Fiasco
Release 1.1

Nils Bandener
Dekanatsgasse 4
D-34369 Hofgeismar
Germany

27.1.96

Contents

Chapter 1

Legal Things

1.1 Disclaimer

The Program "Fiasco" and associated files, in the following called Fiasco, are provided "as is". No representations or warranties are made with respect to accuracy, reliability, correctness of Fiasco, either expressed or implied. In no case I am responsible for any damages caused by this software.

1.2 Copyright

Fiasco is *not* Public Domain. I reserve all rights.
Fiasco Copyright © 1995-1996 Nils Bandener.

Fiasco may be redistributed, as long as these conditions are true:

- The program package has to be complete. See the Filelist for a complete listing of all files Fiasco 1.1 consists of.
- Fiasco may not be distributed for commercial purposes without a written permission by the author. This includes the distribution of Fiasco for too high prices. You may only charge a small fee for media and copying. The distribution on CD-Roms is allowed, if the price of the CD-Rom is not higher than the price of the "Fresh Fish" CD-Roms of Fred Fish.

1.3 Giftware

Fiasco is Giftware , that means, that every User of Fiasco may appreciate the work I have done for Fiasco with some gift. This may be money or

any other little thing (CD-Roms, Books, etc.) or simply a postcard (or nothing, if you think you could use your money for something better; but please note, that ln_Pri of that is -110 ;-).

