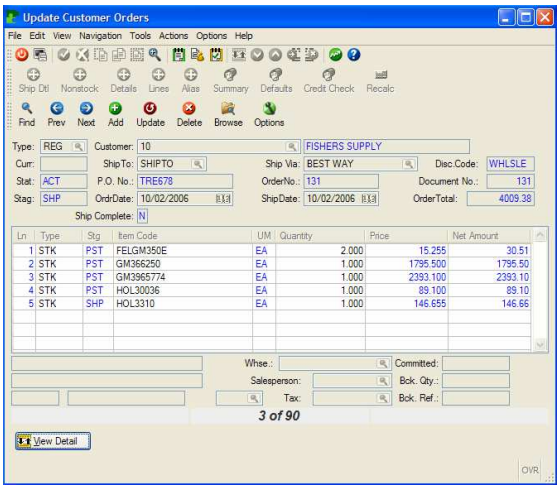
FX-T-SC-TR-AD Fitrix Screen Technical Reference

Version 5.20 Addendum

This outline identifies changes needed to this existing Fitrix document, to make it consistent with version 5.20. It references each section in the base document where updates are needed.

General

The Case Tool screens used by developers continue to operate in character mode. Screens created by the version 5.20 Case Tool are now upgraded to present an interface consistent with a windows look and feel. The Technical Reference manual shows generated screens in their original character format. These images will be upgraded in a later release of this manual, to be consistent with the new windows-based graphical images, at a later time. Here is an example of the upgraded images (See page 1-9):



Preface

1. Character mode is no longer supported at runtime. The VDT tools still operate in character mode, but generated programs operation only in graphical mode.

   There is one exception-- when FGLGUI is set to '0' (console mode), report programs that do not require screen input will produce a report. This is necessary for reports that are 'croned' up to run at a later time.

Introduction – Section 1

Technical Merits

WYSIWYG form development (page 1-2)
The form development tools are character-based, but are used to generate a graphical user interface-based screen for runtime. The general notion of a graphically-based design does not apply to this tool.

Overview

Form Painter (page 1-5)
The interface is character-based.

Getting Started – Section 2

Library Source Files (page 2-6)

A new directory exists under the lib directory, called 'css'. It contains the style-sheet-related XML files used by version 5.20 to render applications in a graphical format. It is pointed to by the environment variable "CSSPATH".

Running the Form Painter (page 2-25)

A new argument exists:

-rw Uses the contents of the .per file to refresh the form painter database tables representing the same contents. See page 5-3 for more details of how screen specifications are stored in database tables.

The Data Entry Interface – Secion 3

Browsing with a Browse Screen (page 3-10)

The 'Go-To' ring menu option is no longer available.

Detail Line Commands (page 3-12)

To access the detail lines on a form generated as header-detail, you may press the Ctrl-Tab key, or click the button 'View Detail'. To return to the Ring Menu commands, press Ctrl-Tab again, or click the button 'Exit Detail'.

The Zoom Form

Page 3-14 - Whenever a zoom is available, the zoom icon will be active. If a zoom is not available, the zoom icon will still display, but as a greyed-out button.

Page 3-15 – The first time a zoom is called, it may display in one of three ways:

- If a filter was passed to the zoom at design time, the results of that filter will display automatically
- If a filter was not passed, the list will display of all rows in the associated table
- If a filter of "1=0" was passed, the zoom will display an empty list, giving the user the ability to define a filter before selecting the rows to display.

The Find Command (Zoom)

In some cases the zoom list can be changed by selecting a different filter. Press the 'New Search' button to enter a different filter, then press 'OK' to create the new list.

The Sort Command (Zoom)

This command allows you to change the order in which the list is sorted. Click on the column heading for the field on which to sort. The list will redisplay, sorted in ascending sequence by the selected column heading. Clicking on the column again will resort in descending sequence.

The Tab Command (Zoom)

This key is no longer used. To select new filter criteria, click the 'New Search' button.

