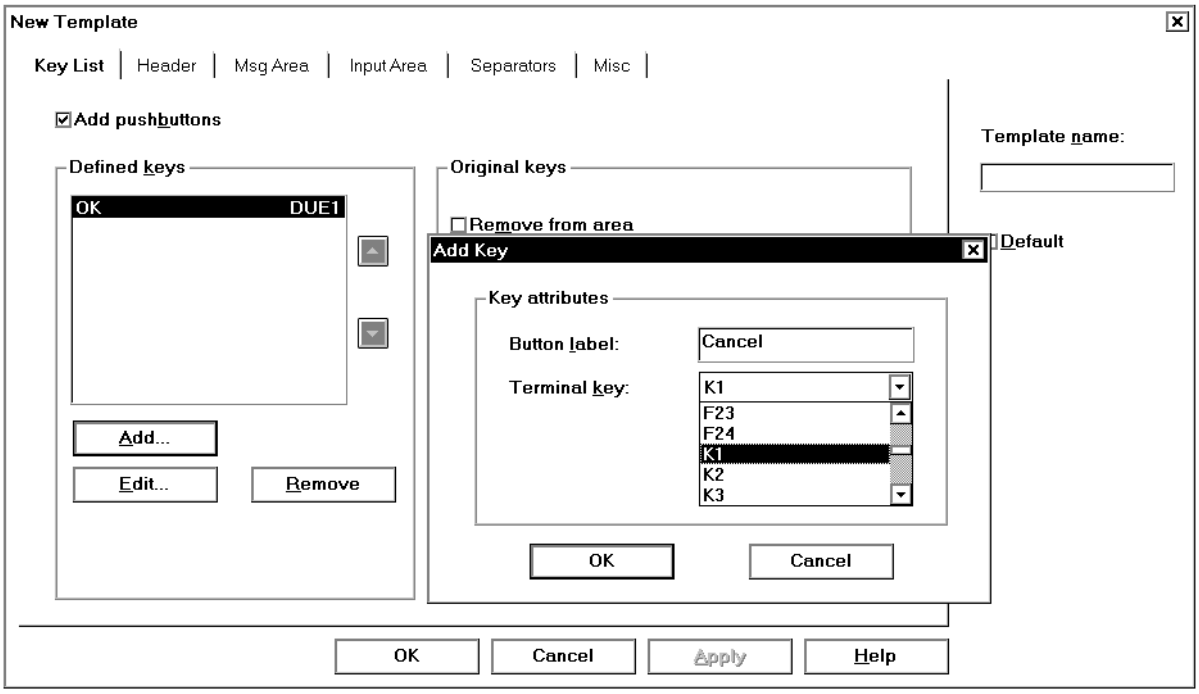
The sample session will run as described if you accepted the default directories when installing FHS-DOORS.

4.2 Working with templates

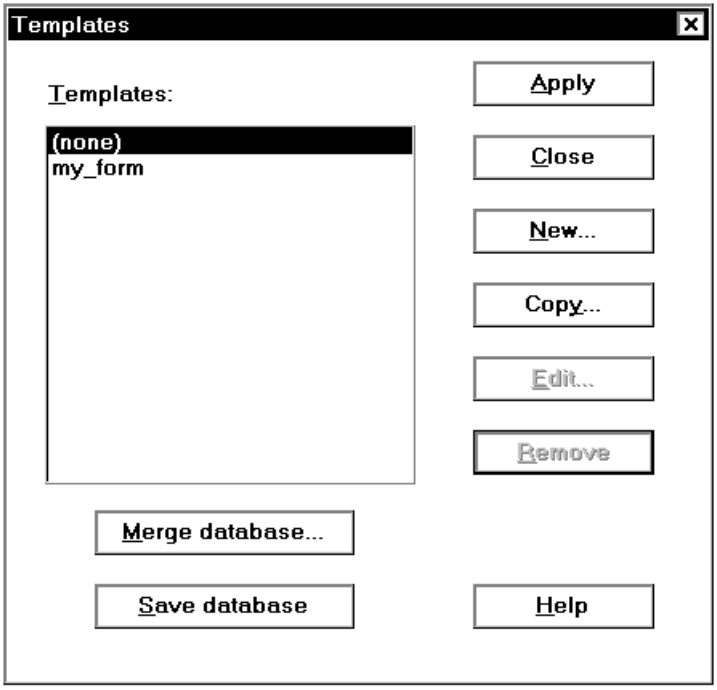
Templates can be used to control how automatic conversion occurs. One example for the use of a template is the creation of *OK* and *Cancel* buttons in every panel.

4.2.1 Creating a template

- ► Select the *Tools/Template* command or click the corresponding icon in the toolbar. The *Template* dialog box will appear. All templates already created would normally appear here in the *Template* list. At present, you will find only the entry *none*.
- ► To create a new template, click the *New* button. The *New Template* dialog box appears. This dialog box comprises a number of tabs that can be selected to set various options for a template.
- ► In order to have buttons created during automatic conversion, you will need to click on the *Keys* tab to activate it.
- ► Select the *Add pushbuttons* option to activate the *Add* button.
- ► Click *Add*. The *Add keys* dialog box appears. This dialog box can be used to define the label for the button and the corresponding action.



- Enter **OK** in the *Button Label* entry field.
- Select the entry *DÜ1* from the *Terminal key* list and confirm your selection with *OK*. The dialog box is closed, and your input appears in the *Defined Keys* list.
- Use the same method to define a **Cancel** button for the *K1* action.
- Now enter a name for your template in the *Template name* field of the *New Template* dialog box, i.e. my_form. If you forget to specify a name for your template, you will receive an error message.
- Select the *Default* option so that FHS-DOORS assigns your template to all forms during automatic conversion.
- Confirm your settings with *OK* and do the same for the prompt that asks you whether you really want to change the default template. This closes the dialog box with the prompt. A new entry called *my_form* should now be visible in the *Templates* list in the *Template* dialog box.



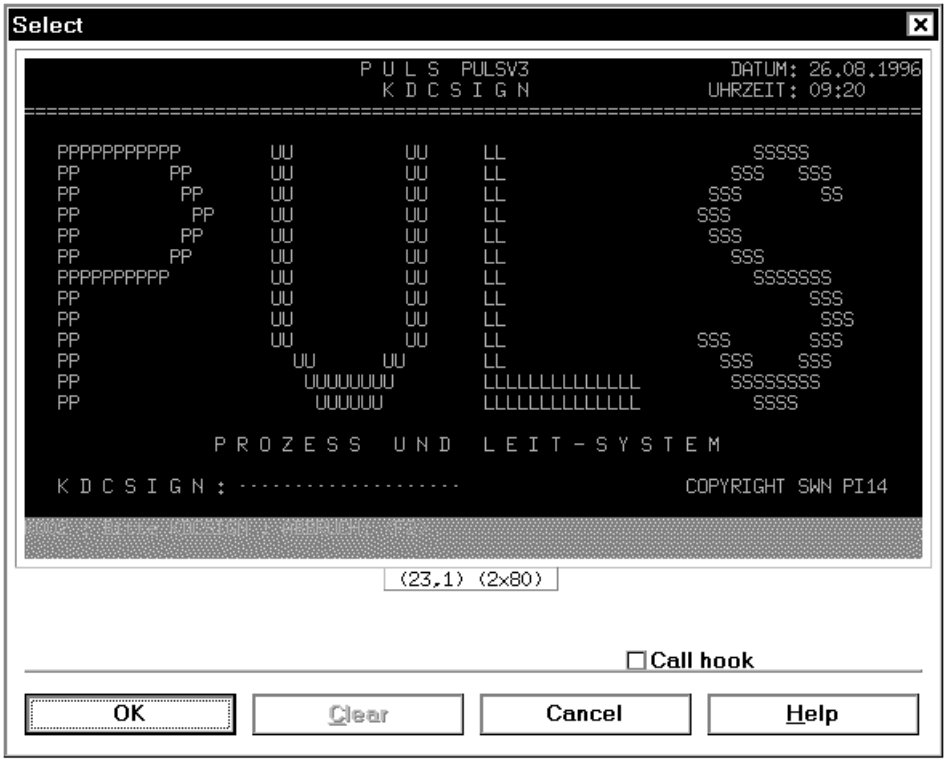
In order to have even the first panel displayed with this new template, you will now need to assign the template to it.

- This can be done by clicking on *Apply* in the *Template* dialog box.

4.2.2 Editing a template

The next step is to edit the template in order to define a message area and a separator for the forms of your application.

- In the *Template* dialog box, click on *Edit*. If you had already closed the dialog box, you will need to open it again with the *Tools/Template* command. The *Edit Template* dialog box appears on the screen.
- Select the *Msg Area* tab in the *Edit Template* dialog box.
- Now choose the *Create message box* option in the *Msg Area*. This activates the *Select* button.
- Click on *Select* to display the *Select* dialog box.



- ► Use the left mouse button to mark the last two lines in the form where messages are usually output by the host application. If you have marked the lines correctly, the last two lines should be displayed in a different color (see diagram).
- ► Confirm the marked lines by clicking on *OK*.
- ► The line values for your marked lines are copied to the fields of the *Original Message* group. They should now have the following values:

  Starting line 23

  Height of area 2

  If you do not succeed at marking the lines for some reason, you can also enter the values directly into the entry fields.
- ► Deactivate the *Protected lines only* option.

This completes the definition of a message area in the form. Messages received from the host application will now be displayed in a separate message panel on the PC display terminal.

You should now replace the alphanumeric separators by graphical ones.

- ► To do this, select the *Separators* tab in the *Edit Template* dialog box.
- ► Select the *Replace by graphical separator* option in the *Horizontal Separator* group.
- ► In the *Alphanumeric Separator* input field, enter the characters **= _ *** without any delimiter between them. This defines the characters to be replaced by a graphical separator during automatic conversion.
- ► Enter the value **3** in the *Min. number of characters* entry field. This value indicates that at least 3 of the specified characters must be present at the start of a line in order for them to be replaced by a graphical separator.

In order to ensure that the editor can edit the representation of labels in the panel, the following setting must still be made:

- ► In the *Edit Template* dialog box, select the *Misc* tab.
- ► Select the *Split at blanks* option and accept the default value (2). Labels that are too long will now be wrapped during automatic conversion.
- ► Confirm your settings with *OK* in the *Edit Template* dialog box.

4.2.3 Assigning a template to a panel

- ► To make your changes effective for the current panel, you will need to click *Apply* in the opened *Template* dialog box. You should now see the following message panel on the screen:

- ► Confirm the message window of the application with *OK*.
- ► Close the *Template* dialog box by clicking on *Close*.
- ► Enter your name in the *KDCSIGN* entry field in the first panel of the sample application and send this value to the host application by clicking on the new *OK* button.

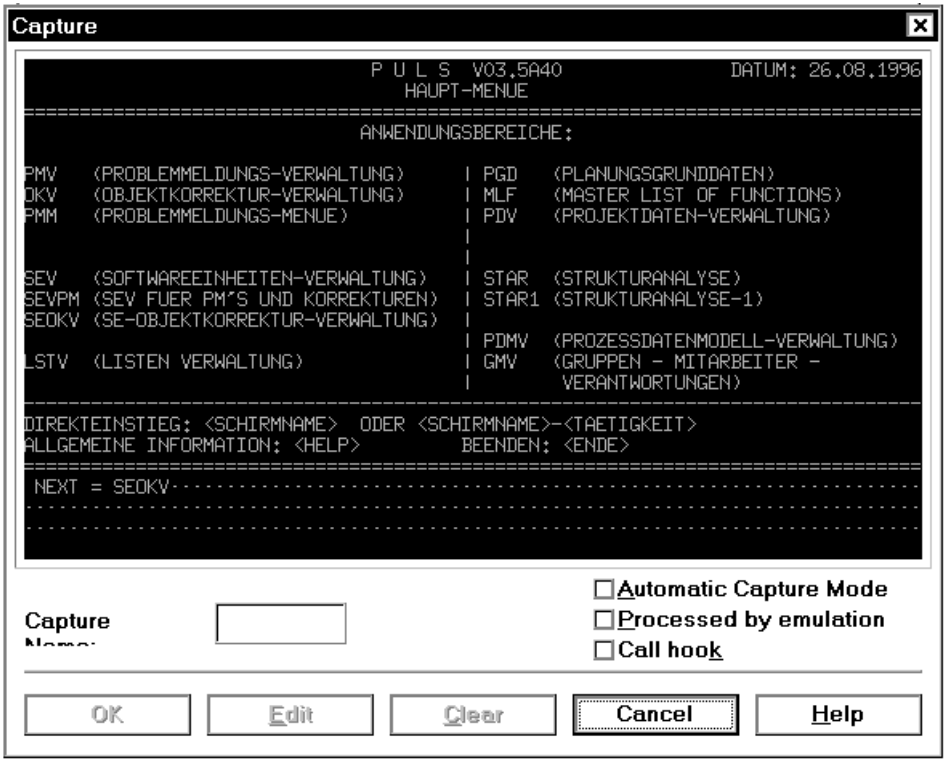
  The next form of the sample application is automatically converted in accordance with the template specifications and displayed as a panel on the screen.

4.3 Optimizing a panel with the DOORS editor

If you are not quite satisfied with the result of the automatic conversion, you can also customize the appearance of the displayed image by using the DOORS editor. In order to do this, the panel must be stored as a resource file. This resource file can then be edited with the editor.

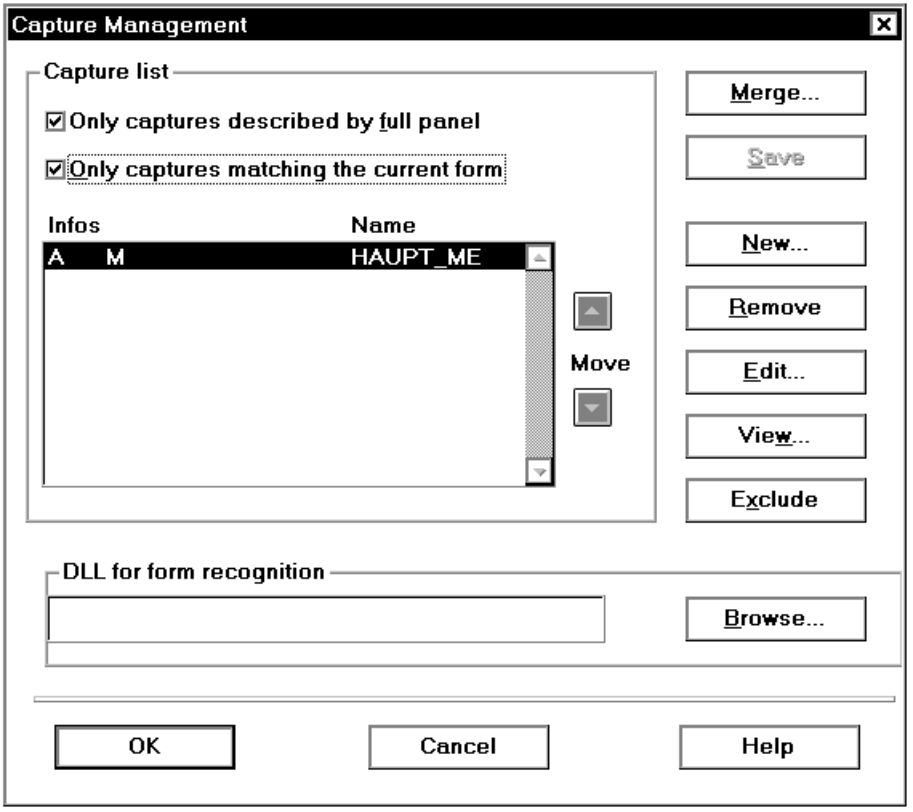
4.3.1 Creating a capture

- Select the *Tools/Capture* command. The *Capture* and *Capture Management* dialog boxes appear on the screen. It contains the alphanumeric representation of the current panel.
- In order to create a capture for a form, you must first select a form identifier. Do this by marking the term *MAIN-MENU* in the displayed form with the left mouse button.



The marked form identifier is copied into the *Capture Name* entry field.

- ► Confirm the name for the identifier with *OK*. This closes the *Capture* dialog box.
- ► You will find an overview of all existing captures in the *Capture Management* dialog box. The new capture will appear in the *Capture List*.



- ► Confirm the new capture with *OK*. This closes the *Capture Management* dialog box.

4.3.2 Generating a resource file

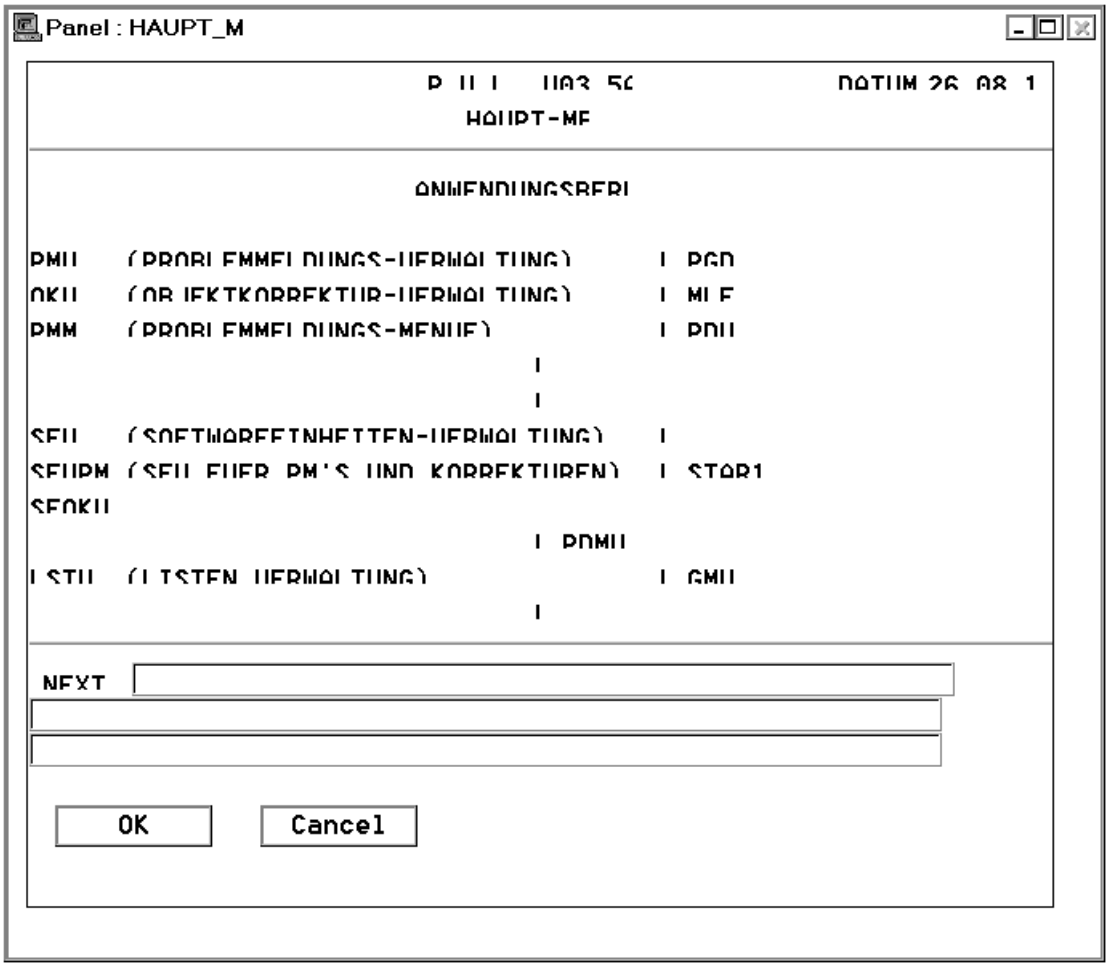
The capture created above can now be used to generate a resource file named MAIN-MEN.sdc.

- ► To do this, select the *Tools/Generate SDC File* command. The resource file is created and placed in the resource directory with no further message.