My Address:
Nils Bandener
Dekanatsgasse 4
D-34369 Hofgeismar
Germany

1.4 Filelist

Fiasco Release 1.1 consists of these files:

```
Fiasco_1.1.info
Fiasco_1.1/ARexx.info
Fiasco_1.1/ARexx/age.rexx
Fiasco_1.1/ARexx/age.rexx.info
Fiasco_1.1/ARexx/print.rexx
Fiasco_1.1/ARexx/print.rexx.info
Fiasco_1.1/ARexx/unlockgui.rexx
Fiasco_1.1/ARexx/unlockgui.rexx.info
Fiasco_1.1/Catalogs/Deutsch/fiasco.catalog
Fiasco_1.1/Databases.info
Fiasco_1.1/Databases/Addresses.info
Fiasco_1.1/Databases/Addresses/Addresses.fdb
Fiasco_1.1/Databases/Addresses/Addresses.fdb.info
Fiasco_1.1/Databases/Addresses/Countries.fdb
Fiasco_1.1/Databases/Addresses/Countries.fdb.info
Fiasco_1.1/Databases/DatatypesDemo.info
Fiasco_1.1/Databases/DatatypesDemo/AmigaWorld.ilbm
Fiasco_1.1/Databases/DatatypesDemo/AmigaWorld.ilbm.info
Fiasco_1.1/Databases/DatatypesDemo/DatatypesDemo.fdb
Fiasco_1.1/Databases/DatatypesDemo/DatatypesDemo.fdb.info
Fiasco_1.1/Databases/DatatypesDemo/Hallelujah.8svx
Fiasco_1.1/Databases/FamilyTree.info
Fiasco_1.1/Databases/FamilyTree/families.fdb
Fiasco_1.1/Databases/FamilyTree/families.fdb.info
Fiasco_1.1/Databases/FamilyTree/persons.fdb
Fiasco_1.1/Databases/FamilyTree/persons.fdb.info
```

```
Fiasco_1.1/Databases/FAQs.info
Fiasco_1.1/Databases/FAQs/FAQS.fdb
Fiasco_1.1/Databases/FAQs/FAQS.fdb.info
Fiasco_1.1/Databases/FAQs/RunMost.rexx
Fiasco_1.1/Databases/FAQs/scantxtdir.rexx
Fiasco_1.1/Databases/FAQs/searchfaqs.rexx
Fiasco_1.1/Databases/FAQs/showtxt.rexx
Fiasco_1.1/Databases/PD-Disks.info
Fiasco_1.1/Databases/PD-Disks/Disks.fdb
Fiasco_1.1/Databases/PD-Disks/Disks.fdb.info
Fiasco_1.1/Databases/PD-Disks/ReadFish.rexx
Fiasco_1.1/Databases/PD-Disks/ReadFish.rexx.info
Fiasco_1.1/Databases/PictureDatabase.info
Fiasco_1.1/Databases/PictureDatabase/Pictures.fdb
Fiasco_1.1/Databases/PictureDatabase/Pictures.fdb.info
Fiasco_1.1/Databases/PictureDatabase/scandir.rexx
Fiasco_1.1/Databases/PictureDatabase/showscr.rexx
Fiasco_1.1/Databases/Videos.info
Fiasco_1.1/Databases/Videos/CalcLen.rexx
Fiasco_1.1/Databases/Videos/Movies.fdb
Fiasco_1.1/Databases/Videos/Movies.fdb.info
Fiasco_1.1/Databases/Videos/Tapes.fdb
Fiasco_1.1/Databases/Videos/Tapes.fdb.info
Fiasco_1.1/Development.info
Fiasco_1.1/Development/fiasco.cd
Fiasco_1.1/Development/fiasco.cd.info
Fiasco_1.1/Development/fiasco.ct
Fiasco_1.1/Development/fiasco.ct.info
Fiasco_1.1/Documentation.info
Fiasco_1.1/Documentation/Deutsch.info
Fiasco_1.1/Documentation/Deutsch/fiasco.dvi
Fiasco_1.1/Documentation/Deutsch/Fiasco.dvi.info
Fiasco_1.1/Documentation/Deutsch/fiasco.guide
Fiasco_1.1/Documentation/Deutsch/Fiasco.guide.info
Fiasco_1.1/Documentation/English.info
Fiasco_1.1/Documentation/English/Fiasco.dvi
Fiasco_1.1/Documentation/English/Fiasco.dvi.info
Fiasco_1.1/Documentation/English/Fiasco.guide
Fiasco_1.1/Documentation/English/Fiasco.guide.info
Fiasco_1.1/Fiasco
Fiasco_1.1/Fiasco.info
Fiasco_1.1/gtlayout.library
```

```
Fiasco_1.1/icons/ARexx.info
Fiasco_1.1/icons/ARexxScript.info
Fiasco_1.1/icons/Databases.info
Fiasco_1.1/icons/Documentation.info
Fiasco_1.1/icons/Drawer.info
Fiasco_1.1/icons/Fiasco.dvi.info
Fiasco_1.1/icons/Fiasco.guide.info
Fiasco_1.1/icons/Fiasco.info
Fiasco_1.1/icons/FiascoProject.info
Fiasco_1.1/icons/XPort.info
Fiasco_1.1/icons/XPortData.info
Fiasco_1.1/Install.info
Fiasco_1.1/Install/Deutsch.info
Fiasco_1.1/Install/English.info
Fiasco_1.1/Install/Install
Fiasco_1.1/Libs/MC68020.info
Fiasco_1.1/Libs/MC68020/gtlayout.library
Fiasco_1.1/XPort.info
Fiasco_1.1/XPort/mpearls_II_findpeals.fxp
Fiasco_1.1/XPort/mpearls_II_findpeals.fxp.info
Fiasco_1.1/XPort/RFF.fxp
Fiasco_1.1/XPort/RFF.fxp.info
Fiasco_1.1/XPort/StdTwist.fxp
Fiasco_1.1/XPort/StdTwist.fxp.info
```

Chapter 2

Introduction

Fiasco is yet another Database for the Amiga. Originally I just wanted to write a simple Program, which tests one‘s vocabulary in English or Latin. Later I implemented the possibility to define more than two fields (answer and question). From then on the program developed more and more and became very similar to a database-program. I had only to make minor changes and there it was! Now Fiasco is grown ‘to a powerful Program with many features.