Program Information Menu (page 3-19)

The Program Information Menu is accessed by:

- Pressing the Ctrl-Y key
- Clicking the 'Technical Status' icon
- Select Technical status from the 'Help' ring-menu option

Hot Keys Menu (page 3-22)

An option to access hot key assignments directly from the Program Information Menu is no longer available. The hot key screen can be display directly from a generated program by pressing the Ctrl-e key, or by selecting the 'Hot Key Button Definition' option from the 'Tools' ring menu option.

Default Screen Attributes (page 3-24)

Attribute Conventions

Windows – All display in the windows native format, with a blue title bar.
First window – title displays from the menu title
Other windows – title supplied from the 'OPEN WINDOW' 4GL statement
Display Statements – All data elements are displayed in test boxes, with a greyed-out attribute.
During Input – All fields allowing input are displayed with a white background. The current field is displayed with a dark background.
During Input Array – Detail rows are displayed in a spreadsheet-like 'grid', with white backgrounds for the fields. Pressing the tab key moves the cursor to the next column in the current row, or to the next row, if at the end of a row. To exit the input array, press either Ctrl-Tab, or OK.

Form Definition – Section 7

Creating Screen Forms – Section 10

Header Screens (page 10-8)

The screen specifications will look slightly different, with the addition of new attributes for the windows interface. See the following example. This is a header-detail screen, discussed later on page 10-11, but the new attributes will be found in any screen type. For the other screen types in this section, where source code is shown, will have similar changes.

```
SCHEMA standard

LAYOUT
VBOX nm_vbox_main (TAG="tg_vbox_main")
GRID
{
                  Group Code: [A0    ]
                 Description: [A1                          ]

}
END -- GRID
TABLE  (WIDTH=61)
{
           Account     Dept   Description
           [A2       ] [A3 ]  [A4                          ]
           [A2       ] [A3 ]  [A4                          ]
           [A2       ] [A3 ]  [A4                          ]
           [A2       ] [A3 ]  [A4                          ]
           [A2       ] [A3 ]  [A4                          ]
           [A2       ] [A3 ]  [A4                          ]
           [A2       ] [A3 ]  [A4                          ]
           [A2       ] [A3 ]  [A4                          ]
           [A2       ] [A3 ]  [A4                          ]
           [A2       ] [A3 ]  [A4                          ]
}
END -- TABLE
END -- VBOX
END -- LAYOUT

TABLES
    stxactgr
    stxactgd
    stxchrtr

ATTRIBUTES
A0 = stxactgr.grp_key, upshift,
     comments = " Enter a unique code to identify this group of accounts.";
A1 = stxactgr.grp_desc, upshift,
     comments = " Enter a description of this group of accounts.";
buttonedit A2 = stxactgd.acct_no, image="gn_zoomf.png", action=ac_zoom,
     comments = " Enter a ledger account number in this account group.";
buttonedit A3 = stxactgd.department, upshift,
     image="gn_zoomf.png", action=ac_zoom,
     comments = " Enter the department code for this account in this group.";
A4 = stxchrtr.acct_desc, noentry, comments = "";

INSTRUCTIONS
screen record s_groupr (stxactgr.grp_key, stxactgr.grp_desc)

screen record s_groupd[10] (stxactgd.acct_no, stxactgd.department,
    stxchrtr.acct_desc)
```

Differences between Triggers and Extensions (page 13-31)

Using Blocks to Define Custom 4GL Files

A new block statement has been created to support full regeneration and re-merge of custom-written 4GL files. There is now an alternative to writing all custom code in a custom.org file, so that the pre-processor will create a custom.4gl. This change now makes it possible for a programmer to regenerate a program, by:

- Deleting all the *.4gl files in a program directory
- Executing the fg.screen command to regenerate 4GL code
- Executing the fg.make command to recompile the program

Programmers can now define a new block statement within an extension file (*.ext), as follows:

```
new file "custom.4gl"

at_eof
        globals "globals.4gl"

        function mycustom()
        .
        .
        .
        end function;
```

The preprocessor will execute the following steps with these block statements:

- Create a new file called custom.org – with copyright statements only. No executable code will be included
- Create a new file called custom.4gl. The source code after 'at_eof' will be included in the .4gl file.