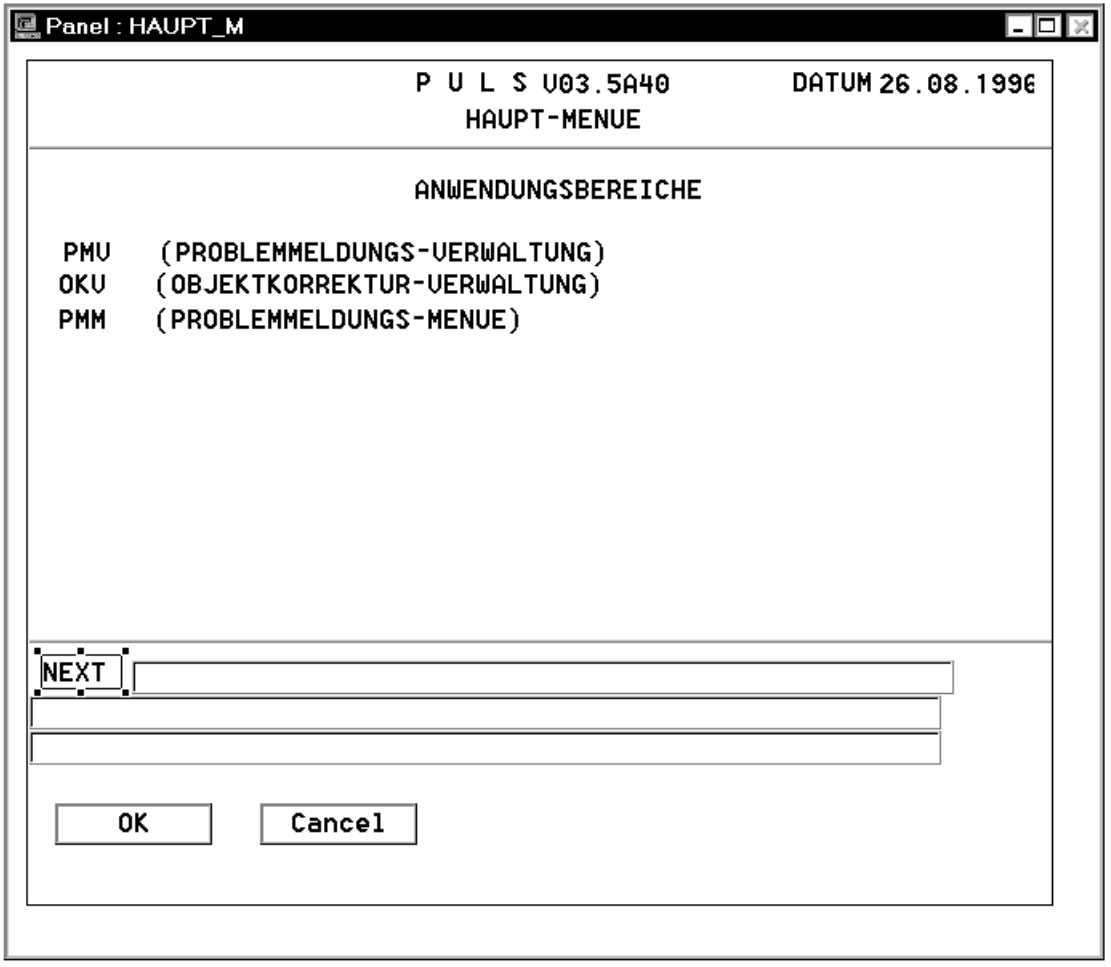
- Call up the DOORS editor with the *Tools/Edit Resource* command. This starts the editor with the current panel displayed in its primary window. The newly created resource file MAIN-MEN.sdc is used by the DOORS editor as the input file in which all changes are stored.

4.3.3 Editing a panel

In this example, the panel with the main menu is to be modified so that only the commands which are useful for an inexperienced user can be selected. To do this, you must first delete the unnecessary objects from the panel.



- To delete the objects, click the selection tool  in the toolbar and draw a frame around the objects that you wish to delete. The selected objects will be shown as marked.
  This can also be done by holding down the Shift key and clicking on the objects individually with the mouse.
- To delete the objects, press the DEL key or select the *Edit/Delete* command.
- Delete all redundant objects in the panel so that only the following objects are visible:

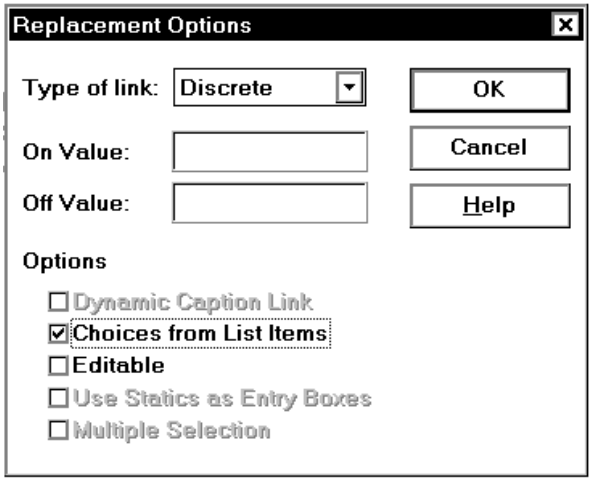


Note that some text fields may also include some hidden text. In this example, all help information is supplied with a variable value by the BS2000 application. These values are shown in angle brackets.

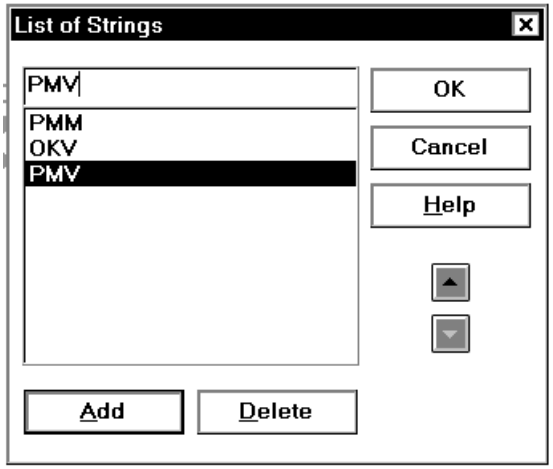
- ► Use the selection tool to mark the six text fields and move them with the mouse to the upper middle area of the panel. If you do not succeed at moving them on your first attempt, you have probably selected the panel as well (as in the diagram). Deselect the panel by pressing the CTRL key and clicking on the panel.
- ► Retain the marked text fields and select the *Graphical Tuning* tab in the properties window.
- ► Double-click on *Font* in the *Graphical Tuning* tab. This opens the *Font* dialog box.
- ► Select some other *Font* and *Size* for the text fields, e.g. the Arial font with a 10 point size.
- ► Confirm your selection with *OK*. The marked text fields will be displayed in the selected font. If the selected font is too wide or the font size is too large, the text frame may be too small, and the displayed text may be truncated. If this occurs, you will need to click on the text frames individually and extend them as required to make the entire text visible.
- ► Now arrange the fields of the first column so that they end at the same height. This can be done by marking the fields and selecting the *Layout/Align Objects>Right* command. All objects will be aligned on the last selected object.
- ► Arrange the fields of the second column so that they all begin at the same height as well. This can be done by marking the fields and selecting the *Layout/Align Objects>Left* command. All objects are aligned on the last selected object.

So far, you have only processed the visual aspects of the panel in the sample session. You can, however, also use the DOORS editor to create graphical objects in the panel and associate them with actions. Existing objects can be converted to graphical objects by using the “Replace” tool. In the next step, the multi-line entry field *NEXT* at the end of the panel is to be replaced by a list.

- ► To do this, mark the first line of the multi-line entry field.
- ► Press the right-hand mouse button to display the context menu.
- ► Select the *Replace By>Combo Box* command. This opens the *Replacement Options* dialog box.



- Select the *Choices from List items* option and confirm it with *OK*. The dialog box is closed, and the
  entry field is replaced by an empty list.
- Change the size of the frame for the choicelist by extending it downward to define the size of the list in a drop-down state.
- Check whether the list is still marked or mark it if required and click the *Graphical Tuning* tab in the properties window.
- In the *Graphical Tuning* tab, double-click the *List Items* property. The *List of Strings* dialog box appears.
- Enter the names of the individual elements in the entry fields of the *List of Strings* dialog box and confirm each entry with *Add*.

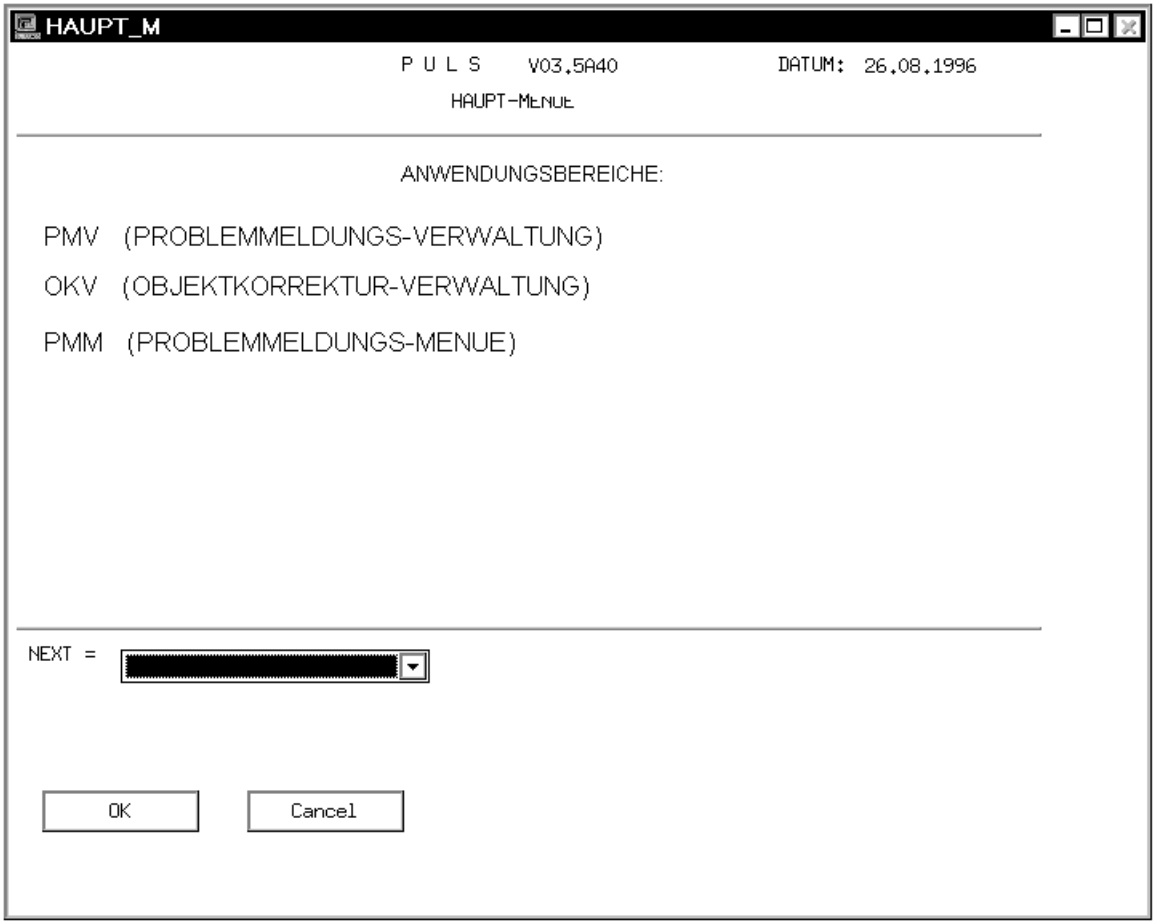


- Close the dialog box with *OK*. You have now finished defining list items, but these will not yet appear in the list. The items will be presented to the user as a choicelist only at runtime.
- Save the edited panel in the resource file using the *File/Save* command.
- Exit the DOORS editor with *File/Exit*. FHS-DOORS is still active with the sample application and will now become visible on the screen.

4.4 Testing an edited panel

The panel displayed on the screen will still reflect its state prior to editing.

► To display the edited panel, select the *Tools/Refresh* command. FHS-DOORS will directly read the resource file and display the edited panel.



The entries in the new list are now also shown.

► Make sure that only the entered items in the list are selectable.

► Now click on *OK* in the panel to display the third panel of the sample application.

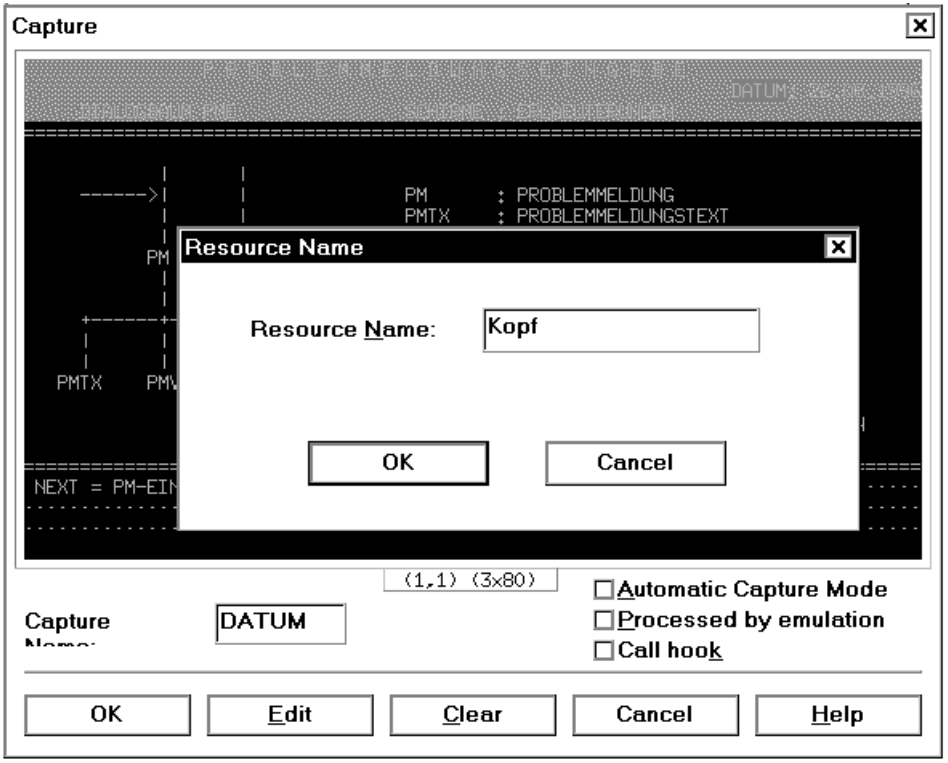
4.5 Creating and editing subpanels

Subpanels enable you to manage the standard and “dynamic” areas of a form separately. Many forms have identical headers and footers with the same design and content and differ only in terms of their work areas. Consequently, you could define a subpanel for the header area and another subpanel for the footer area of such forms so that the underlying templates are displayed on the screen whenever a match is detected during automatic conversion. This also reduces the optimization overhead when converting an application.

- Select the *Tools/Capture* command. The *Capture* and *Capture Management* dialog boxes are displayed on the screen with the alphanumeric representation of the current panel.

4.5.1 Capturing an area for a header panel

- In order to create a capture for a form, you must first select a form identifier. You can do this in the displayed form by using the left mouse button to mark the term DATE, for example. The marked term is again copied into the *Capture Name* field.

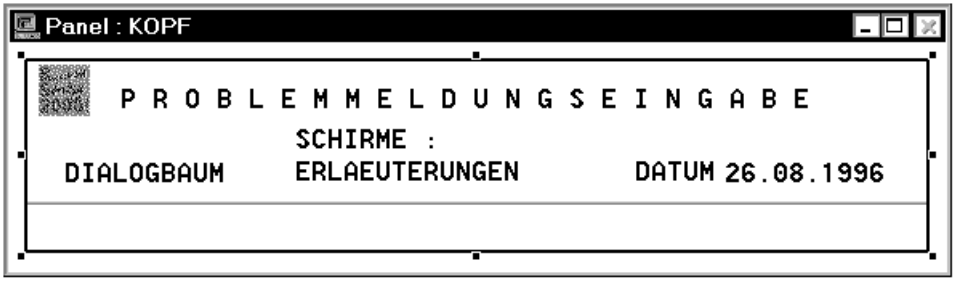


- ► Now use the right mouse button to mark the first three lines of the form. The right mouse button selects the lines in the *Capture* dialog box that are to be included in the subpanel. The marked lines are shown in a different color, and the *Form Name* dialog box appears.
- ► Enter a name for the subpanel, e.g. header, and confirm it with *OK*.
- ► Confirm the *Capture* dialog box with *OK*. The subpanel will now appear in the *Capture List* of the *Capture Management* dialog box. The number 1 in the third column of the list indicates that DATE is a subpanel.
- ► Confirm the *Capture Management* dialog box with *OK*.
- ► Use the *Tools/Generate SDC File* command to create a resource file.
- ► Call the DOORS editor with the *Tools/Edit Resource File* command. This starts the DOORS editor with the current subpanel opened as the input file.

4.5.2 Editing a header panel

In this step, you will insert an image (i.e. a bitmap file) as a small graphical logo in the header panel.

- ► Click on the icon for a bitmap in the toolbar and hold the left mouse button pressed while moving the cursor to the upper left corner of the subpanel on the screen. When you now release the mouse button, a marked empty frame will be inserted at the position of the cursor.
- ► Open the properties window for the bitmap by using the *View/Properties* command. Note that the bitmap must be marked for this purpose.
- ► Select the *Graphical Tuning* tab in the properties window.
- ► Double-click on the *Bitmap Name* property and enter the name of the bitmap to be placed in the frame, i.e. **bstrateg.bmp**, in the list box.
- ► Confirm the file name with *OK*. The bitmap will be placed in the frame and can then be moved to any position or sized as required.



- Exit the DOORS editor with the *File/Exit* command and answer the prompt to save the resource file with *YES*.

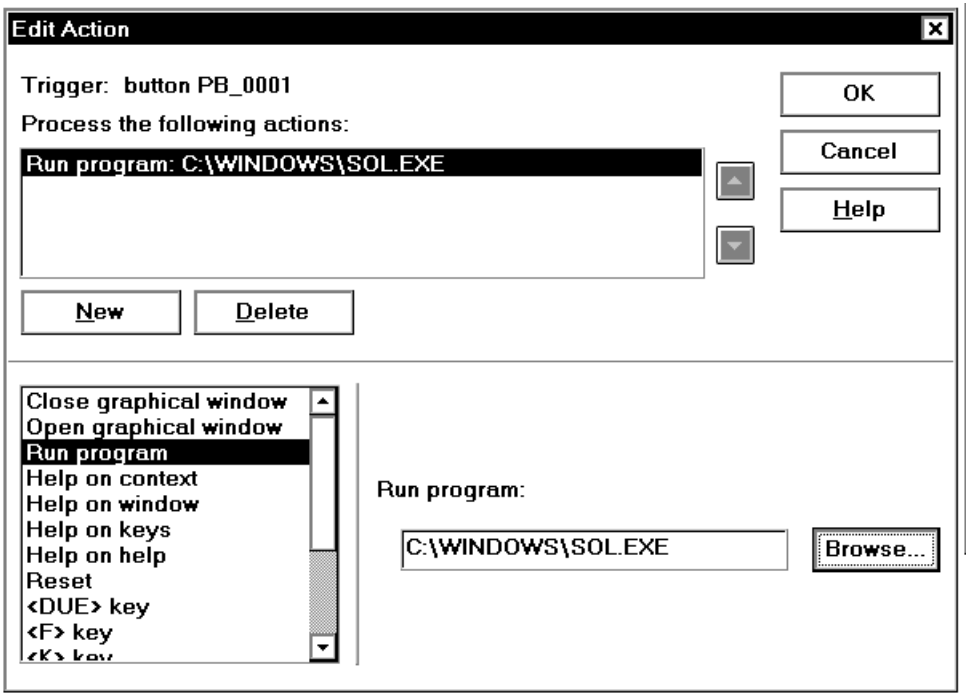
4.5.3 Capturing an area for a footer panel

- Select the *Tools/Capture* command. The *Capture Management* dialog box appears with the alphanumeric representation of the third panel of the sample application still displayed in it.
- Select *New*.
- In order to create a capture for a form, you must first select a form identifier. You can do this in the displayed form by using the left mouse button to mark the term NEXT. The marked term is again copied into the *Capture Name* field.
- Now use the right mouse button to mark the last three lines of the form. The marked lines are shown in a different color, and the *Form Name* dialog box appears.
- Enter a name for the subpanel, e.g. footer, and confirm it with *OK*.
- Confirm the *Capture* dialog box with *OK*. The subpanel will now appear in the *Capture List* of the *Capture Management* dialog box. The number 1 in the third column of the list indicates that NEXT is a subpanel.
- Confirm the *Capture Management* dialog box with *OK*.
- Use the *Tools/Generate SDC File* command to create a resource file.
- Call the DOORS editor with the *Tools/Edit Resource File* command. This starts the DOORS editor with the current subpanel opened as the input file.

4.5.4 Editing a footer panel

In this step, you will add a button and an additional text field to the footer panel.

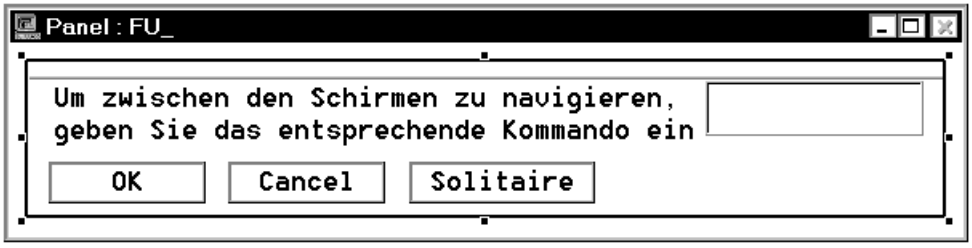
- ► Click on the icon for a button in the toolbar and hold the left mouse button pressed while moving the cursor to a position next to the existing buttons. When you now release the mouse button, a new button will be inserted at the position of the cursor. This button can then be moved or sized as required.
- ► Open the properties window for the bitmap by using the *View/Properties* command. Note that the button must be marked for this purpose.
- ► Select the *Graphical Tuning* tab in the properties window.
- ► Double-click on the *Title* property and enter a label for the button in the entry field of the property window, e.g. **Solitaire**.
- ► Confirm the button label by pressing the ENTER key.
- ► In the property window, select the *Main* tab.
- ► Double-click on *Action*. The *Edit Action* dialog box appears.



- ► Click on *New* to link one of the actions from the list with the new button. This activates the list so that you can select an entry.
- ► Select the *Run Program* entry.
- ► Now click on the *Browse* button to search in the Windows list box for the program file that is to be executed on clicking the button.
- ► Go to the C:\WINDOWS directory and select the program *SOL.EXE*.
- ► Close the list box with *OK*. The name of the program file is copied to the *Run Program* entry field along with the path.
- ► Close the *Edit Actions* dialog box with *OK*.

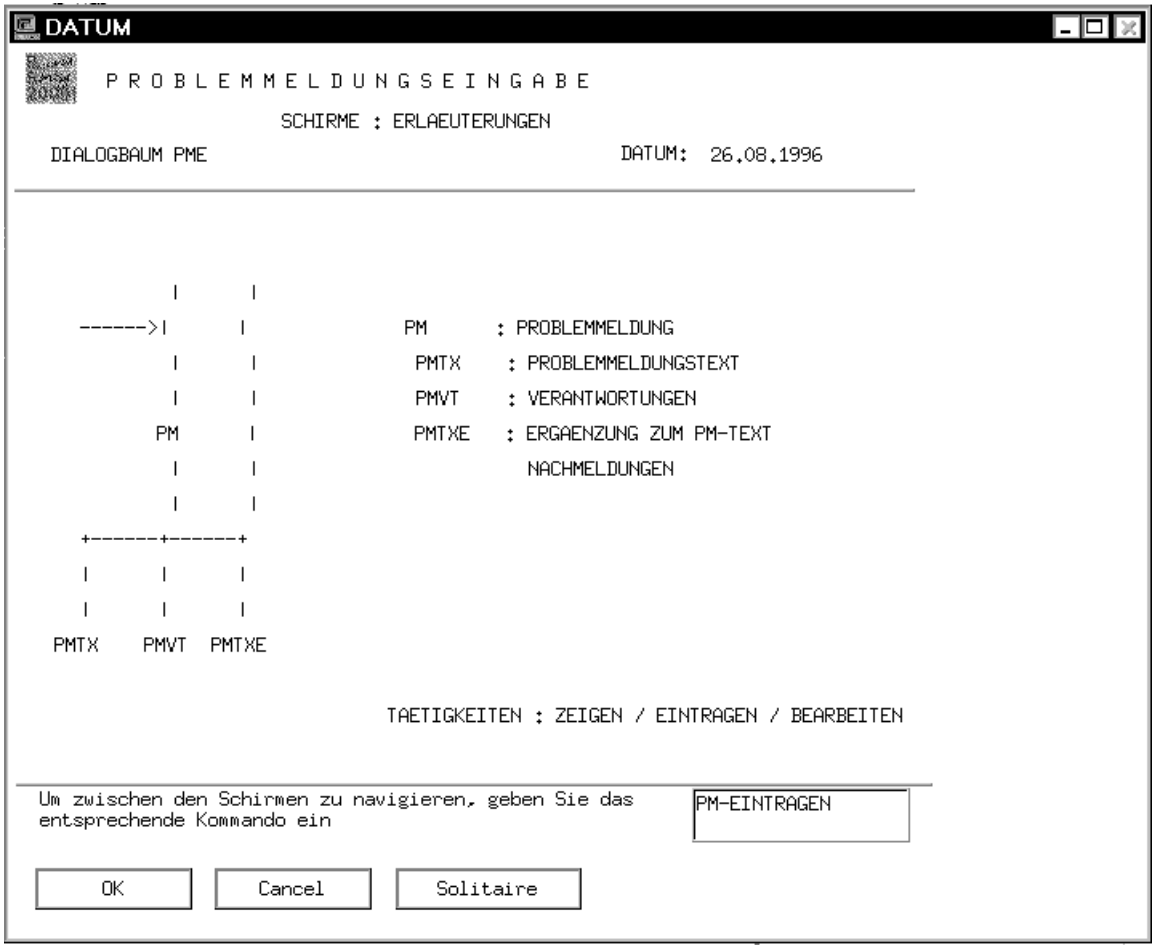
You have now completed the process of adding a new button and associating it with a program to be executed whenever the user clicks on that button at runtime. The next step is to add a text field, but you will first need to create some space for it.

- ► Delete the text field *NEXT* and the second and third line of the entry field.
- ► Then add a new text field by using the icon for a text field and following the usual procedure.
- ► Select the *Graphical Tuning* tab in the properties window for the text field. Note that the text field must be marked for this purpose.
- ► Double-click on the *Title* property and enter the following or any similar text: **Um zwischen den Schirmen zu navigieren, geben Sie das entsprechende Kommando ein**.
- ► Confirm the new label for the text field with the Enter key. The text will appear in the text field.
- ► Adjust the size of the text field to suit the entered text. If desired, you can also change the font or other display attributes of the text field such as font and background colors.



- ► Exit the DOORS editor with the *File/Exit* command and answer the prompt to save the resource file with *YES*. The editor will be closed, and you will see the panel and the primary window of FHS-DOORS again.

► Select the *Tools/Refresh* command to have the optimized subpanel displayed. Depending on your entries, a screen similar to the one below will appear:

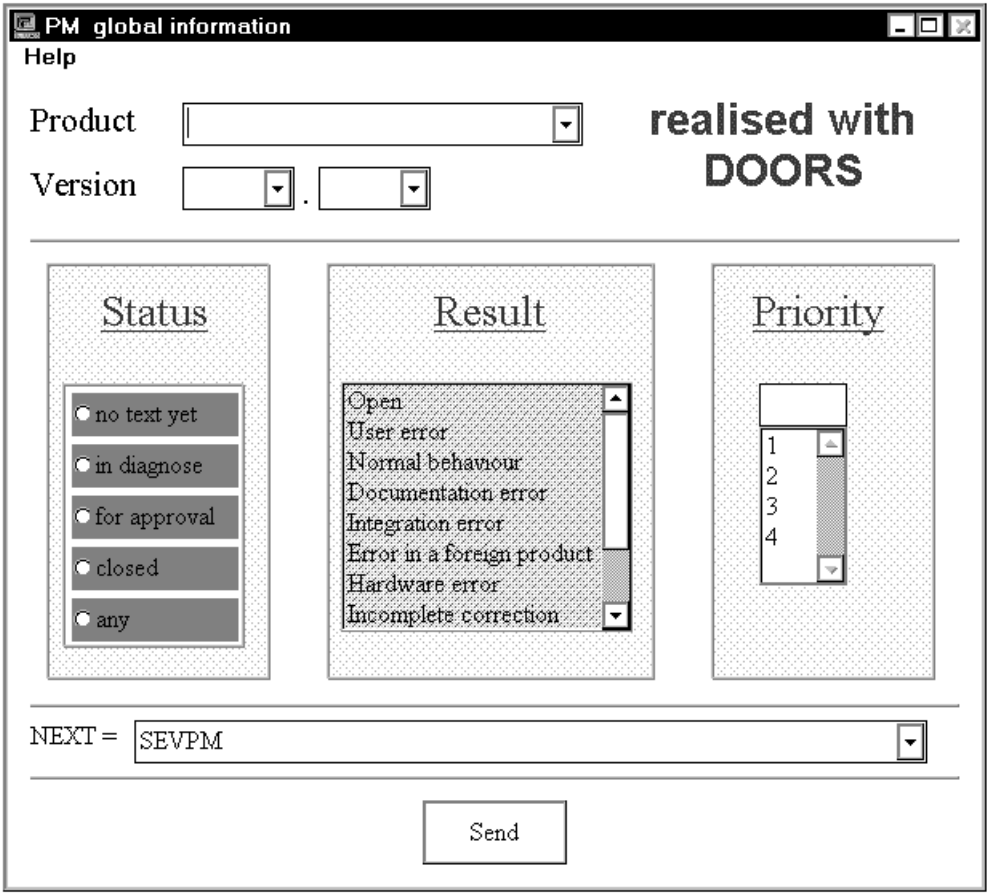


The panel is now shown with the edited subpanel. It consists of three parts, of which only the work area has not been edited. The same subpanels will also be used in other panels whenever the defined areas match.

► Click *OK* to display the fourth panel of the sample application. The footer panel will be reused in the display, since the contents of the two areas match. The contents of the header area differ, so the header of the fourth panel is converted automatically.

- ► Confirm the fourth panel with *OK* to display the fifth panel. The optimized header and footer panels are used here.
- ► Confirm this panel with *OK* as well. The sixth panel of the sample application will appear.
- ► Create a capture for this panel with the name SEGPM_Z; see section “Creating a capture” on page 55 for details.

As soon as you confirm the new capture with *OK* in the *Capture Management* dialog box, the already optimized *PM Global Information* panel will be displayed instead of the automatically converted one.



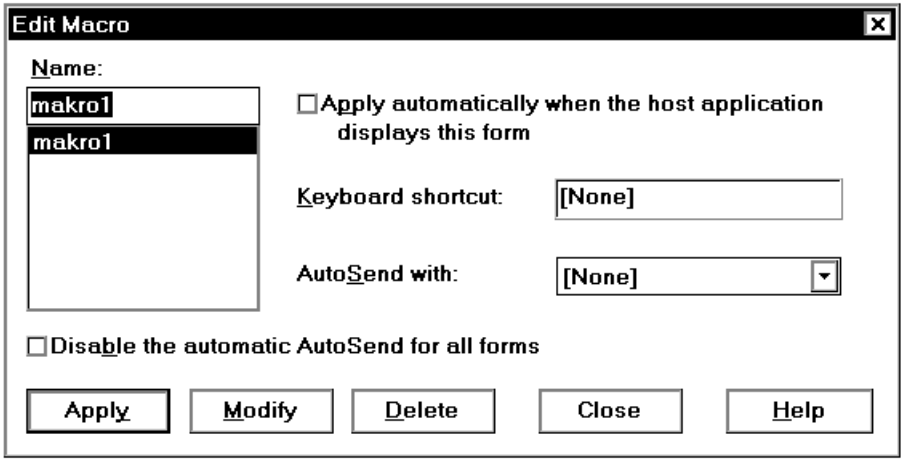
The difference between the original and optimized panels can be seen by checking the emulation window, which retains the original format while executing an application under FHS-DOORS.

4.6 Using macros

Macros allow you to automate the process of making inputs in the panel. This is done by first recording individual steps in the panel and then assigning the macro to the panel. This feature has several advantages, as you will see below.

4.6.1 Creating a macro

- In the optimized *PM Global Information* panel, select the entry *FHS-DOORS-ED* from the *Product* list.
- Select the radio button *any* from the *Status* group.
- In the main menu of FHS-DOORS, select the *Tools/Macro* command. The *Macro* dialog box appears.



- Assign the macro a name, e.g. macro1, and click *Add*. The dialog box is closed. You have now created your first macro, which recorded the last few steps in the optimized panel.
- In the optimized *PM Global Information* panel, select the entry *FHS-DOORS-PC* from the *Product* list.

- ► Select the radio button *for approval* from the *Status* group.
- ► Select the entry *Software Error* from the *Result* list.
- ► Save these three steps in a second macro.

4.6.2 Assigning a macro

- ► Ensure that the mouse cursor is placed on the panel and press the right-hand mouse button. The context menu is opened.
- ► Select the first macro. The settings that were recorded with this macro are made in the panel. Selecting the second macro causes the settings recorded in that macro to be made. You can now switch between the two states in the panel before confirming the entries with *Send*.

4.7 Subpanels

On sending the *PM Global Information* panel with *Send*, the *PM list* secondary window (or subpanel) appears. The emulation window can be used once again to verify what information of the form is contained in this panel. If a form contains too many units of information, it may be worth splitting it up and distributing its content over multiple subpanels.

- ► Close the subpanel with *Cancel*.

A Help panel was also created for the *PM Global Information* panel.

- ► Open the Help panel via the context menu. You can create such subpanels and Help panels for your applications as well.
- ► Close the Help panel with *OK*.
- ► Exit the sample session from the main menu of FHS-DOORS by choosing *Session/Exit*.

5 Panel management and distribution

The following sections describe where FHS-DOORS looks for the panels (resources) for an application at runtime and the mechanisms it provides for panel distribution.

5.1 Resource path and buffer directory

When the forms have been converted to panels (sdc files) with the form converter in BS2000 and when these panels have been optimized with the DOORS editor as required, they must be made available on all PCs on which the host application is to be used under FHS-DOORS.

FHS-DOORS supports the following resources:

– Semantic resources (panels/sdc files)
"abstract" description of the objects and their properties.

– Graphical resources (res files)
graphical description of the panels.

– Binary graphical resources (rbn files) for Dialog Builder as of V2.1.

– Graphical resources with a semantic description in callbacks (FHS-DOORS V1.0).

FHS-DOORS searches through the directories in the following sequence when accessing resources.

1. Resource path specified on the PC.
FHS-DOORS first searches for files with the extension *.sdc*, then for files with the extension *.rbn*, and then for files with the extension *.res*.

2. Buffer directory on the PC.
FHS-DOORS first searches for sdc files with the extension *.sdc*, then for files with the extension *.rbn*, and then for files with the extension *.res*.

3. FHS or FORMPLAG library in BS2000 (for downloading).
FHS-DOORS loads resource files (*sdc* and *res* files) from the forms library in BS2000 on the PC. rbn files and bitmap files cannot be downloaded.

All resources (*sdc*, *res*, *rbn* and bitmap files) are stored in a resource path. To avoid ambiguity among form names, a separate resource directory should be created for each host application. It is essential that all resources of a host application be stored in the same resource path.

When starting up a host application under FHS-DOORS you select the resource path from which the resources for that host application are to be loaded (with the *Session/New* command in FHS-DOORS).

Buffer directory

Downloadable resources (see section “Automatic distribution of panels (downloading)” on page 75ff) are transferred at runtime from the forms library on BS2000 to the PC and stored in a buffer directory. The buffer directory is the *maskbuf* subdirectory of the resource directory selected for the host application, and it is created by FHS-DOORS.

The number of panels that can be downloaded to the buffer directory is governed by the size chosen for the buffer directory and by the size of the panels (sdc files, res files). If you have enough free space, you should make the buffer large enough for the buffer directory to accommodate all the panels belonging to a host application. Then only one download will be required to transfer all the panels to the PC.

You can specify the size of the buffer directory in the *New Session* dialog box (FHS-DOORS *Session/ New* command) or *Edit Session* dialog box (FHS-DOORS *Setup/Session* command). Once the available space in the directory is full, files already in it are overwritten, and the most recently accessed files are the last to be deleted. If the available space has been used up and the *Prompt before mask delete* option has been checked in the *New Session* dialog box, a message is displayed and you can choose either to delete the panels or to abort downloading.

File naming conventions

The DOS/Windows conventions apply.

The file names for the panels are the form names with the extension *.sdc*, *.res* or *.rbn*.

To provide compatibility with the Dialog Builder, the # character in the form name is replaced by _ (underscore) in the panel. If the first character in the form name is a digit, an _ (underscore) is prefixed to the form name in the UIL file.

5.2 Automatic form conversion

If FHS-DOORS cannot access the host application resources because they are errored or simply unavailable to FHS-DOORS, the form description is generated from the form automatically, and a message indicating that the form was generated automatically is displayed. Furthermore, an empty sdc file is created in the buffer directory to notify FHS-DOORS that the resources for this form are not downloadable.

5.3 Automatic distribution of panels (downloading)

The downloading function described here is only supported for FHS/UTM applications and FORMPLAG applications.

The most convenient form distribution mechanism is downloading. The download mechanism transfers the panels (sdc file, res files) directly from the forms library on BS2000 to the PC and stores them in the host application's *maskbuf* buffer directory.

A download is performed whenever a form needs to be displayed and there is no panel (sdc file, res file) for it in the resource directory or the buffer directory (also refer to the notes in section "Important notes on panel distribution" on page 77).

When forms are converted by the form converter in BS2000, the panels which are generated are written directly to the forms library, thus making the converted forms immediately available for downloading.

To ensure that the available resources are always up to date, downloading is also controlled on the basis of a time stamp. Every form can be uniquely identified by its name and its time stamp. The message FHS sends to FHS-DOORS at application runtime includes both the form's name and its time stamp. To construct forms on the PC, FHS-DOORS needs panels with time stamps compatible with those in the forms in the forms library. Downloading is performed even if the panel is available in the resource path or the buffer directory but the time stamp of that panel does not tally with the time stamp of the panel in the forms library.

If the panel has been optimized with Dialog Builder, which means that a res file exists, and the res file is not available in the resource path or the buffer directory, the res file is also downloaded to the PC.

If FHS-DOORS fails to find a res file either in the resource or buffer directory or in the forms library, it generates the layout of the panel on the basis of the sdc file. If the sdc file is not available either, the panel is generated automatically (see section "Automatic form conversion" on page 75).

If the panels have been optimized with the DOORS editor and the optimized panels are to be downloaded to the PC, the latest optimized panels first have to be transferred from the PC where the panels were optimized to BS2000 and added to the forms library as members of type S, typically with the aid of LMS or the BS2000 File Manager.

Download error handling

When downloading panels, FHS-DOORS checks that the panels (sdc files, res files) are in the forms library and are valid. If the panels are not there or are invalid, they are not downloaded but are generated automatically, and an empty file is created in the PC's buffer directory. If the missing panels are subsequently added to the forms library, the empty files must be deleted from the PC's buffer directory, as otherwise the new panels will not be downloaded.

If a system error occurs while the panels in the forms library are being accessed, an error message is displayed on the PC in diagnostic mode. This error message contains:

– the PLAM error code (BS2000)
  This code is added to message class "PLA" to enable information on the error to be displayed with the HELP-MSG command.
– a second error code (if available)
  This code is available if the error occurred in a different system component. The PLAM message identifies the system component in which the error occurred, together with the message class (such as "DMS") where appropriate.
– the code for the PLAM action which could not be executed:
  A = attach, C = check, G = get, O = open, P = posa

5.4 Manual panel distribution

If panels are distributed manually, the latest panels (sdc files, res files) must be made available on all PCs on which interaction with the application under FHS-DOORS occurs. This involves the following steps.

Manual distribution of converted panels

► Using a tool such as LMS or the BS2000 File Manager, extract the panels (sdc files) from the forms library into SAM files if they were not stored in SAM files on conversion.
► Transfer the SAM files to the PC, for example using file transfer or the BS2000 File Manager.
► Store the sdc files in the host application's resource directory.