If modifications are needed to the program source after initial implementation, you may:

- Change the above extension file directly
- Create another extension file with blocks to modify the original source code. This assumes:
    - The original used its own source level block tags to facilitate later modification
    - The added extension file is referenced AFTER the original extension file, in the *.set file.

Compiling and Running Your Programs – Section 14

The version 5.20 compiler now compiles source into *.42r files only. The –F option in the fg.make is no longer needed.

Compiling Generated Code (page 14-2)

Step 3a. converts form source (.per) files to .42f files.
Step 3b. converts 4GL source (.4gl) files to .42m files.
Step 4b. converts form source (.per) files to .42f files.
Step 4c. converts 4GL source (.4gl) files to .42m files.

Differences between RDS and 4GL Compilation (page 14-3)

There is no longer a '4GL' option. All .4gl's compile to .42m files, and programs are created as .42r's.

Using `fg.make` to Compile your Program (page 14-4)

The –F option is no longer available.

Running your Programs (page 14-24)

Invoking Compiled Programs

All compiled programs are invoked in the following way:

```
fglrun <program_name.42r> [args]
```

There are no longer different approaches to invoking. Multiple command line arguments can be used when invoking a program generated by Fitrix *Screen.*

`fglrun program_name.42r [-dbname` *`database`*`] [order` *`order_by_clause`*`] [filter` *`filter_clause`*`] [-a] [-A] [-u] [-U] [f]`

`-dbname`	Specifies the database with which to connect
`-d`	Same as `-dbname`
`order`	Specifies the order of initial selection
`filter`	Limits the initial selection
`-a`	Enters directly into the Add mode
`-A`	Same as above, but exits after add
`-u`	Enters directly into the Update mode
`-U`	Same as above, but exits after update
`-f`	Enters directly into the Find mode

Any other references to **fglgo** in this section should read **fglrun**.

Executing Programs When Using Version Control

The **fglrun** command should still be used, when working with Version Control.

Advanced Features

Event Handling Logic (page 15-2)

Types of Event Handling Logic

There are three types of internal events:

1a. **Local Actions –** Occur at specific locations within the program logic. They are added as `ON ACTION` statements inside a screen interaction statement (`INPUT, INPUT ARRAY, DISPLAY ARRAY, PROMPT, MENU`). Local actions can be added to program logic, attached to activating icons, and attached to hot keys.

Event Flow (page 15-3)

Local actions are processed BEFORE the hot_local and hot_key functions. A local action can directly execute program logic and return. It can also be used to set a hot_key value, then pass control onto the hot_local and hot_key functions.

Creating an Event that Calls a Program (page 15-14)

The 'mz' command is no longer available to execute an external program. To call an external you must incorporate statements into your program as follows:

```
let scratch = "cd $fg/accounting/ar.4gm/i_custr.4gs;",
                  "fglrun i_custr.42r –dbname ",
                  fgl_getenv("DBNAME")
run scratch
```

Shell Escapes and UNIX/Linux Commands (page 15-51)

When running a generated program from Visual Menus, you can press Ctrl-o, and prompt appears to enter an operating system command. For example, if you enter the following command, from any Fitrix Accounting generated program:

`cd $fg/accounting/all.4gm/i_actgrp.4gs;fglrun i_actgrp.42r –d sample`

the 'Update Account Groups' program executes.

Preventing Shell Escapes from VM

Only users that are member of group 'root' are allowed to shell out. This function may be found under the Execute Menu. In addition, for the shell icon to appear on the Button Bar, the environment variable 'mn_buttonbar' must be set to '2'.

Version Control – Section 16

Using fg.go and fg.db (page 16-20)

These commands are no longer used . The `fglrun` command is now used in all cases to execute a compiled program, whether version control is being used or not.