Manual distribution of optimized panels

- ► If the optimized panels are available on a PC, transfer them to all the PCs on which users are to be able to interact with the host application under FHS-DOORS and store the panels there in the resource directory.
- ► If the optimized panels are available in the forms library, extract them from the forms library into SAM files, transfer them to the PCs and store them in the resource directory.

5.5 Important notes on panel distribution

You should never store optimized panels in the PC's buffer directory, since they could be overwritten by downloaded panels!

1. If you want to optimize panels which are stored in the buffer directory, first copy the panels to the resource directory and optimize the panels in the resource directory.
2. If you want to store optimized panels in the forms library on BS2000, you must add the sdc files and the res files (as necessary) to the forms library as members of type S.
3. If the panels start taking longer to be drawn on the screen, you should check that you have made the buffer directory large enough.
4. Performance is best if the panels are stored in the resource directory on the PC.
5. If forms are to be distributed by the download mechanism, you should specify a directory that does not contain any panels as the resource directory or first delete any panels in it. This will prevent potential problems with incompatible time stamps. Do not forget to transfer the latest panels to BS2000 and add them to the forms library beforehand.
6. Binary panels (rbn files) and bitmap files are not downloadable.
7. A res file generated on the basis of an sdc file does not contain any semantic information and thus cannot be used without the sdc file.

6 Form conversion

The FHS-DOORS-LC form converter (BS2000/OSD) allows you to convert FHS forms for use under FHS-DOORS. The FHS forms which are to be converted must have been created with IFG for FHS (without “fast” formatting) and stored in a PLAM library (forms library) as members of type R.

FHS-DOORS-LC can also convert FORMPLAG forms as of V2.4C. FORMPLAG form conversion works in exactly the same way as FHS form conversion. FORMPLAG and FHS forms can also be stored in a common forms library. FHS-DOORS-LC automatically detects whether it is dealing with a FORMPLAG form or an FHS form. For further information on FORMPLAG form support, refer to the current FORMPLAG release notes.

Form conversion runs on BS2000. The converted forms (panels) are stored in the same forms library by default, but can also be stored in a SAM file via SYSLST. The panels (sdc files) created by the form converter contain all the information needed to run under FHS-DOORS unmodified.

During form conversion, you can specify whether graphical objects are to be generated in the panels and also which objects should be generated, e.g. a pushbutton which simulates the DUE1 key.

If you want to optimize the panels using the DOORS editor or the Dialog Builder editor, you must first extract the sdc files from the library, with LMS for instance (only if the panels were not stored in SAM files immediately), and then transfer them to MS-DOS. If the panels were stored in a SAM file via SYSLST, this file must be split before it is transferred (with the *Tools/ Split Resource Files* command in the DOORS editor).

The FHS-DOORS-LC V3.0 form converter (BS2000/OSD) generates panels (sdc files) for FHS-DOORS V3.0.

To generate panels for FHS-DOORS V1.0, you must use the FHS-DOORS-C form converter (BS2000).

To generate panels for FHS-DOORS V2.0, you must use the FHS-DOORS-LC V1.0 (BS2000) form converter.

If help forms or message forms have been defined for a form, they must be converted along with the forms themselves. Otherwise, the FHS-DOORS message “Help not found” or “Message not found” will be displayed when attempts are subsequently made to access the help system or the message system respectively.

6.1 Generated objects

The form converter generates semantic information for each form and stores it in a panel, which in turn is stored in an sdc file. This sdc file can also be used to generate graphical objects for the panel, either dynamically at runtime or in the form of a res file.

A distinction is made in forms conversion between “text fields” (static objects) and “entry fields”. FHS-DOORS in V2.0 and above also takes account of semantic information already present in the FHS-DE forms in the forms library, for example.

– option buttons generated from single-choice fields
– check boxes generated from multiple-choice fields
– pushbuttons generated from KEY formats
– menus generated from KEY formats and menus
– title bar of the window generated from the panel title (specified in IFG)

Entry fields

The form converter always generates only “entry fields for data input”.

The DOORS editor recognizes the following types of entry fields:

– entry fields for entering data, e.g. text
– entry fields with exactly two options (binary entry fields), such as Yes/No or On/Off
– entry fields with multiple preset options (discrete entry fields), such as Copy/Delete/Move/Save/Quit
– markable entry fields which can be activated or deactivated with the MARK key
– multiple-line entry fields
– alphanumeric object

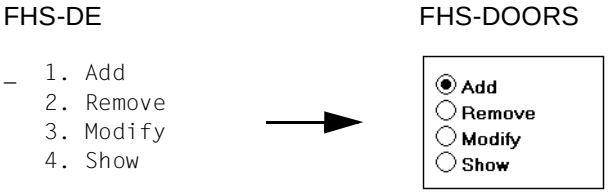
Pushbuttons

Pushbuttons are generated on the basis of the parameters specified for the PUSH-BUTTONS operand of the MODIFY-FHS-DOORS-OPTIONS statement or on the basis of KEY formats. The pushbuttons are generated to appear along the bottom of the panel.

Single-choice fields

Single-choice fields are generated as option buttons. The entry field is deleted, and the selection numbers are replaced by option buttons.

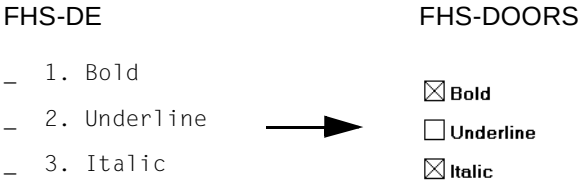
Example



Multiple-choice fields

Multiple-choice fields are generated as check boxes. Each entry field is replaced by a check box.

Example



Messages in the message area

Messages are held in the forms library as members of type R. Messages are converted to form a separate sdc file.

KEY formats

A KEY format (= key mapping table) defines a fixed allocation of function keys to commands. The function keys in the KEY formats can be generated as pushbuttons along the bottom of the panel or as pull-down menus (menu title key list) along the top of the panel.

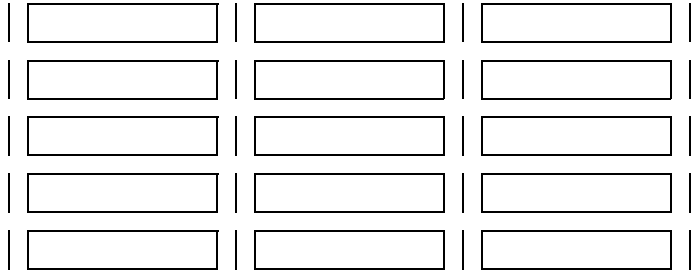
List boxes

List boxes are generated as entry boxes and static objects. The same entry boxes and static objects are generated for every item on the list, but the object names and the semantics differ. In addition, an “end-of-list” message is generated. If the number of list items sent at application runtime is less than the number defined, the “superfluous” fields are disabled and the end-of-list message is displayed.

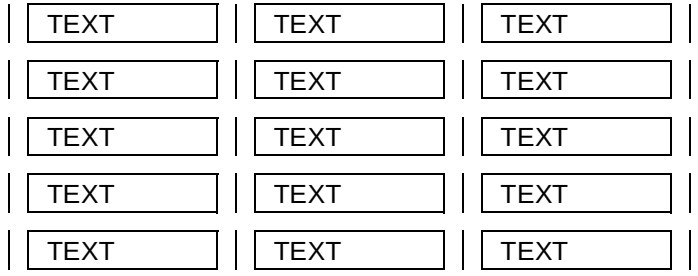
Example

The form defines five list items and the end-of-list message “END OF FILE”. Each item consists of three entry boxes and four static objects in the form of “|”:

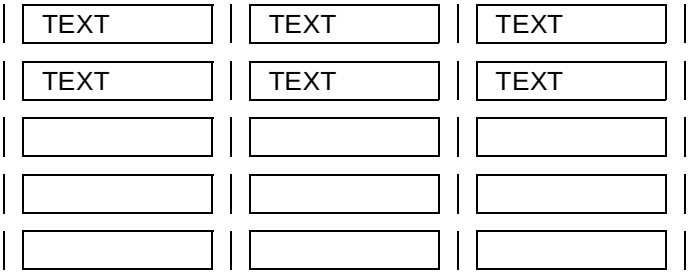
The form converter generates the following objects:



If the application sends data for five items, all the objects are used. The end-of-list message is not displayed:



If the application sends data for only two items, the last three list objects are disabled and the end-of-list message is displayed:



Help boxes

A help window is generated to handle help forms. Under FHS-DOORS there is no command line in the help window.

Only one help window is generated. It is used to displayed all the help texts. The *OK* pushbutton is used to close the window.

Message boxes

FHS-DOORS directly outputs message texts and parameters in message boxes. A single message window is generated to handle message output. All messages are displayed in this window.

Menus

Menus in FHS-DE forms are generated as pull-down menus. In addition, an action rule is automatically generated for each menu command.

Menus can also be generated on the basis of the parameters specified for the MENU operand of the MODIFY-FHS-DOORS-OPTIONS or on the basis of KEY formats. Menus are generated to appear along the top of the panel.

6.2 Starting FHS-DOORS-LC

The FHS-DOORS-LC form converter (BS2000/OSD) is started by the command below. The syntax description is given in the appendix as of page 123ff or in the manual “Introductory Guide to the SDF Dialog Interface” [10].

START-FHS-DOORS-LC
VERSION=**<u>*STD</u>** / <product-version 6..10> / product-version 4..8 without-corr /   <product-version 3..7 without-man> **,MONJV = <u>*NONE</u>** / <full-filename 1..54 without-gen-vers> **,CPU-LIMIT = <u>JOB-REST</u>** / <integer 1..32767 *seconds*>

VERSION =
Version of the form converter.

VERSION = <u>*STD</u>
Before the form converter is called, the SELECT-PRODUCT-VERSION command is issued (in system mode) to set the version. The version set is used as the default version.

VERSION = <product-version 6..10>
Explicit version specification.

VERSION = <product-version 4..8 without-corr>
Version name without correction status.

VERSION = <product-version 3..7 without-nam>
Version name without release or correction status.

MONJV = <u>*NONE</u> / <full-filename 1..54 without-gen-vers>
Job variable to monitor the form converter run. The job variable must already be cataloged (only for users with the software product Job Variables).

During the form converter run, the system sets the job variables to the following values:

Value	Meaning of value assignment
$R	Form converter running
$T	Form converter terminated normally
$A	Form converter terminated abnormally

MONJV = <u>*NONE</u>
No job variable is used for monitoring.

CPU-LIMIT = JOB-REST / <integer 1..32767 *seconds*>
Maximum CPU time allowed for the form converter during execution. In interactive mode, the user is informed by the system if this time is exceeded; in batch mode the operation is terminated.

CPU-LIMIT = JOB-REST
If the operand CPU-LIMIT=STD has been specified in the SET-LOGON-PARAMETERS command, there is no time limit on the program.
If the operand CPU-LIMIT=t has been specified in the SET-LOGON-PARAMETERS command, the value defined during system generation is used as the time limit for the form converter run.

Terminating FHS-DOORS-LC

The following statement is used to terminate the FHS-DOORS-LC V3.0 (BS2000/OSD) form converter:

```
//END
```

6.3 FHS-DOORS-LC (BS2000/OSD) statements

Once the form converter has been started, the following statements are available for converting forms:):

Statement	Meaning
ASSIGN-LIBRARY	Assign an (FHS) library
CONVERT-MASKS	Convert the forms
END	Terminate FHS-DOORS-LC
MODIFY-FHS-DOORS-OPTIONS	Specify conversion options
STEP	Define the restart point in a sequence of statements

Table 5: Statements of the form converter

These statements are described together with their operands in alphabetical order below.

SDF standard statements

The SDF standard statements MODIFY-SDF-OPTIONS, REMARK, RESTORE-SDF-INPUT, SHOW-SDF-OPTIONS, WRITE-TEXT and SHOW-INPUT-HISTORY are offered throughout the system for programs which read in their statements using SDF. Consequently, they are also available in the form converter. A description of the standard statements is provided in the manual “Introductory Guide to the SDF Dialog Interface” [10].

ASSIGN-LIBRARY - Assign (FHS/FORMPLAG) library

The ASSIGN-LIBRARY statement enables you to specify the forms library containing the forms which are to be converted. The forms are read from this library, and the converted forms (panels) are written back to it. If there is already a forms library open, it is closed, even if the new library cannot be opened.

The forms library is a PLAM library. The forms are type-R members of this PLAM library. The panels are written back to the library (with the suffix *.sdc*) as members of type S.

ASSIGN-LIBRARY
TO-FILE = ***LINK /** <full-filename 1..54>

TO-FILE =
Assigns the forms library.

TO-FILE = *LINK
The forms library is assigned to the file link name DOORS2L via a SET-FILE-LINK command.

TO-FILE = <full-filename 1..54>
Name of the forms library. Any forms library assigned previously via SET-FILE-LINK is ignored.

CONVERT-MASKS - Convert forms

The CONVERT-MASKS statement enables you to convert forms to panels (sdc files). The panels are written back to the forms library as members of type S.

CONVERT-MASKS
MASK-ID = ***ALL** / ***FILE**(...) / list-poss(2000): <full-filename 1..32 without-cat-user-gen-vers with-wild> / <alphanum-name 1..32 with-wild> ***FILE**(...) **FILE-NAME =** <full-filename 1..54>

MASK-ID =
Specifies the forms to be converted.

MASK-ID = *ALL
All forms of the forms library assigned with ASSIGN-LIBRARY are converted.

MASK-ID = *FILE(...)
The forms to be converted can also be written into a file in list form. Each record of this SAM or ISAM file contains the full name of a form.

FILE-NAME = <full-filename 1..54>
Name of the file which contains the list of forms to be converted.

MASK-ID = list-poss(2000):
<full-filename 1..32 without-cat-user-gen-vers with-wild> /
<alphanum-name 1..32 with-wild>
Names of the forms to be converted in the FHS library assigned with ASSIGN-LIBRARY. If wildcards are used, all forms matching the specified wildcard syntax are converted.

You can convert a maximum of 2000 forms with one CONVERT-MASKS statement, even if wildcard syntax is used. Each form is converted only once, even if several wildcards are used in a statement and a form matches more than one of them.

END - Terminate FHS-DOORS-LC (BS2000/OSD)

The END statement terminates the FHS-DOORS-LC form converter (BS2000/OSD). All open files are closed.

END

MODIFY-FHS-DOORS-OPTIONS - Specify conversion options

The MODIFY-FHS-DOORS-OPTIONS statement enables you to specify extra options which are to apply when the forms are converted. These allow you to:

– list the pushbuttons and menus which are to be inserted in the panels

– generate KEY formats as pushbuttons or menus

– split up static texts

– do native language conversion

– send output to SYSLST

MODIFY-FHS-DOORS-OPTIONS

PUSH-BUTTONS = *UNCHANGED / ***NONE** / list-poss(40): ***PARAMETERS**(...)
  ***PARAMETERS**(...)
    LABEL = <alphanum-name 1..12> / <c-string_1..12 with-low>
    ,KEY-NAME = *DUE1 / *DUE2 / *F1 / *F2 / *F3 / *F4 / *F5 / *F6 / *F7 / *F8 / *F9 / *F10 / *F11 / *F12 / *F13 / *F14 / *F15 / *F16 / *F17 / *F18 / *F19 / *F20 / *F21 / *F22 / *F23 / *F24 / *K1 / *K2 / *K3 / *K4 / *K5 / *K6 / *K7 / *K8 / *K9 / *K10 / *K11 / *K12 / *K13 / *K14
,MENU = *UNCHANGED / ***NONE** / list-poss(40): ***PARAMETERS**(...)
  ***PARAMETERS**(...)
    LABEL = <alphanum-name 1..12> / <c-string_1..12 with-low>
    ,KEY-NAME = *DUE1 / *DUE2 / *F1 / *F2 / *F3 / *F4 / *F5 / *F6 / *F7 / *F8 / *F9 / *F10 / *F11 / *F12 / *F13 / *F14 / *F15 / *F16 / *F17 / *F18 / *F19 / *F20 / *F21 / *F22 / *F23 / *F24 / *K1 / *K2 / *K3 / *K4 / *K5 / *K6 / *K7 / *K8 / *K9 / *K10 / *K11 / *K12 / *K13 / *K14
,KEY-LIST = *UNCHANGED / ***NO** / ***PUSH-BUTTON / *MENU**
,TEXT-SEPARATORS =*UNCHANGED / *NONE / ***PARAMETERS(**...**)**
  ***PARAMETERS(**...**)**
    BLANKS = <integer_1..80>
,LANGUAGE = *UNCHANGED / *NONE / *DANISH / *ENGLISH-UK / *ENGLISH-USA / *FRENCH / *FRENCH-BELGIAN / *GERMAN / *INTERNATIONAL / *ITALIAN / *NORWEGIAN / *SPANISH / *SWEDISH / *SWISS
,SYSLST-OUTPUT = *UNCHANGED / *NO / *YES

PUSH-BUTTONS =
Specifies the pushbuttons which are to be added to the panels.

PUSH-BUTTONS = *UNCHANGED
The specifications from the preceding MODIFY-FHS-DOORS-OPTIONS statement remain unchanged. The default value is *NONE.

PUSH-BUTTONS = *NONE
No pushbuttons are added to the panels.

PUSH-BUTTONS = list-poss(40): *PARAMETERS(...)
Specifies which pushbuttons are to be added to the panels.

LABEL = <alphanum-name 1..12> / <c-string_1..12 with-low>
Text (title) of the pushbutton.

KEY-NAME = *DUE1 / *DUE2 / *F1 / *F2 / *F3 / *F4 / *F5 / *F6 / *F7 / *F8 / *F9 / *F10 / *F11 / *F12 / *F13 / *F14 / *F15 / *F16 / *F17 / *F18 / *F19 / *F20 / *F21 / *F22 / *F23 / *F24 / *K1 / *K2 / *K3 / *K4 / *K5 / *K6 / *K7 / *K8 / *K9 / *K10 / *K11 / *K12 / *K13 / *K14
The pushbutton simulates one of the keys DUE1, DUE2, F1 ... F24, K1 ... K14 .

MENU =
Specifies the menus/menu items which are to be added to the panels.

MENU = *UNCHANGED
The specifications from the preceding MODIFY-FHS-DOORS-OPTIONS statement remain unchanged. The default value is *NONE.

MENU = *NONE
No menus are added to the panels.

MENU = list-poss(40): *PARAMETERS(...)
Specifies which menus/menu items are to be added to the panels.

LABEL = <alphanum-name 1..12> / <c-string_1..12 with-low>
Text (title) of the menu item.

KEY-NAME = *DUE1 / *DUE2 / *F1 / *F2 / *F3 / *F4 / *F5 / *F6 / *F7 / *F8 / *F9 / *F10 / *F11 / *F12 / *F13 / *F14 / *F15 / *F16 / *F17 / *F18 / *F19 / *F20 / *F21 / *F22 / *F23 / *F24 / *K1 / *K2 / *K3 / *K4 / *K5 / *K6 / *K7 / *K8 / *K9 / *K10 / *K11 / *K12 / *K13 / *K14
The menu item simulates one of the keys DUE1, DUE2, F1 ... F24, K1 ... K14 .

KEY-LIST = *UNCHANGED / *NO / *PUSH-BUTTON / *MENU
Applies to FHS-DE forms only.
If *PUSH-BUTTON is specified, a pushbutton is generated for each entry in the KEY format of an FHS-DE form.
If *MENU is specified, a “KEY-List” menu is generated. A menu item is generated for each entry in the KEY format of an FHS-DE form.
If the form is not an FHS-DE form, the operand is ignored. The default value is *PUSH-BUTTON.

TEXT-SEPARATORS =
Defines the separator characters in static texts. If a static text (such as a field title) contains separator characters, the text is split up into multiple objects. This makes it possible to adapt the texts at the conversion stage to match the character set which will subsequently be used.

Static texts in single-choice and multiple-choice fields cannot be split up.

TEXT-SEPARATORS = *UNCHANGED
The text separators specified in the preceding MODIFY-FHS-DOORS-OPTIONS statement remain unchanged. The default value is *NONE.

TEXT-SEPARATORS = *NONE
Static texts are not split up.

TEXT-SEPARATORS = *PARAMETERS(...)
Static texts are split up.

BLANKS = <integer_1..80>
Static texts are split up if they are separated by at least the number of blanks specified in BLANKS.

Example

If the text separator specification is:

```
TEXT-SEPARATORS=*PARAMETERS(BLANK=2)
```

```
'IName/surname....!Street/number...!ZIP code/town.........I'
```

the text is split up like this:

```
            'IName/surname'
                                '!Street/number'
                                                    '!ZIP code/town'
                                                                        'I'
```

LANGUAGE =
Defines the language for which form conversion is to be performed. The language can be specified for FHS forms only, not for FORMPLAG forms.

LANGUAGE = *UNCHANGED
The language specified in the preceding MODIFY-FHS-DOORS-OPTIONS statement remain unchanged. The default value is *INTERNATIONAL.

LANGUAGE = *NONE
The form texts are used as they are. No country-specific character conversion is performed. *NONE should always be used when converting 8-bit forms, because 8-bit forms already include the country-specific characters.

LANGUAGE = *DANISH / *ENGLISH-UK / *ENGLISH-USA /
***FRENCH / *FRENCH-BELGIAN / *GERMAN / *INTERNATIONAL / *ITALIAN /**
***NORWEGIAN / *SPANISH / *SWEDISH / *SWISS**
A character set table is used to convert the form texts to country-specific characters.

SYSLST-OUTPUT = *UNCHANGED / *NO / *YES
Specifies whether the panels are also to be written to the SAM file assigned to SYSLST. If you specify *NO, the panels are only written to the forms library.
If you specify *YES, the panels are also written to the SAM file assigned to SYSLST. A SAM file must already have been assigned to SYSLST with the ASSIGN-SYSLST command. All the panels are then written to a SAM file. This SAM file can then be transferred to the PCs e.g. with file transfer, where it must be split into separate sdc files with FHS-DOORS (using the *Tools/Split Resource Files* command in the DOORS editor).

STEP - Define restart point

The STEP statement enables you to define a restart point for error recovery (spinoff mechanism) in a sequence of statements.

If an error occurs during syntax analysis of the form converter statements, the subsequent statements are skipped until STEP or END is reached. The statement following STEP or END is read and processed.

Error recovery is triggered only for syntax errors. That means that if, for example, you assign a non-existent FHS library in ASSIGN-LIBRARY, no branch is made to the next STEP or END statement. The next statement is processed.

The STEP statement may only be used in procedures and ENTER jobs.

STEP

6.4 Additional notes

– The BLS messages relating to the modules IDHSLNG, IDHSCHD, IDHSCHC and the corresponding language variants, only have one meaning for FHS-DE forms. Otherwise, these messages can be ignored.

– The form converter replaces all # (hash) characters in the identifiers for graphical objects with the _ (underscore) character.

7 Interfaces

FHS-DOORS offers you the option of creating and linking libraries for form recognition that are specially customized to your application as well as an option to have data from your host application processed directly by other programs on a PC via the OLE object “Auto9750.Connection”. These two interfaces are described below.

7.1 Library for form recognition

When you run your application, FHS-DOORS usually recognizes forms sent by the host application automatically and looks in the resource directory on the PC or in a form library in BS2000 for an appropriate panel to be used in the display. If now predefined panel is available, the forms are automatically converted and displayed on the PC.

In some cases, however, it may be expedient to develop a separate algorithm for form recognition, especially if your application processes a lot of dynamic data, i.e. when querying a database. FHS-DOORS provides you with a skeleton program for this purpose as well as some predefined functions with which this skeleton program can be processed.

Prerequisite

In order to develop your own dynamic library for form recognition, you will need a C++ compiler to edit and compile the skeleton program.

Procedure

- Switch to the installation directory of FHS-DOORS (default: *C:\FHSDOORS*) and then to the subdirectory *fhsd/hook*. This directory contains the skeleton program *Hook_gen.cpp*, which is written in C++, and the following files to create a dynamic library:
  - *Hook_gen.h* (header file with predefined functions)
  - *Hookdata.h* (header file with data structures)
  - *Pandata.h* (header file with data structures)
  - *Hook_gen.def* (definition file)
  - *Hook_gen.mak* (makefile with a list of these files)

- Extend the *GenerateRow()* function in the skeleton program with the predefined functions and adapt it to your requirements.
- Compile the program with a C++ compiler into a dynamic library (DLL).
- When you create a new capture, activate the option *Call DLL* in the *Tools/Capture* dialog box.
- Specify the library in the *DLL for Form Recognition* entry field of the *Capture Management* dialog box.
- Alternatively, you could also enter the name of the library directly in the parameter file for your application. This can be done by entering the following line under the *[misc]* section in the *.drs* file:

  HookDllPath="library-name**"**

7.1.1 The GenerateRow() function

The *GenerateRow()* function is defined in the *Hook_gen.cpp* skeleton program as follows:

```
void GenerateRow(PanelHookData *pData, char *pszText, char *pszAtts);
```

**pData* Points to an array with form descriptions.

**pszText* Points to a vector containing the text of the current line in the form.

**pszAtts* Points to a vector containing the attributes of the current line in the form.

You can process any line in the form with the predefined functions.

Macros

The following test functions can be logically combined with the *AND* and *OR* macros. In addition, you can navigate within the form by using the macros below:

PREV_LINE Points to the previous line in the form, relative to the current line.

THIS_LINE Points to the current line.

NEXT_LINE Points to the next line in the form, relative to the current line.

You can use these macros to identify a line position in the form in the functions described below just like pure integer values. The macros always refer to line specifications relative to the current line in the form.

Example

If the current line in the form is line 5, for example, you can switch to line 4 with the *PREV_LINE* macro. This could also be achieved by specifying -1.

Make sure that the macros or integer specifications do not address lines outside the form. The maximum number of lines in the form is restricted by the screen size. The line count begins with 1.

Variables

If you need your own variables to specify dependencies between different lines in the form, you should define such variables as static, since the line values would otherwise be overwritten. Furthermore, all such variables should only be locally accessible for the corresponding *CAPTURE_IS()* function and be initialized on processing the first line.

7.1.2 Predefined functions for GenerateRow()

The predefined functions for the *GenerateRow()* function are subdivided into

– test functions

– processing functions

Test functions

All test functions return the value TRUE or FALSE as a result.

`BOOL ATTRIBUTE_IS(int x, char c)`
Tests whether the attribute at position *x* of the current line has the value *c*.

`BOOL CAPTURE_IS(string name)`
Tests whether the name of the currently processed capture is *name*.

`BOOL COLUMN_IS(int x, char c)`
Tests whether the character at position *x* of the current line has the value *c*.

`BOOL COLUMNS_ARE(int x1, int x2, string s)`
Tests whether the string between positions *x1* and *x2* of the current line matches string *s*.

`BOOL EMPTY(int n)`
Tests whether line *n* is empty, where *n* is counted relative to the current line.

`BOOL LINE_BETWEEN(int n1, int n2)`
Tests whether the current line lies between lines *n1* and *n2*.

`BOOL LINE_IS(int n)`
Tests whether the current line is line *n* of the form.

`BOOL NOT_EMPTY(int n)`
Tests whether line *n* contains one or more characters, where *n* is counted relative to the current line.

`BOOL NULL_LINE(int n)`
Tests whether line *n* has the value 0, where *n* is counted relative to the current line.

`BOOL NOT_NULL(int n)`
Tests whether line *n* contains a non-zero value, where *n* is counted relative to the current line.

`BOOL XCOLUMN_IS(int n, int x, char c)`
Tests whether the character at position *x* in line *n* is the character *c*.

`BOOL XCOLUMN_ARE(int n, int x1, int x2, string c)`
Tests whether the character between positions *x1* and *x2* in line *n* matches the string *s*.

Processing functions

`char *GET_DSP_ATTS(int n)`
Returns a pointer to the display attribute of line *n* as a result, where *n* is counted relative to the current line.

`char *GET_SEM_ATTS(int n)`
Returns a pointer to the semantic attribute of line *n* as a result, where *n* is counted relative to the current line.

`char *GET_TEXT(int n)`

Returns a pointer to the text of line *n* as a result, where *n* is counted relative to the current line.

`void PANEL_SET(string name, int n)`

This function is used to manage subpanels and to fill the array *pData*, which is defined in the *GenrateRow()* function. It determines the subpanel with *name* and *n* number of lines that is required for displaying and processing the next line. If the interaction between subpanels is involved, you should make sure that a resource file has been assigned with the *PANEL_SET()* function for every line of the form. Without this assignment, nothing will be displayed on the screen for the line, and no action will be executed.

Example

```
void GenerateRow(...)
{
if (CAPTURE_IS("capture1))
   {
   if (LINE_IS(n))
      {
      if (COLUMNS_ARE(...))
         PANEL_SET("panel1",3);
      }
   else if (LINE_BETWEEN(n1, n2))
      {
      if (COLUMN_IS(...))
         PANEL_SET("panel2",1)
      else
         PANEL_SET("panel3", 1)
      }
   }
   else if (CAPTURE_IS("capture2"))
      {
      ...
      }
   return;
}
```

7.2 OLE automation

OLE allows FHS-DOORS to offer you an interface which gives you simple access to the host application at program level.

This provides you, for instance, with a simple method of inserting fields from BS2000 forms in your Excel spreadsheets or to control work processes with a Visual Basic script or a C++ program.

What is OLE automation?

OLE automation allows you to process the OLE automation objects provided with FHS-DOORS from other programs. This means that you can write a program outside FHS-DOORS (e.g. an Excel V5.0 script) which accesses the fields in a host application.

The OLE automation objects are instances of classes. Each object provides properties and methods. Properties can be read and set. Methods are functions which can be called from outside the object.

The OLE automation controller includes actions such as creation of the object, the setting of properties and the invocation of methods.

OLE automation ensures that parameters are passed when methods are called across different applications and that the correct values are returned.

FHS-DOORS OLE automation objects

FHS-DOORS makes one OLE automation object of the type "Auto9750.Connection" available.

You can use "Auto9750.Connection" in any macro language or programming language which provides support for OLE automation. This includes in particular the following languages:

– Visual C++

– Visual Basic

– Visual Basic for Applications (programming interface for Excel V5.0, for instance).

7.2.1 The OLE automation object “Auto9750.Connection”

The OLE automation object "Auto9750.Connection" represents a BS2000 connection to a host application. It offers you methods

– for defining a connection to the BS2000 host and opening and closing this connection

– for manipulating BS2000 forms directly, either by entering data in entry fields or by selecting fields

– for fetching data from BS2000 forms.

If you set the connection properties for the Auto9750.Connection object, you define the name of the connection (identifier), the emulation to be used and the emulation parameters.

If a method returns "TRUE", this does not necessarily mean that the action has been successfully executed on the host computer, but only that the request has been accepted by the emulation used.

There are three properties for this object: *ConnectionIdentifier*, *ConnectionParameters* and *ConnectionCmdLine*. These three properties allow direct access to the parameters of the SetConnection() method.

All methods which require a connection generate exceptions (with a value of 1000) if no connection has been established. This exception can be intercepted using the ON ERROR statement in Visual Basic, for instance.

Registration

OLE automation objects must be “registered” to allow them to be accessed from other applications.

The FHS-DOORS installation program registers the FHS-DOORS OLE automation objects automatically. You can repeat the registration of the FHS-DOORS OLE automation objects (e.g. if the Registry is faulty or has been deleted inadvertently) using the OLE automation server *AUTO9750.EXE*.

Restrictions

OLE automation provides the option of having the OLE server inform the OLE client when an event occurs (e.g. the arrival of a new screen). This means that the Visual Basic for Applications script must be written in such a way that screen navigation is implemented with the WaitUntil(...) or WaitUntilCondition() method.

Type libraries

The FHS-DOORS OLE automation objects are declared in the following files:

– A9750DEU.TLB (German)

– A9750ENG.TLB (English)

These files can be read by compilers (e.g. Visual C++) and browsers (e.g. OLE2VIEW) to allow the objects to be displayed and accessed.

7.2.2 Methods for “Auto9750.Connection”

`BOOL ActivateEmulation()`

Activates the window in which the connection to the host application is displayed. Returns TRUE if the connection exists and the window could be activated; otherwise, FALSE.

`BOOL CloseConnection()`

Closes the connection.
Returns FALSE if the connection could not be closed or if the connection was not open. Returns TRUE in all other cases.

`BOOL DefineCondition(short ConditionIdf, short Line, short Column, LPCSTR String)`

Defines a condition for the WaitUntilCondition() method defined immediately afterwards. The condition has a numeric identifier (*ConditionIdf*), a *Line/Column* coordinate and a string (*String*).
ConditionIdf must be greater than 0.
Line and *Column* start at 1.
If 0 is specified for *Line*, the condition applies to all lines.
Returns TRUE if the condition is accepted and stored in memory. Returns FALSE in all other cases (*ConditionIfd* is less than or equal to 0, a condition with the same *ConditionIdf* has already been defined, invalid *Line/Column*).

`BSTR GetArea(short Line, short Column, short Length)`

Reads a number of characters defined by *Length* from the position on the emulation screen defined by *Line/Column* and returns these characters in a string. If the end of the line is reached, the characters in the following line are read until the number of lines defined by *Length* have been read or until the end of the screen is reached.
$1 \leq Line \leq 24$ and $1 \leq Column \leq 80$.
An empty string is returned if no connection is open or if the *Line/Column* specifications are invalid or if it was not possible to read the emulation window.

`short GetCursorColumn()`

Returns the column of the current cursor position in the form. Column numbering begins with 1.

`short GetCursorLine()`

Returns the line of the current cursor position in the form. Line numbering begins with 1.

`BOOL IsConnected()`

Returns TRUE if a connection is open. Returns FALSE in all other cases.

`BOOL IsScreenModified()`

Returns TRUE if the contents of the emulation window have been changed since the last time the GetArea() method was called. Returns FALSE in all other cases.

`BOOL Mark(short Line, short Column)`

Selects (marks) the field at the screen position in the emulation window defined by *Line/Column*.
Line and *Column* start at 1.
If 0 is specified for *Line* or *Column*, the cursor is not moved in the emulation window. The field at the current position is then selected.
Returns TRUE if the field was selected successfully. FALSE is returned in all other cases (no connection, invalid *Line/Column*, errors generated by the emulation).

`BOOL OpenConnection()`

Establishes a connection previously defined with SetConnection().
If the parameters and/or the program required for establishing a connection have not been defined, a standard dialog box is displayed by the dispatcher and the user can enter the missing values.
OpenConnection uses the connection with the specified identifier if this connection has already been established. If not, it establishes a new connection.
Returns FALSE if a connection for an object of the type "Auto9750.Connection" has already been established or if the connection could not be established. In all other cases, TRUE is returned.

`BOOL SendScreen(LPCSTR Key)`

Sends the screen off to the host application using the key specified by *Key*.
The following values are permitted for *Key*: DUE1, DUE2, F01 ... F24 and K01 ... K14.
Returns TRUE if the screen was sent successfully. Returns FALSE in all other cases (no connection, invalid *Key*, errors generated by the emulation).

`BOOL SetArea(short Line, short Column, LPCSTR Text)`

Writes the string defined by *Text* to the screen location in the emulation window defined by *Line/Column*. Any existing data at this location is overwritten by *Text*.
Line and *Column* start at 1.
If 0 is specified for *Line* or *Column*, the cursor is not moved in the emulation window. *Text* is then written at the current position.
Returns TRUE if *Text* is written successfully. FALSE is returned in all other cases (no connection, empty *Text*, invalid *Line/Column*, errors generated by the emulation).

`void SetConnection(LPCSTR Name, LPCSTR Parameters, LPCSTR Program)`

Sets the parameters required for a connection. SetConnection only defines the connection and does not yet establish it. Use the OpenConnection() method to establish the connection.
Name: Identifier for the connection
Parameters: Parameters for the connection; name of the parameter file
Program: File name of the emulation program to be used

`BOOL SetCursorPos(short Line, short Column)`
: Sets the cursor to the specified position in the form, where line and column numbering begins with 1. Returns TRUE if the cursor could be successfully positioned; otherwise, FALSE (e.g. for an invalid *Line*/*Column* or a disconnect).

`void SetTimer(short Limit)`
: Defines a timeout (in seconds) indicating the latest time after which a WaitUntil/WaitUntilCondition method should return a result.
The preset value for *Limit* is 60 seconds.

`void Wait(short Seconds)`
: Interrupts the script/programs for the period specified by *Seconds*.

`BOOL WaitUntil(short Line, short Column, LPCSTR ComparisonValue)`
: Waits until the screen area in the emulation window defined by *Line/Column* corresponds to the string given in *ComparisonValue*.
Line and *Column* start at 1.
If 0 is specified for *Line*, all lines are compared.
A result is only returned if the condition is fulfilled or if an error occurs.
Returns TRUE if the condition is fulfilled. Returns FALSE in all other cases (invalid *Line/Column* or TIMEOUT - see also SetTimer() method).

`short WaitUntilCondition()`
: Waits until a condition defined previously with the DefineCondition() method is fulfilled.
Returns -1 of no condition has been fulfilled.
Returns 0 if a timeout occurs (see also the SetTimer() method).
Returns the identifier for the condition which has been fulfilled (as defined with DefineCondition()). If a number of conditions are fulfilled simultaneously, the smallest identifier is returned.
The list of conditions defined previously using DefineCondition() is reset before a result is returned. This means that the conditions must be redefined for e new WaitUntilCondtion query.

Example

The following example illustrates the use of Auto9750.Connection in an Excel table (“Visual Basic for Applications” script).

```
Option Explicit
' Declaration of the global variables

   Dim myconn As Object
   Dim rc As Integer
   Dim msg As String
   Dim connect As Boolean
Sub start_BS2000()
' Start a BS2000 connection
```

```
    Dim pass As String

    If connect = True Then
       MsgBox "Connection already started"
       Exit Sub
    End If

    On Error GoTo err

    Set myconn = CreateObject("Auto9750.Connection")

    myconn.SetConnection "TEST", "bs2000.mts", "c:\mt9750w\mt9750.exe"
    ' Set connection parameters

    myconn.OpenConnection
    ' Open connection

    rc = myconn.WaitUntil(0, 1, "/ ")
    ' WAit for LOGON prompt

    If rc = 0 Then
       MsgBox "Time out"
       Exit Sub
    End If

    rc = myconn.SetArea(0, 0, ".JOBNAME SET-LOGON-PARAMETERS
            uid,account,'password'")
    ' LOGON ...

    rc = myconn.SendScreen("due1")
    ' and send the screen

    connect = True
    MsgBox "Connection established."
    Exit Sub

err:

    MsgBox "Error establishing connection."
    connect = False

End Sub

Sub file_status()
' Issue SHOW-FILE-ATTRIBUTE command
' Output is sent to the first two columns of the Excel spreadsheet

    Dim i As Integer
    Dim j As Integer

    If connect = False Then
       MsgBox "Connection not started"
       Exit Sub
    End If
    On Error GoTo err
```

```
    rc = myconn.SetArea(0, 0, "SHOW-FILE-ATTRIBUTE * ")
    ' SHOW-FILE-ATTRIBUTE ...

    rc = myconn.SendScreen(„due1“)
    ' and send the screen

    rc = myconn.WaitUntil(2, 1, "%   ")
    ' Wait for response

    myconn.SetTimer (10)

    ' The following loop
    ' traps the “PLEASE ACKNOWLEDGE” prompt
    ' automatically

    i = 2
    j = 2

    While myconn.GetArea(i, 1, 1) = "%" And myconn.GetArea(i, 2, 1) <> ":"
       If myconn.GetArea(i, 1, 7) = "%PLEASE" Then
          i = 1
          rc = myconn.SendScreen("DUE1")
          rc = myconn.Wait(10)
          ' Pause
       Else
          Cells(j, 1) = myconn.GetArea(i, 2, 8)
          ' Copy file size
          Cells(j, 2) = myconn.GetArea(i, 11, 54)
          ' Copy file name
          i = i + 1
          j = j + 1
       End If
    Wend
    Exit Sub

err:
    MsgBox “Connection error”
    connect = False

End Sub

Sub stop_BS2000()
' Stop BS2000 connection

    If connect = False Then
       MsgBox "Connection not started"
       Exit Sub
    End If

    rc = myconn.SetArea(0, 0, "exit-job")
        rc = myconn.SendScreen("due1")
        connect = False
End Sub
```

8 Installing and configuring FHS-DOORS

8.1 Hardware and software requirements

Hardware requirements

– A PC with a 486-based or better processor (Pentiums are recommended) with at least 8 MB RAM or a BS2000 Client PC (MFT2)

Software requirements

MS-DOS/Windows:

- 16-bit version
  - MS-DOS as of V5.0 and Microsoft Windows V3.1x
  - Dialog Builder runtime system as of V2.2; 16-bit version (included in the FHS-DOORS delivery package)
- 32-bit version
  - Windows 95 or Windows NT as of Version 4.0
  - Dialog Builder runtime system as of V2.2; 32-bit version (included in the FHS-DOORS delivery package)
- Communication Layer (TRCOMMS is included in the FHS-DOORS delivery package)
- An emulation that supports the DOORS-DDE protocol (DOORS emulation is included in the FHS-DOORS delivery package)

BS2000:

- BS2000-BC as of V9.5 or BS2000/OSD as of V1.0

For the extended model of FHS-DOORS

– DCAM as of V10.0A
– VTSU as of V9.0B

For UTM applications
- UTM as of V3.2

For TIAM applications
- BS2000-GA as of V10.0 or BS2000/OSD as of V1.0

When using FHS
- FHS as of V7.1

When using FORMPLAG
- FORMPLAG-E V2.4C

8.2 Working on a PC

FHS-DOORS converts alphanumeric forms of BS2000 applications graphically and displays them as windows on the display terminal of a PC. The following section describes which steps must be performed on the PC in order to enable and use the graphical conversion of FHS-DOORS.