Language Translation – Section 17

Utility Menu

Not available in current version

Fitrix Screen Utilities – Appendix A

The Demo Script (page A-2)

Not available in current version

Viewing Database Table Layouts (page A-9)

The syntax for the `imap` script is as follows:

`imap [-d` *`database_name`*`] table_name`

Cleaning your Database (`fg.delfrm`) (page A-11)

Not available in current version

The .per Specification File – Appendix B

DATABASE SECTION

The section should now be called the SCHEMA Section. The new syntax is:

```
SCHEMA stores
```

SCREEN SECTION

Defines the image of the data-entry screen. The Fitrix 5.20 version incorporates many new tags into the screen section, to be compatible with the windows presentation.

An example:

```
LAYOUT
VBOX nm_vbox_main (TAG="tg_vbox_main")
GRID
{
 Type:[A0 ]Customer:[A1                  |A2                      ]
 Curr:[A3 ]  ShipTo:[A4    ]    Ship Via:[AX            ]Disc.Code:[C1    ]
 Stat:[A5 ]P.O. No.:[A6        ]OrderNo.:[A7        ]  Document No.:[A8    ]
 Stag:[A9 ]OrdrDate:[AA        ]ShipDate:[AB        ]OrderTotal:[AC        ]
      Ship Complete:[A]
}
END -- GRID
TABLE
{
Ln  Type  Stg  Item Code         UM  Quantity    Price        Net Amount
[AD|AE |AF ][AG                  |AH][AI        ][AJ          |AK          ]
[AD|AE |AF ][AG                  |AH][AI        ][AJ          |AK          ]
[AD|AE |AF ][AG                  |AH][AI        ][AJ          |AK          ]
[AD|AE |AF ][AG                  |AH][AI        ][AJ          |AK          ]
[AD|AE |AF ][AG                  |AH][AI        ][AJ          |AK          ]
[AD|AE |AF ][AG                  |AH][AI        ][AJ          |AK          ]
[AD|AE |AF ][AG                  |AH][AI        ][AJ          |AK          ]
[AD|AE |AF ][AG                  |AH][AI        ][AJ          |AK          ]
}
END -- TABLE
GRID
{
[AL                            ]    Whse.:[AM          ]Committed:[AN        ]
[AO                            ]  Salesperson:[AP      ]Bck. Qty.:[AQ        ]
[AR    |AU                  ]       [AV ] Tax:[AY      ]Bck. Ref.:[AW        ]
}
END -- GRID
END -- VBOX
END -- LAYOUT
```

Each screen has the tags `LAYOUT and VBOX` as the first tags after the SCHEMA section. Each tag also has an associated `END – VBOX, END – LAYOUT` tag as well, indicating the end of a tag level.

Screen Tables – Appendix D

Table layouts have changed for the following tables:

stxactnr – the navigation event reference table

```
language char(3),
act_key char(15),
description char(30),
os_command char(74),
press_enter char(1),
buttontext char(32) default null
```

stxhotkd – the navigation event reference table

```
hot_key smallint,
act_key char(15),
hot_module char(8),
hot_program char(8),
hot_user char(10),
hot_visable char(1),
hot_bmp char(1),
hot_bmpname char(24)
```

Control Key Defaults – Appendix E

	Control Key Defaults	**Trapped During Input**
^a	Highlights all data in the input field to be replaced	
^b	Moves the cursor to the first input field on the data-entry form	Yes
^c		
^d	Deletes the character immediately after the cursor	
^e	Activates the Hot Key Definition window	Yes
^f		Yes
^g	Navigate (go)	Yes
^h		Yes
^i		
^j		
^k		
^l		
^m		
^n	Activate the Freeform Notes Window	Yes
^o	Operating System Exit	Yes
^p		Yes
^q		
^r		
^s		
^t	Activates the Personal To-Do Window	Yes
^u		Yes
^v		Yes
^w	Activates the Help Window	Yes
^x		
^y	Activates the Program Information Window	Yes
^z		Yes