8.2.1 Installing and uninstalling FHS-DOORS

Place the CD-ROM with FHS-DOORS in the CD-ROM drive and start the *setup* program.

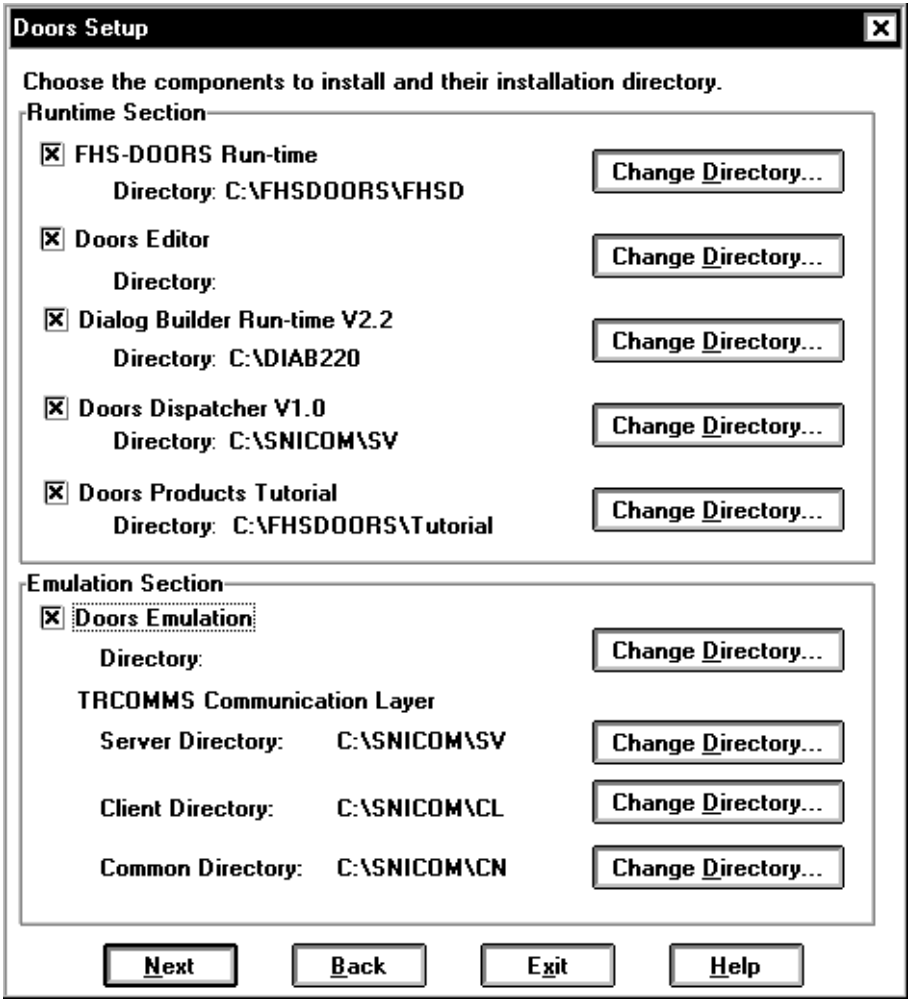
Windows V3.1x — Select the *File/Run...* command from the Windows Program Manager and enter the following in the command line:
drive-letter:setup
This starts the installation program.

Windows 95 — Select *Settings*→*Control Panel* from the Start menu. In the Control Panel, double-click the *Add/Remove Programs* icon and then click the *Install* button. Follow the installation instructions displayed on the screen.

The following installation methods can be used to install FHS-DOORS:

- *Developer installation*
  This includes the installation of the FHS runtime system and the development environment needed to provide BS2000 applications with a graphical interface. In order to use this installation method, you will need a password, which is included in the supplied documentation.
- *User installation*
  This includes the installation of all FHS-DOORS components that are needed to display the converted BS2000 applications.

In both cases, the following window will be subsequently displayed so that you can define the scope and the directories for the installation:



The table below shows which components are included in the FHS-DOORS package and in which directories the individual components are installed during a default installation.

Components	Default directory	
	16-bit	32-bit
FHS-DOORS runtime system	C:\FHSDOORS\FHSD	C:\Programs\ FHSDOORS\FHSD
Dialog Builder runtime system	C:\DIAB220	C:\Programs\DIAB232
DOORS Dispatcher	C:\SNICOM\SV	C:\SNICOM\SV
DOORS Emulation	C:\DOORS_EM	C:\Programs\DOORS_EM
DOORS Editor	C:\FHSDOORS\DOORS_ED	C:\Programs\ FHSDOORS\DOORS_ED
TRCOMMS Communication Layer	C:\SNICOM\CL (driver) C:\SNICOM\CN (dir1 file) C:\SNICOM\SV	C:\SNICOM\CL (driver) C:\SNICOM\CN (dir1 file) C:\SNICOM\SV

Table 6: Default directories for the FHS-DOORS installation

8.2.2 Configuring the emulation

FHS-DOORS handles communication with BS2000 via 9750 emulation. This emulation must support the DDE protocol. The following emulations support this protocol:

– DOORS Emulation as of Version V1.1
– MT9750 as of Version V4.0C
– MPS as of Version V3.0
– LOG-TE as of Version V4.2
– CONWARE as of Version V4.5

The DOORS Emulation Version V1.1 or later that is supplied with FHS-DOORS is sufficient to set up the connection with BS2000. The method used to configure an emulation is described below using the DOORS emulation as an example. If you plan to use some other emulation, the environment variables, entries and parameters may differ. For a description of the parameters, refer to the documentation supplied with the emulation.

► First verify that the environment variable *CMXPATH* for the Communication Layer and the variables *DIABUILD_HOME, LANG, DIABPATH* and *BMPATH* for the Dialog Builder have been set in the *autoexec.bat* file. The following (or similar) entries must be made for these variables in the *autoexec.bat* file:

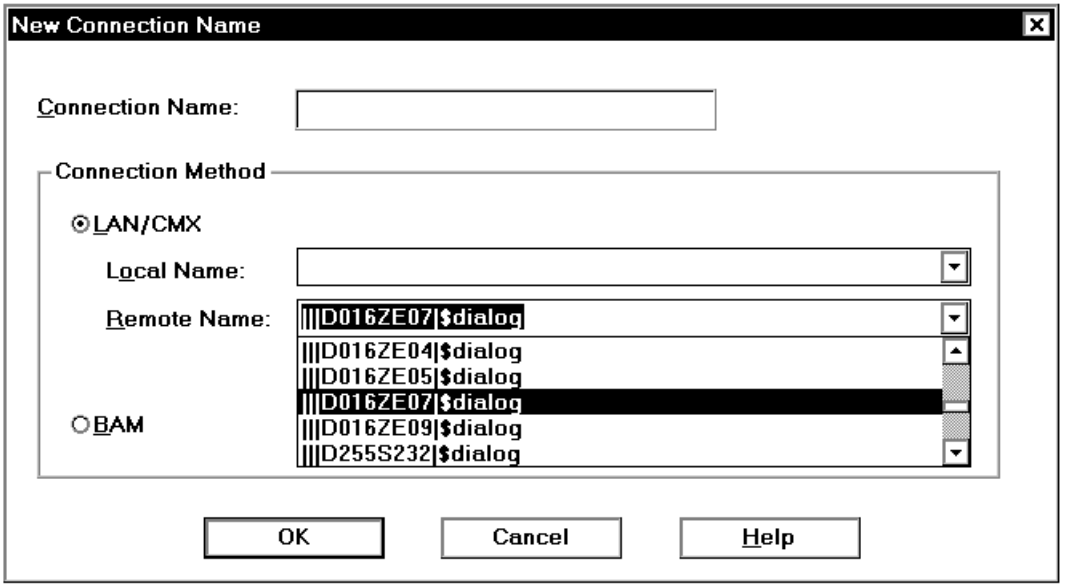
```
SET CMXPATH=C:\SNICOM\CN
SET DIABPATH=C:\DIAB220\%%L\%%F
SET DIABUILD_HOME=C:\DIAB220
SET BMPATH=C:\DIAB220\BITMAP
SET LANG=DE
```

Ensure that the installation path for both programs is included in the *PATH* variable.

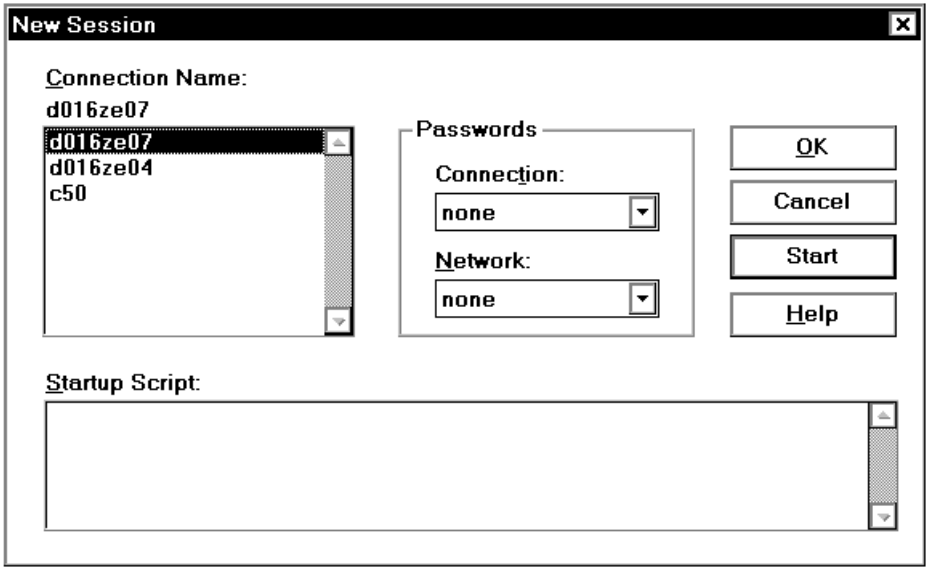
- If you are working via LAN1, you should also check whether the BS2000 host is entered in the transport name directory file *dir1* and whether this file can be accessed via the *PATH* variable (extend the *PATH* variable with the entry *C:\SNICOM\SV* if required). You will find more information on this subject in the manual for the DOORS Emulation V1.1 [3] or the communication manager CMX [9].

Creating a parameter file for the emulation

- Start the DOORS emulation by double-clicking on the program icon in the FHS-DOORS program group.
- Activate the *Configuration/Connection Name* command to set up a new connection.
- To set up the connection with the BS2000 host, click the *Setup* button in the *New Connection Name* dialog box.



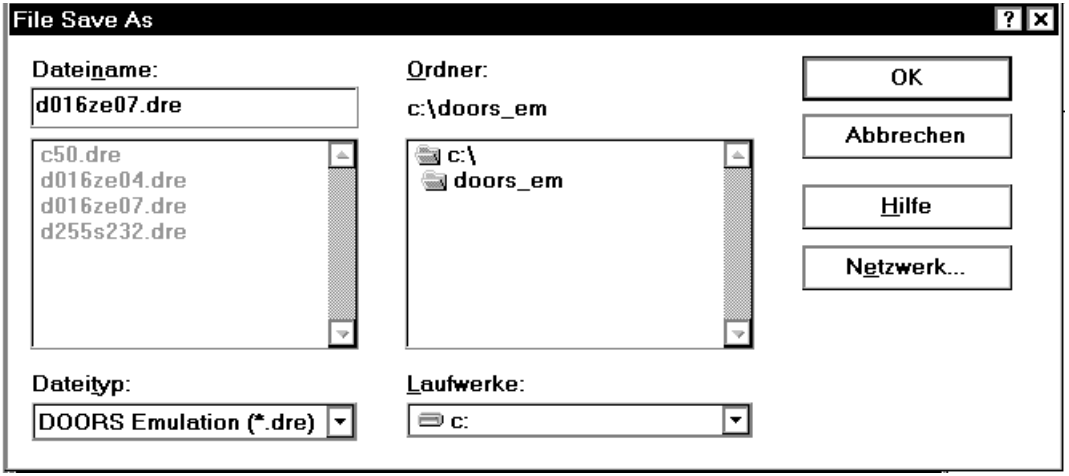
- Enter the name of the new connection (e.g. the name of the BS2000 host) in the *Connection Name* field of the *New Connection Name* dialog box. This name is freely selectable and may be up to 8 characters in length.
- Then select the connection type. In this example, you should select a LAN1/CMX connection.
- The *Remote Name* list shows all entries that are present in the transport name directory file *dir1*. Select the entry for the BS2000 host on which your application program is installed.
- Close both dialog boxes with *OK*.
- To configure a new session, activate the *File/New* command. The *New Session* dialog box appears.



- The *Connection Name* list in the *New Session* dialog box shows the newly installed connection (*D016ZE07* in this case). This connection will be already marked as selected.

  When you click *OK* in this dialog box, only an emulation window appears. Clicking on *Start* causes the connection to the specified host to be established.
- Click either the *OK* or *Start* button. An emulation window with or without a connection will be displayed.

► Close the emulation window using the *Close* command of the window menu. A dialog box will appear asking you whether you want to save the session parameters. Confirm this dialog box with *OK*.



► Enter a name for the new session. The *.dre* suffix is automatically appended to the name of the configuration file. In this example, the configuration file is given the name of the BS2000 host on which the application program is installed: d016ze07.dre.

► Exit the emulation with *File/Exit*.

Potential problems

Depending on the BS2000 application, the required terminal type may vary:

– FHS-DOORS and Desk2000 expect a 9763 terminal type. The only emulation that behaves like a 9763 terminal in the default configuration and thus requires no changes is the DOORS Emulation.

  All other emulations behave like a 9750 terminal and must therefore be adapted:

  ► Set the terminal type to 9763 in the respective emulation being used.

  ► If you are working with a BAM connection, you will also need to adapt the PDN.

– All other BS2000 applications expect a 9750 terminal type. Consequently, only the DOORS Emulation needs to be adapted in this case:

  ► To do this, open the parameter file (with the *.dre* extension) and add the following line in the *[connection]* section:

  StationType=9750

  ► If you are working with a BAM connection, you will also need to adapt the PDN.

8.2.3 Configuring FHS-DOORS

FHS-DOORS can be configured by using the commands in the *Setup* menu of FHS-DOORS. The settings are saved in the *fhsd.ini* file:

Whenever a new session is opened, FHS-DOORS uses default values from the *fhsd.ini* file. If you want to change these default values:

– Open a new session (see section “Opening a new session” on page 41 for details)
– Configure the session
– Save the session under the file name *fhsd.ini* in the Windows directory

8.2.4 Uninstalling FHS-DOORS

If you are working with the 32-bit version and Windows 95 or Windows NT, select *Settings*→*Control Panel* from the Start menu. In the Control Panel, double-click the *Add/ Remove Programs* icon and then click the *Uninstall* button. Follow the uninstallation instructions displayed.

If you are working with the 16-bit version and Windows V3.x, run the program *deinstall.exe* from the installation directory of FHS-DOORS by double-clicking on it from within the File Manager or by entering the path and program name in the *File/Run* command of the Program Manager. This starts the Uninstall program of FHS-DOORS. Follow the program instructions displayed.

8.2.5 Notes on the Dialog Builder runtime system

The Dialog Builder runtime system is configured using two configuration files, *diabuild.ini* and *dbrun.ini*. To avoid problems with panel presentation under FHS-DOORS, you must observe the following instructions:

– In *diabuild.ini* (in the Dialog Builder directory), the default alignment must be set to left-justified, and a default font must be specified:

[DiaBuild]
DefaultFont="-*-courier-*-*---*-0090-*-*-m-0000-FHS-DOORS-*"
DefaultAlignment=XmALIGNMENT_BEGINNING

– In *dbrun.ini* (in the Windows directory), the following lines must be added:

[XtXm]
Alignment=XmALIGNMENT_BEGINNING
Background=LightGrey

– The default font set using the FHS-DOORS *Setup/Font...* function must be the same as that defined in *diabuild.ini* (which in the above example would be FHS-DOORS, 9-point).

8.3 Working in BS2000

In order to work with FHS-DOORS, you will also need to perform the following tasks in BS2000.

8.3.1 Installing FHS-DOORS-LC (BS2000/OSD)

FHS-DOORS-LC (BS2000/OSD) is installed with the standard software delivery system SOLIS.

The file names and their significance are described in the file SOLDOC.FHS-DOORS-LC.ver.

The FHS-DOORS-LC statements are contained in a system syntax file and must be merged into the global system syntax file.

The FHS-DOORS-LC program must be stored under $.FHS-DOORS-LC.

8.3.2 Configuring UTM applications

In order to work with UTM applications under FHS-DOORS, the communication partner must be declared for a terminal of type 9763 . This can be done by modifying the KDCFILE with the KDCDEF control statement PTERM as shown in the example below. A separate PTERM statement must be specified for each communication partner.

Example

```
PTERM DSN25999,PRONAM=D241KR25,PTYPE=T9763,LTERM=DSN25999
LTERM DSN25999
```

For further details on configuring UTM, see the "*open*UTM Generating and Administering Applications" manual [18].

As FHS also passes the form name to FHS-DOORS, the messages generated by FHS for FHS-DOORS are longer than normal:
13 bytes for each form + 13 bytes for each subform.

8.3.3 Configuring user programs

The TSTAT macro is used to query terminal attributes. TSTAT returns the value X'78' (for front-end terminal).

Programs which query terminal attributes with the TSTAT macro and which pass the returned terminal type to FHS are fully executable under FHS-DOORS without modifications. These programs include all COBOL programs that use the FHS-COBOL interface.

Programs which do not use TSTAT must be first adapted so that they query the terminal type by invoking TSTAT and then pass the type returned to FHS. You can also specify the terminal type when defining the control area with the FHS macro MDCBL, or change it before each MCMAP call by using the FHS macro MUCBL.

Name	**Operation**	**Operands**
name	MDCBL	... [,DEVICE={ ... | FE }] ...

DEVICE= specifies the type of the terminal on which you wish to output the form if you have specified a different type in the form definition.

Name	Operation	Operands
name	MUCBL	... [,DEVICE={...|FE}] ...

The FHS macros MDCBL and MUCBL are described in detail in the "FHS" manual [15].

Since FHS also passes the form name to FHS-DOORS, the messages generated by FHS for FHS-DOORS are longer than normal:
13 bytes for each form + 13 bytes for each subform.
This may mean that the buffer in the program has to be enlarged.

8.3.4 Configuring OMNIS

You can also use OMNIS with FHS-DOORS. For working with a UTM application, the PC must be configured as follows in the KDCFILE:

Example

```
TPOOL LTERM=OMNIS0, NUMBER=9, PRONAM=D123ZE40, PTYPE=T9763,
      PROTOCOL=NO
```

PROTOCOL=NO **must** be specified.

On a PC, the emulation used requires a 9763 terminal type for the BS2000 connection. This terminal type is the default value in the DOORS emulation, but must be declared explicitly for the MT9750 emulation, for instance.

8.3.5 Defining the PDN for a BAM connection

It is not possible to directly specify a communication partner for a terminal of type 9763 for a PC in the PDN. To make such a declaration, you must first edit the code of the PDN to be loaded. To do this, enter the following command at the BS2000 console or add it to the “soffile” file:

Example:

/DADM WDT,27,801,13,4F

WDT	Write Data Table
27	Processor name of the system on which the PDN will be loaded (entry under PROC after a SHOW-JOB-STATUS command for D241KR27)
801	Station name of the PC which uses FHS-DOORS as an emulation (entry under STATION after a SHOW-JOB-STATUS command for DSN27801)
13	Shift in the table
4F	Emulation type 9763

If the communication partner has been declared correctly in the PDN, the terminal type of the PC that is used by FHS-DOORS as the emulation must be DSS-FE.

Alternatively, you can select the following PDN generation:

```
...
* EMULATION DIALOG-STATION
         XSTAT STATNAM=...
               ...
               STATTYP=PC-9750
               ...
               STASEML=TYP79
```

9 The ESS-DOORS Event Stream Service

ESS-DOORS is available under FHS-DOORS and provides you with a graphical interface for the Event Stream Service (ESS) in BS2000/OSD (as of BC V3.0) and thus for BS2000 system operation as well. ESS-DOORS is intended for system operators and BS2000 users who want the convenience of a graphical interface even with the Event Stream Service.

ESS-DOORS consists of form description files and is not an application in a strict sense. FHS-DOORS uses the ESS-DOORS forms and automatically converts them into panels. If FHS-DOORS is started, a session is opened, and the presentation function of ESS is called in a connection to the BS2000 host, the ESS-DOORS forms will be used for the display.

In order to use the ESS-DOORS application, you will need to be familiar with ESS. A detailed description of ESS can be found in the manuals “BS2000/OSD-BC V3.0 Introductory Guide to Systems Support” [11] and “BS2000/OSD-BC V3.0 Commands”, Volumes 1 [12] and 3 [14].

9.1 Installing and configuring ESS-DOORS

Hardware and software requirements

The hardware and software requirements are the same as for FHS-DOORS. ESS-DOORS works with FHS-DOORS as of V3.1.

The following additional software requirements must be met in BS2000:

– BS2000/OSD-BC as of V3.0

– FHS-TPR V8.2

Installing ESS-DOORS

ESS-DOORS consists of the language-specific LMS libraries *SYSFHS.ESS-DOORS.12x.E* and *SYSFHS.ESS-DOORS.12x.D* in BS2000/OSD, which contain all form description files. These libraries are part of the basic configuration of BS2000/OSD as of V3.0.

In order to install ESS-DOORS on the PC, the optimized ESS-DOORS panels must be copied from the LMS library in BS2000 to a directory on the PC.

- First create the new language-specific directories *C:\FHSDOORS\ESSDOORS\E* and *C:\FHSDOORS\ESSDOORS\D*.
- Transfer all library members to the PC into the appropriate directory (see also section “Manual panel distribution” on page 76.
- If you want to use other directories on your PC, you will need to edit the path names of the bitmap files (*.bmp) in the *NBESTAP* panel and the Help panels *NBESCLH* and *NBESCRH* with the DOORS editor. Note that you must enter the full path names of the *.bmp files.

These steps are required, since the automatic distribution of ESS-DOORS forms by downloading is not supported for all forms of ESS-DOORS.

Configuring ESS-DOORS

The subsystem FHS-TPR V8.2 must be active in BS2000.

In order to work with the graphical interface of ESS-DOORS on a PC, you will need to inform FHS-DOORS of the directory in which the ESS-DOORS panels are stored.

This is done on creating a new session with FHS-DOORS by selecting the appropriate directory in the *Resource Path* list in the *New Session* dialog box (*C:\FHSDOORS\ESSDOORS\E* for English or *C:\FHSDOORS\ESSDOORS\D* for German).

Note that the language setting that is made for the forms on the PC via the selected resource path must match the task-local language setting of your BS2000 task (MODIFY-MSG-ATTIBUTES commands).

- In the *Options/Setup* dialog box, turn off the *Optimized 810 Protocol* option in the *Misc* tab.
- All other setup options such as the font, display attributes or “3D Look” can be selected directly via the *Setup* menu of FHS-DOORS. Note that it is important to select a suitable font here (Arial with a font size of 10 or FHS-DOORS with a font size of 8). If you select a font size that is too large, all characters cannot be displayed in some fields.

Configuring the first session for ESS-DOORS

If no saved ESS-DOORS session exists as yet, you will need to perform the following steps:

- Start FHS-DOORS by double-clicking on the FHS-DOORS program icon.
- Select the command *Session/New*. The *New Session* dialog box is opened.

- In the *Resource Path* list, select the directory containing the ESS-DOORS forms in the desired language (*C:\FHSDOORS\ESSDOORS\E* for English or *C:\FHSDOORS\ESSDOORS\D* for German). Note that the language setting that is made for the forms on the PC via the selected resource path must match the task-local language setting of your BS2000 task.
- Enter the following in the entry fields for the *Emulation Parameters*:

  Filename File name of the used terminal emulation

  Parameters Parameters for the terminal emulation or file name of the parameter file created for the terminal emulation.

- Click *Start* to begin the ESS-DOORS session.
- Switch to the emulation window and start ESS from there as usual, i.e. either with the SHOW-SYSEVENT-LOG command or with a LOGON procedure.
- Save the session with the *Session/Save As...* command in a parameter file with the suffix *.drs.

9.2 Working with ESS-DOORS

Starting ESS-DOORS

To start ESS-DOORS, you must first start FHS-DOORS by double-clicking on its program icon and then select the parameter file at the end of the *Session* menu from within FHS-DOORS.

The easiest way to start ESS-DOORS is a direct start with a double-click on a previously prepared ESS-DOORS program icon. If you have not defined any startup script, the set terminal emulation will first respond with a prompt to log on to the host (SET-LOGON-PARAMETERS command). After you have logged on successfully, you can set your event stream by means of commands. As soon as you start the presentation function of ESS with the SHOW-SYSEVENT-LOG command, ESS-DOORS will respond by displaying its first panel on the PC.

ESS-DOORS is operated in exactly the same way as the presentation function in BS2000 under FHS-TPR. The BS2000 menus and menu commands are also available under ESS-DOORS, and important functions can be activated directly with buttons. It is, however, not possible to enter commands in ESS-DOORS. More information on the operation of ESS-DOORS can be found in the online help.

Creating a program icon for ESS-DOORS

If desired, you can use the Windows Program Manager to create a separate program icon for an ESS-DOORS session so that the session can then be directly started by simply double-clicking on that icon.

To create an ESS-DOORS program icon, proceed as follows:

- Copy the FHS-DOORS program icon (using the *File/Copy* command in the Program Manager).
- Change the program properties (via the *File/Properties* command in the Program Manager) by entering the following in the input fields:

  Description Title for the program icons, e.g. ESS-DOORS

  Command Line File name of FHS-DOORS, followed by the file name of the parameter file for FHS-DOORS, e.g. FHSD.EXE ESSDOORS.DRS

Exiting ESS-DOORS

Select the *Session/Exit* command to exit FHS-DOORS, and thus ESS-DOORS as well. The connection to the BS2000 host remains open, and the emulation window is displayed on the screen. If desired, you can change the settings for your event stream in the emulation window or enter other BS2000 commands. To start ESS-DOORS again, enter the SHOW-SYSEVENT-LOG command.

The command *Session/Close* can also be used to exit the ESS-DOORS session. In this case, the connection to the BS2000 host is cleared, but FHS-DOORS remains loaded.

Requesting help

The Help panels of ESS-DOORS were converted from the FHS Help forms and adapted to the Windows Help system. Consequently, you can also request help for ESS-DOORS via the corresponding buttons in panels and dialog boxes. Help on ESS-DOORS is also available via commands in the *Help* menu. The *Help/Index* command offers a comprehensive list of all available help topics together with dynamic links.

Changing the size of ESS-DOORS panels

ESS-DOORS panels can be increased or decreased in size, but the absolute size of the font and that of the fields always remain the same. It is not possible to size the panels proportionally. You could, however, change the font by using the *Setup/Font* command. By default, the panels are created by using an Arial font with a 10-point size for ESS-DOORS and an 8-point size for FHS-DOORS. Larger fonts could result in some fields not being fully displayed.
It is not possible to change the font in the Help panels.

10 Appendix

SDF syntax representation

This syntax description is valid for SDF V4.0A.The syntax of the SDF command/statement language is explained in the following three tables.

Table 7 on page 125: notational conventions

The meanings of the special characters and the notation used to describe command and statement formats are explained in table 7.

Table 8 on page 127: data types

Variable operand values are represented in SDF by data types. Each data type represents a specific set of values. The number of data types is limited to those described in table 8.

The description of the data types is valid for the entire set of commands/statements. Therefore only deviations (if any) from the attributes described in table 8 are explained in the relevant operand descriptions.

Table 9 on page 132: suffixes for data types

Data type suffixes define additional rules for data type input. They can be used to extend or limit the set of values. The following short forms are used in this manual for data type suffixes:

cat-id	cat
completion	compl
construction	constr
correction-state	corr
generation	gen
lower-case	low
manual-release	man
odd-possible	odd
path-completion	path-compl
separators	sep
underscore	under
user-id	user
version	vers
wildcards	wild

The description of the ‘integer’ data type in table 9 contains a number of items in italics; the italics are not part of the syntax and are only used to make the table easier to read.

The description of the data type suffixes is valid for the entire set of commands/statements. Therefore only deviations (if any) from the attributes described in table 9 are explained in the relevant operand descriptions.

Metasyntax

Representation	Meaning	Examples
UPPERCASE LETTERS	Uppercase letters denote keywords. The keywords for constant operand values begin with *	**HELP-SDF** **SCREEN-STEPS** = ***NO**
UPPERCASE LETTERS in boldface	Uppercase letters printed in boldface denote guaranteed or suggested abbreviations of keywords.	**GUID**ANCE-**MODE** = ***Y**ES
=	The equals sign connects an operand name with the associated operand values.	**GUID**ANCE-**MODE** = ***NO**
< >	Angle brackets denote variables whose range of values is described by data types and suffixes (see Tables 8 and 9).	**SYNTAX-F**ILE = <full-filename 1..54>
Underscoring	Underscoring denotes the default value of an operand.	**GUID**ANCE-**MODE** = ***NO**
/	A slash serves to separate alternative operand values.	**NEXT-FIELD** = ***NO** / ***Y**ES
(…)	Parentheses denote operand values that initiate a structure.	,**UNGUID**ED-**DIA**LOG = ***Y**ES (...) / ***NO**
[ ]	Square brackets denote operand values which introduce a structure and are optional. The subsequent structure can be specified without the initiating operand value.	**SELECT** = [***BY-ATTR**IBUTES](...)
Indentation	Indentation indicates that the operand is dependent on a higher-ranking operand.	,**GUID**ED-**DIA**LOG = ***Y**ES (...) ***Y**ES(...) **SCREEN-STEPS** = ***NO** / ***Y**ES

Table 7: Metasyntax (section 1 of 2)

Representation	Meaning	Examples
|	A vertical bar identifies related operands within a structure. Its length marks the beginning and end of a structure. A structure may contain further structures. The number of vertical bars preceding an operand corresponds to the depth of the structure..	**SUP**PORT = ***TAPE**(...) *TAPE(...) | **VOL**UME = ***ANY**(...) | ***ANY**(...) | | ...
,	A comma precedes further operands at the same structure level.	**GUID**ANCE**-MODE** = ***NO** / ***Y**ES ,**SDF-COM**MANDS = ***NO** / ***Y**ES
list-poss(n):	The entry “list-poss” signifies that a list of operand values can be given at this point. If (n) is present, it means that the list must not have more than n elements. A list of more than one element must be enclosed in parentheses.	list-poss: ***SAM** / ***ISAM** list-poss(40): <structured-name 1..30> list-poss(256): ***OMF** / ***SYSLST**(...) / <full-filename 1..54>
Abbreviation:	The name that follows represents a guaranteed alias for the command or statement name.	**HELP-SDF** Abbreviation: **HPSDF**

Table 7: Metasyntax (section 2 of 2)

Data types

Data type	Character set	Special rules
alphanum-name	A…Z 0…9 $, #, @	
cat-id	A…Z 0…9	Not more than 4 characters; must not begin with the string PUB
command-rest	freely selectable	
composed-name	A…Z 0…9 $, #, @ hyphen period catalog ID	Alphanumeric string that can be split into multiple substrings by means of a period or hyphen. If a file name can also be specified, the string may begin with a catalog ID in the form :cat: (see data type full-filename).
c-string	EBCDIC character	Must be enclosed within single quotes; the letter C may be prefixed; any single quotes occurring within the string must be entered twice.
date	0…9 Structure identifier: hyphen	Input format: yyyy-mm-dd yyyy: year; optionally 2 or 4 digits mm: month dd: day
device	A…Z 0…9 hyphen	Character string, up to 8 characters in length, corresponding to a device available in the system. When guided dialog is used, SDF displays the valid operand values. For notes on possible devices, see the relevant operand description.
fixed	+, - 0…9 period	Input format: [sign][digits].[digits] [sign]: + or - [digits]: 0...9 must contain at least one digit, but may contain up to 10 characters (0...9, period) apart from the sign.

Table 8: Data types (section 1 of 6)

Data type	Character set	Special rules
full-filename	A…Z 0…9 $, #, @ hyphen period	Input format: [:cat:][$user.] { file | file(no) | group | group { (*abs) | (+rel) | (-rel) } } :cat: optional entry of the catalog identifier; character set limited to A...Z and 0...9; maximum of 4 characters; must be enclosed in colons; default value is the catalog identifier assigned to the user ID, as specified in the user catalog. $user. optional entry of the user ID; character set is A…Z, 0…9, $, #, @; maximum of 8 characters; first character cannot be a digit; $ and period are mandatory; default value is the user' s own ID. $. (special case) system default ID file file or job variable name; may be split into a number of partial names using a period as a delimiter: $name_1[.name_2[...]]$ $name_i$ does not contain a period and must not begin or end with a hyphen; file can have a maximum length of 41 characters; it must not begin with a $ and must include at least one character from the range A...Z.

Table 8: Data types (section 2 of 6)

Data type	Character set	Special rules
full-filename (continued)		#file (special case) @file (special case) # or @ used as the first character indicates temporary files or job variables, depending on system generation. file(no) tape file name no: version number; character set is A...Z, 0...9, $, #, @. Parentheses must be specified. group name of a file generation group (character set: as for “file”) group{(*abs), (+rel), (-rel)} (*abs) absolute generation number (1-9999); * and parentheses must be specified. (+rel) (-rel) relative generation number (0-99); sign and parentheses must be specified.
integer	0…9, +, -	+ or -, if specified, must be the first character.
name	A…Z 0…9 $, #, @	Must not begin with 0...9.

Table 8: Data types (section 3 of 6)

Data type	Character set	Special rules
partial-filename	A…Z 0…9 $, #, @ hyphen period	Input format: [:cat:][$user.][partname.] :cat: see full-filename $user. see full-filename partname optional entry of the initial part of a name common to a number of files or file generation groups in the form: $name_1$.[$name_2$.[...]] $name_i$ (see full-filename). The final character of “partname” must be a period. At least one of the parts :cat:, $user. or partname must be specified.
posix-filename	A...Z 0...9 special characters	String with a length of up to 255 characters; consists of either one or two periods or of alpha-numeric characters and special characters. The special characters must be escaped with a preceding \ (backslash); the / is not allowed. Must be enclosed within single quotes if alter-native data types are permitted, separators are used, or the first character is a ? or ! A distinction is made between uppercase and lowercase.
posix-pathname	A...Z 0...9 special characters structure identifier: slash	Input format: [/]$part_1$/.../$part_n$ where $part_i$ is a posix-filename; max. 1024 characters; must be enclosed within single quotes if alter-native data types are permitted, separators are used, or the first character is a ? or !

Table 8: Data types (section 4 of 6)

Data type	Character set	Special rules
product-version	A…Z 0…9 period single quote	Input format: [[C]'][V][n]n.nann['] correction status release status where n is a digit and a is a letter. The release and correction status must be specified if product-version does not include a suffix (see suffix without-corr and without-man in Table 9). product-version may be enclosed within single quotes (possibly with a preceding C). The specification of the version may begin with the letter V.
structured-name	A…Z 0…9 $, #, @ hyphen	Alphanumeric string which may comprise a number of substrings separated by a hyphen. First character: A...Z or $, #, @
text	freely selectable	For the input format, see the relevant operand descriptions.
time	0…9 structure identifier: colon	Time-of-day entry: Input format: { hh:mm:ss / hh:mm / hh } hh: hours mm: minutes ss: seconds } Leading zeros may be omitted
vsn	a) A…Z 0…9 b) A…Z 0…9 $, #, @	a) Input format: pvsid.sequence-no max. 6 characters pvsid: 2-4 characters; PUB must not be entered sequence-no: 1-3 characters b) Max. 6 characters; PUB may be prefixed, but must not be followed by $, # or @.

Table 8: Data types (section 5 of 6)

Data type	Character set	Special rules
x-string	Hexadecimal: 00…FF	Must be enclosed in single quotes; must be prefixed by the letter X. There may be an odd number of characters.
x-text	Hexadecimal: 00…FF	Must not be enclosed in single quotes; the letter X must not be prefixed. There may be an odd number of characters.

Table 8: Data types (section 6 of 6)

Suffixes for data types

Suffix	Meaning
x..y *unit*	a) with data type integer: interval specification x minimum value permitted for “integer”. x is an (optionally signed) integer. y maximum value permitted for “integer”. y is an (optionally signed) integer. *unit* with “integer” only: additional units. The following units may be specified: *days* *byte* *hours* *2Kbyte* *minutes* *4Kbyte* *seconds* *Mbyte* b) with the other data types: length specification x minimum length for the operand value; x is an integer. y maximum length for the operand value; y is an integer. x=y the length of the operand value must be precisely x.
with	Extends the specification options for a data type.
-compl	When specifying the data type “date”, SDF expands two-digit year specifications in the form yy-mm-dd to: 20yy-mm-dd if yy < 60 19yy-mm-dd if yy ≥ 60
-low	Uppercase and lowercase letters are differentiated.
-under	Permits underscores “_” for the data type “name”.

Table 9: Data type suffixes (section 1 of 6)

<table>
<tr><th>Suffix</th><th colspan="2">Meaning</th></tr>
<tr><td>with (contd.)</td><td colspan="2"></td></tr>
<tr><td>-wild(n)</td><td colspan="2">Parts of names may be replaced by the following wildcards.
n denotes the maximum input length when using wildcards.
Due to the introduction of the data types posix-filename and posix-pathname, wildcards from the UNIX world (referred to below as POSIX wildcards) are now accepted in addition to the usual BS2000 wildcards. However, only POSIX wildcards or only BS2000 wildcards should be used within a search pattern. Only POSIX wildcards are allowed for the data types posix-filename and posix-pathname. If a pattern can be matched more than once in a string, the first match is used.</td></tr>
<tr><td></td><td>BS2000 wildcards</td><td>Meaning</td></tr>
<tr><td></td><td>*</td><td>Replaces an arbitrary (even empty) character string. If the string concerned starts with *, then the * must be entered twice in succession if it is followed by other characters and if the character string entered does not contain at least one other wildcard.</td></tr>
<tr><td></td><td>Termina-ting period</td><td>Partially-qualified entry of a name.
Corresponds implicitly to the string “.*”, i.e. at least one other character follows the period.</td></tr>
<tr><td></td><td>/</td><td>Replaces any single character.</td></tr>
<tr><td></td><td><s_x:s_y></td><td>Replaces a string that meets the following conditions:
– It is at least as long as the shortest string (s_x or s_y)
– It is not longer than the longest string (s_x or s_y)
– It les between s_x and s_y in the alphabetic collating sequence; numbers are sorted after letters (A...Z0...9)
– s_x can also be an empty string (which is in the first position in the alphabetic collating sequence)
– s_y can also be an empty string, which in this position stands for the string with the highest possible code (contains only the characters X'FF')</td></tr>
<tr><td></td><td><s_1,...></td><td>Replaces all strings that match any of the character combinations specified by s. s may also be an empty string. Any such string may also be a range specification “s_x:s_y” (see above).</td></tr>
</table>

Table 9: Data type suffixes (section 2 of 6)

Suffix	Meaning	
with-wild(n)		
(continued)	-s	Replaces all strings that do not match the specified string s. The minus sign may only appear at the beginning of string s. Within the data types full-filename or partial-filename the negated string -s can be used exactly once, i.e. -s can replace one of the three name components: cat, user or file.
	Wildcards are not permitted in generation and version specifications for file names. Only system administration may use wildcards in user IDs. Wildcards cannot be used to replace the delimiters in the name components cat (colon) and user ($ and period).	
	POSIX wildcards	Meaning
	*	Replaces any single string (including an empty string). An * appearing at the first position must be duplicated if it is followed by other characters and if the entered string does not include at least one further wildcard.
	?	Replaces any single character; not permitted as the first character outside single quotes.
	$[c_x\text{-}c_y]$	Replaces any single character from the range defined by c_x and c_y, including the limits of the range. c_x and c_y must be normal characters.
	[s]	Replaces exactly one character from string s. The expressions $[c_x\text{-}c_y]$ and [s] can be combined into $[s_1c_1\text{-}c_2s_2]$
	$[!c_x\text{-}c_y]$	Replaces exactly one character not in the range defined by c_x and c_y, including the limits of the range. c_x and c_y must be normal characters. The expressions $[!c_x\text{-}c_y]$ and [!s] can be combined into $[!s_1c_1\text{-}c_2s_2]$
	[!s]	Replaces exactly one character not contained in string s. The expressions [!s] and $[!c_x\text{-}c_y]$ can be combined into $[!s_1c_1\text{-}c_2s_2]$

Table 9: Data type suffixes (section 3 of 6)

<table>
<tr><th>Suffix</th><th colspan="2">Meaning</th></tr>
<tr><td>with (contd.)</td><td colspan="2"></td></tr>
<tr><td>-constr</td><td colspan="2">Specification of a constructor (string) that defines how new names are to be constructed from a previously specified selector (i.e. a selection string with wildcards). See also with-wild.
The constructor may consist of constant strings and patterns. A pattern (character) is replaced by the string that was selected by the corresponding pattern in the selector.
The following wildcards may be used in constructors:</td></tr>
<tr><td></td><td>Wildcard</td><td>Meaning</td></tr>
<tr><td></td><td>*</td><td>Corresponds to the string selected by the wildcard * in the selector.</td></tr>
<tr><td></td><td>Termina-
ting period</td><td>Corresponds to the partially-qualified specification of a name in the selector;
corresponds to the string selected by the terminating period in the selector.</td></tr>
<tr><td></td><td>/ or ?</td><td>Corresponds to the character selected by the / or ? wildcard in the selector.</td></tr>
<tr><td></td><td><n></td><td>Corresponds to the string selected by the n-th wildcard in the selector, where n is an integer.</td></tr>
<tr><td></td><td colspan="2">Allocation of wildcards to corresponding wildcards in the selector:
All wildcards in the selector are numbered from left to right in ascending order (global index).
Identical wildcards in the selector are additionally numbered from left to right in ascending order (wildcard-specific index).
Wildcards can be specified in the constructor by one of two mutually exclusive methods:
1. Wildcards can be specified via the global index: <n>
2. The same wildcard may be specified as in the selector; substitution occurs on the basis of the wildcard-specific index. For example:
the second “/” corresponds to the string selected by the second “/” in the selector</td></tr>
</table>

Table 9: Data type suffixes (section 4 of 6)

Suffix	Meaning
with-constr (continued)	The following rules must be observed when specifying a constructor: – The constructor must include at least one wildcard of the selector. – If the number of identical wildcards exceeds those in the selector, the index notation must be used. – If the string selected by the wildcard <...> or [...] is to be used in the constructor, the index notation must be selected. – The index notation must be selected if the string identified by the wildcard “*” is to be duplicated. For example: “<n><n>” must be specified instead of “**”. – The wildcard * can also be an empty string. Note that if multiple asterisks appear in sequence (even with further wildcards), only the last asterisk can be a non-empty string, e.g. for “****” or “*//*”. – Valid names must be produced by the constructor. This must be taken into account when specifying both the constructor and the selector. – Depending on the constructor, identical names may be constructed from different names selected by the selector. For example: “A/*” selects the names “A1” and “A2”; the constructor “B*” generates the same new name “B” in both cases. To prevent this from occurring, all wildcards of the selector should be used at least once in the constructor. – If the selector ends with a period, the constructor must also end with a period (and vice versa).

Table 9: Data type suffixes (section 5 of 6)

<table>
<tr><th>Suffix</th><th>Meaning</th></tr>
<tr><td>with-constr (contd.)</td><td>Examples:
<table>
<tr><th>Selector</th><th>Selection</th><th>Constructor</th><th>New name</th></tr>
<tr><td>A//*</td><td>AB1
AB2
A.B.C</td><td>D<3><2></td><td>D1
D2
D.CB</td></tr>
<tr><td>C.<A:C>/<D,F></td><td>C.AAD
C.ABD
C.BAF
C.BBF</td><td>G.<1>.<3>.XY<2></td><td>G.A.D.XYA
G.A.D.XYB
G.B.F.XYA
G.B.F.XYB</td></tr>
<tr><td>C.<A:C>/<D,F></td><td>C.AAD
C.ABD
C.BAF
C.BBF</td><td>G.<1>.<2>.XY<2></td><td>G.A.A.XYA
G.A.B.XYB
G.B.A.XYA
G.B.B.XYB</td></tr>
<tr><td>A//B</td><td>ACDB
ACEB
AC.B
A.CB</td><td>G/XY/</td><td>GCXYD
GCXYE
GCXY. 1)
G.XYC</td></tr>
</table>
1) The period at the end of the name may violate naming conventions (e.g. for fully-qualified file names).</td></tr>
<tr><td>without</td><td>Restricts the specification options for a data type.</td></tr>
<tr><td>-cat</td><td>Specification of a catalog ID is not permitted.</td></tr>
<tr><td>-corr</td><td>Input format: [[C]'][V][n]n.na[']
Specifications for the data type product-version must not include the correction status.</td></tr>
<tr><td>-gen</td><td>Specification of a file generation or file generation group is not permitted.</td></tr>
<tr><td>-man</td><td>Input format: [[C]'][V][n]n.n[']
Specifications for the data type product-version must not include either release or correction status.</td></tr>
<tr><td>-odd</td><td>The data type x-text permits only an even number of characters.</td></tr>
<tr><td>-sep</td><td>With the data type “text”, specification of the following separators is not permitted: ; = ( ) < > ? (i.e. semicolon, equals sign, left and right parentheses, greater than, less than, and blank).</td></tr>
<tr><td>-user</td><td>Specification of a user ID is not permitted.</td></tr>
<tr><td>-vers</td><td>Specification of the version (see “file(no)”) is not permitted for tape files.</td></tr>
</table>

Table 9: Data type suffixes (section 6 of 6)

Glossary

active window
Only one window can be activated at one time. All actions relate to the active window. Clicking on a visible part of a window or selecting the entry for that window from the Window menu makes that window the active window.

alphanumeric object
An alphanumeric object is a reduced representation of the form window that requires fewer system resources at runtime. It is displayed alphanumerically in lines, columns, height and width and covers a selected area of the form window. This area must not contain any truncated fields.

application
An application is a program used for a specific task.

attribute
Attributes define the properties of objects. Attributes can refer to aspects such as the color, size and position of an object.

BAM
Bit-oriented Asynchronous Multipoint procedure; a connection method for setting up a connection to a remote host across a network.

callback
A callback is assigned to an object and describes a so-called higher-level event (e.g. "pushbutton pressed").

capture
A selection mechanism that is activated using the *Options/Capture* command. A capture requires a form identifier.

click on
Clicking on is a mouse technique combining the actions of pointing and clicking.

clicking
Clicking is a mouse technique which involves pressing and quickly releasing a mouse button.

clipboard

A temporary storage area used to move or copy data between applications.

command

An entry in a menu which initiates actions. Click on the command in the menu to select it.
To help you to find commands, the menu is separated from the command by a slash in the description. Thus, for instance, *Session/New* indicates that you should select the *New* command from the *Session* menu.

connection method

A method such as LAN, BAM or HDLC/AFP, used to set up a connection to a remote host across a network.

connection name

Connections are identified by their connection names.
A connection name is an alias for the parameters of a connection to the host application. A connection name contains the following information: the connection method, the local name, the remote name and whether the connection is active or passive.

control menu

The control menu is located in the title bar and contains commands for moving, sizing, maximizing or minimizing windows.

cursor

The cursor is an on-screen symbol which identifies the current position within the screen. See also *mouse pointer* and *insertion point*.

dialog box

A dialog box is a window-like area which provides information or allows you to make settings.

double-click

Double-clicking is a mouse technique in which a mouse button is clicked twice in rapid succession.

dragging

Dragging is a mouse technique in which the mouse is moved with one of its buttons held down.

dre file

Contains the settings for a connection to the DOORS emulation; also called the parameter file for the DOORS emulation.

drs file

Contains the settings for an FHS-DOORS session; also called the parameter file for FHS-DOORS.

field

Fields are used to output data from or input data to the application. Fields can be: output fields, entry fields, option buttons (radio buttons), check boxes or drop-down combo boxes (not for forms).

form

A form (sometimes referred to as a mask) is displayed on screen and allows data to be output from and input to an application.

form identifier

The form identifier is a square area in the form with which FHS-DOORS can uniquely identify the form. FHS-DOORS uses the content of the area (text) for this purpose. It is usually best to select the “form name” as the identifier. An identifier can be defined by using the left mouse button in a capture.

HDLC/AFP

A connection method for setting up a connection to a remote host across a network. This connection method employs the HDLC (high-level data link control) protocol and the AFP (alternating pulse-edge modulation) method.

host application

Application which runs on a remote computer in a network.

icon

Symbol which appears on the screen to represent items such as applications, files, commands or buttons.

insertion point

The insertion point indicates the position at which data will be inserted in text or graphics. See also *cursor*.

LAN/CMX

Local Area Network; a connection method for setting up a connection to a remote host across a network. LAN/CMX covers a number of different communication methods, such as LAN1, Win-Sockets, LAN-Manager, LAN/CMX, WAN/CMX.

list box

A list box contains a list of items (such as file names). A list box allows you to select one of the items by clicking on it.
A drop-down list box is a variant which saves space, since only the list item which is currently valid is visible at all times. If you click on the arrow to the right of the combo box, this opens or closes the portion of the list which is not always visible.

menu

A menu is an on-screen list of grouped commands represented by menu items. Selecting one of the items causes the associated command to be executed.

modal dialog box

A modal dialog box is one where input is mandatory, i.e. an entry must be made before the next step can be carried out.

mouse

A mouse is an input device equipped with two or three buttons. Moving the mouse causes the mouse pointer to move on the screen. The mouse buttons are used to initiate a number of different actions: see *click*, *drag*, *double-click*.

mouse pointer

The mouse pointer is an on-screen cursor controlled by the mouse. Moving the mouse across your desk or a mouse pad causes the pointer to move on the screen.

object

Objects are uniquely identifiable processing units, which can be distinguished on screen. Objects can be pushbuttons, fields or list boxes, for instance.

panel

A *panel* is a graphical window on the PC stored in a semantics file (sdc file). The panel is generated from a form using FHS-DOORS or the form converter (BS2000/OSD).

partial panel

See subpanel.

point

Pointing is a mouse technique in which the mouse pointer is moved onto an object (such as a pushbutton).

press

Pressing is a mouse technique in which a mouse button is pressed and held down.

primary window

The primary window (or main window) is the window assigned to an application. It is the first window to appear after the application is started.

pushbutton

Pushbuttons are used to confirm entries and settings and to initiate further actions.

rbn file

An rbn file (resource binary file) contains the description of the layout of a form in binary form.

res file

A res file (resource file) contains the description of the layout of a form.

scroll bar

Scroll bars let you quickly move screen components up, down, left or right if only part of a window, field or list box is currently visible.

sdc file

An sdc file (semantic description compressed file) describes the graphical representation of a form in a host application. An sdc file contains a description of the form, the way in which it is represented graphically (panel) and the links between the graphical objects in the panel and the fields in the form as a semantic description.
sdc files are created using the WIN-DOORS/FHS-DOORS Capture function or using the FHS-DOORS-LC (BS2000/OSD) form converter. They can then be edited and optimized with the DOORS editor.

subdialog box

A subdialog box is a dialog box opened as an extension to a dialog box which is already open.

subpanel

A subpanel corresponds to a selected area of a form. This area can be selected with the right mouse button in the *Capture* dialog box. The subpanel is line oriented and has an identifier. A line of a form can only be used in a subpanel.

title bar

The title bar contains the control menu and the name of the window or dialog box and can be used to move the window around the screen.

window

Windows are rectangular, framed screen areas which divide the working area into mutually independent, resizeable and freely movable segments.

Abbreviations

AFP	alternating pulse-edge modulation (German acronym)
BAM	Bit-oriented Asynchronous Multipoint procedure
BCAM	Basic Communication Access Method (BS2000/OSD)
DCAM	Data Communication Access Method (BS2000/OSD)
DSS	terminal (German acronym)
FHS	Format Handling System (BS2000/OSD)
FHS-DE	FHS Dialog Extension (BS2000/OSD)
FT	File Transfer
HDLC	High-level Data Link Control
IFG	Interactive Format Generator (BS2000/OSD)
ISAM	Indexed Sequential Access Method (BS2000/OSD)
LAN	Local Area Network
LMS	Library Maintenance System
OMNIS	control system for concurrent use of multiple applications
PLAM	Program Library Access Method (BS2000/OSD)
rbn	binary resources for Dialog Builder as of V2.1
res	FHS-DOORS resources
SAM	Sequential Access Method (BS2000/OSD)
sdc	semantic description compressed; FHS-DOORS semantic description files
SDF	System Dialog Facility (BS2000/OSD)
TIAM	Terminal Interactive Access Method (BS2000/OSD)
UTM	Universal Transaction Monitor (BS2000/OSD)
VTSU	Virtual Terminal SUpport (BS2000/OSD)
WAN	Wide Area Network

Related publications

Please apply to your local office for ordering the manuals.

Windows

[1] **WIN-DOORS** (BS2000/OSD, MS-Windows)
Graphical Interface for BS2000/OSD Applications
User Guide

Target group
This manual is intended for BS2000 developers who wish to equip BS2000 applications with a graphical interface.
Contents
The manual describes the productive model and the functions of WIN-DOORS. A sample session is used to illustrate how you work with WIN-DOORS. The parameters that allow you to tailor sessions to applications are also described, as are the interfaces for creating a library for format recognition and OLE automation.

[2] **WIN-DOORS/FHS-DOORS** (BS2000/OSD, MS-Windows)
Optimizing Panels with the DOORS Editor
User Guide

Target Group
The manual addresses BS2000 developers who wish to optimize formats for use under WIN-DOORS/FHS-DOORS.
Contents
The manual describes how converted formats can be processed using the DOORS Editor. It explains how you work with the DOORS Editor, and the op-tions available for user-specific extensions using the Dialog Builder. It also contains a reference section on the object attributes. A description of the interface covers the online help system for the DOORS Editor.

[3] **DOORS Emulation** (Windows, BS2000/OSD)
Basic Emulation 9750 for BS2000/OSD Connections
User Guide

Target group
The manual addresses users wishing to use DOORS emulation for connections to BS2000/OSD partners.
Contents
The manual describes the functions of DOORS emulation. It describes working with DOORS emulation, connection setup (connection names) and the use of startup scripts for automating connection setup. A full description of the DOORS emulation interface is included in the online help for DOORS emulation.

[4] **Dialog Builder**
Development Environment for MS-WINDOWS
User Guide

[5] **MT9750**
9750 emulation
User Guide

[6] **LAN1 V3.0** (MS-DOS)
User´s Guide

[7] **LAN1 V3.0** (MS-DOS)
Administrator´s Guide

[8] **LAN1 V3.0** (MS-DOS)
Configuration Guide

[9] **CMX** (MS-DOS)
Communication Manager User Guide

Target group
Users and network administrators
Contents
CMX is a transport access system that complies with ISO 8072 and enables communication between applications in PCS and other computer systems. It can be operated using either menus or commands.

BS2000/OSD

[10] **SDF** (BS2000/OSD)
Introductory Guide to the SDF Dialog Interface
User Guide

Target group
BS2000/OSD users
Contents
This manual describes the interactive input of commands and statements in SDF format. A Getting Started chapter with easy-to-understand examples and further comprehensive examples facilitates use of SDF. SDF syntax files are discussed.

[11] **BS2000/OSD-BC**
Introductory Guide to Systems Support
User Guide

Target group
This manual is addressed to BS2000/OSD systems support staff and operators.
Contents
The manual covers the following topics relating to the management and monitoring of the BS2000/OSD basic configuration: system initialization, parameter service, job and task control, memory/device/user/file management, assignment of privileges, accounting and operator functions.

[12] **BS2000/OSD-BC**
Commands, Volume 1, A-L
User Guide

Target group
The manual addresses both nonprivileged BS2000/OSD users and systems support.
Contents
This manual contains BS2000/OSD commands ADD-... to LOGOFF (basic configuration and selected products) with the functionality for all privileges. The introduction provides information on command input.

[13] **BS2000/OSD-BC**
Commands, Volume 2, M-SG
User Guide

Target group
The manual addresses both nonprivileged users and systems support.
Contents
This manual contains BS2000/OSD commands MODIFY-... to SET-... (basic configuration and selected products) with the functionality for all privileges.

[14] **BS2000/OSD-BC**
Commands, Volume 3, SH-Z
User Guide

Target group
The manual addresses both nonprivileged users and systems support.
Contents
This manual contains BS2000/OSD commands SHOW-... to WRITE-... (basic configuration and selected products), with the functionality for all privileges. By means of SDF-P users of SHOW commands can make use of output in structured S variables which are described in Volume 4.

TRANSDATA

[15] **FHS** (TRANSDATA)
User Guide

Target group
Programmers
Contents
Program interfaces of FHS for TIAM, DCAM and UTM applications. Generation, application and management of formats.

[16] **IFG for FHS** (TRANSDATA)
User Guide

Target group
Terminal users, application engineers and programmers
Contents
The Interactive Format Generator (IFG) is a system that permits simple, user-friendly generation and management of formats at a terminal. In conjunction with FHS, these formats can be used on the host computer. This user guide describes how formats are generated, modified and managed, plus also the new functions of IFG V8.1.

[17] ***open*UTM**
Concepts and Functions
User Guide

Target group
Anyone who wants information about the functionality and performance capability of *open*UTM.
Contents
The manual contains a general description of all the functions and features of *open*UTM, plus introductory information designed to help first-time users of *open*UTM.

[18] ***open*UTM** (BS2000/OSD)
Generating and Handling Applications
User Guide

Target group
This manual is intended for application planners, technical programmers, administrators and users of UTM applications.
Contents
The manual describes the generation of UTM applications with distributed processing, the tools available with *open*UTM for this purpose, and the UTM objects created in the course of generation. It also contains all the information necessary for structuring, operating and monitoring a productive UTM application.

[19] **UTM** (TRANSDATA)
Programming Applications
User's Guide

Target group
Programmers of UTM applications
Contents
– Language-independent description of the KDCS program interface
– Structure of UTM programs
– KDCS calls
– Testing UTM applications
– All the information required by programmers of UTM applications
Applications
BS2000 transaction processing

[20] **TIAM** (TRANSDATA, BS2000)
User Guide

Target group
– BS2000 users (non-privileged)
– Programmers
Contents
– All TIAM commands and macros
– The TIAM COBOL interface with the TIAM COBOL macros
– Examples
Applications
BS2000 timesharing mode

[21] **DCAM** (TRANSDATA)
Program Interfaces
Reference Manual

Target group
– Managers
– Application planners
– Programmers
– System and network administrators

Contents
Description of the Data Communication Access Method DCAM

Index

D

E

T

X

Contents

FHS-DOORS V3.1 (BS2000/OSD, MS-Windows)

Graphical interface for BS2000/OSD applications

User Guide

Target group
This manual adresses BS2000 developers who wish to equip BS2000 applications with a graphical interface.
Contents

The manual describes the usage model and the functions of FHS-DOORS. A sample session provides an example of how to work with FHS-DOORS. The manual also describes the parameters with which sessions can be configured and as required by applications, and the interfaces for a library for format recognition and OLE automation. It also contains a description of the format converter FHS-DOORS-LC and of the Event Stream Service (ESS-DOORS).

Edition: March 1997

File: FHS_DOR1.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.