Basically Fiasco does not differ much from other database-programs. Fiasco does not support hierarchical structures (like AmigaBase), but supports relations[1]. Fiasco has also an ARexx interface, which can be used for controlling Fiasco from other Programs or for assigning ARexx scripts to Fields.

The Mask of Fiasco is not defined by a graphic-file, but is created using internal Images and any non-proportional font. Fiasco provides a lot of Fieldtypes. My personal favorite is the Datatypes fieldtype, which can be used to display graphics, animations, texts etc. directly in the Mask (cf. section 4.3).

There is a second way to display the data: The List (cf. section 4.4). The List is much like the Mask fully configurable. However, you cannot use the List for modifying data.

The searchsystem (cf. section 6) of Fiasco supports “blurred” search and patterns. “Blurred” search means, that Fiasco tests for similarity and not for equality. The threshold, from which Fiasco detects an entry as “similar” may be freely adjusted.

[1] honestly: I have only read about relational Databases, yet. I cannot guarantee, that Fiasco is really relational. Fiasco does at least something quite similar :-)

Besides there are sort-, filter- and count-functions, which are related to the searchsystem.

2.1 Features

Fiasco is able to do this:

- Several projects may be in RAM at the same time. The number of these projects is only limited by the available RAM.
- Masks can be used like any other GUI.
- Masks, lists and requesters are fully font sensitive.
- Many fieldtypes: String, Integer, Float, Cycle, Boolean, Slider, Date, Time, Extern and Datatypes.
- Datatypes fields can be used to display graphics etc. directly in the mask.
- ARexx interface for external control and scripts for fields.
- Freely configurable "Usermenu", which can be used to invoke CLI- and ARexx- Programs.
- Searching allows "blurred" search and patterns.
- Very flexible list, which supports hiding and resizing of entries
- Easy relation handling

2.2 News

Features added in Fiasco 1.1:

- You may convert the fieldtype of a field. E.g. you may convert a string field to a cycle field, whose labels correspondent to the old string contents.
- Import/Export of data.
- Read Only-field attribute.
- Virtual fields
- Fiasco can be run on an own screen.

- Fiasco detects itself in the system and does not start itself twice.
- Less memory fragmentation by using pools.
- Records may be marked.
- Extern and Datatypes fields may be now edited using filerequesters.
- Datatypes fields support "Save" and deferred loading
- Datatypes fields display their messages in themselves
- Changes in the range of the labels for a cycle field don't affect the real contents anymore
- Button fields
- Relations are *much* faster!
- The requesters are created using gtlayout.library by Olaf Barthel
- Old ARexx commands extended and new added.
- Fiasco can talk to you
- The string fields in the mask cycle after Enter
- If Amiga OS 3.0 is avaible, Fiasco increases the size of the file-buffer when it reads or saves a project.
- The ARexx interface examines the arguments using ReadArgs() of the dos.library
- ARexx-Debug displays more information and has a help-button
- Documentation in dvi format for printed manuals
- A editor may be called from fieldreqs to edit ARexx scripts.
- Better pattern matching
- Support of clipboard

Bugs fixed in Fiasco 1.1:

- Usermenu did not allow to move entries in the list.
- Didn't close workbench.library

- Listwindow did not update the up/down-scrollbar, if records were added
- Did not use the information about the font for the mask in the settings file.
- F_GetFieldCont and F_SetFieldCont could not be used for Float, Date, Time, Extern and Datatypes Fields.
- Erased the graphic of the fields sometimes not correctly.
- Crashed, if you select Mask-Mode twice.
- Open in the mask mode did not activate the field under the cursor
- If you activated a project in mask mode, the service window was not updated.
- The options requester did not update the service window.

Chapter 3

Getting Started

3.1 Requirements

The minimum requirements are an Amiga with OS 2.04 (37.175) and 1 MB RAM. Recommended configuration: Amiga with OS 3.x (39.x or higher), 68020 Processor, 2 MB RAM and a Hard Disk.

Features and required OS-Versions:

Localization: Amiga OS 2.1 (38.x)
Screenmode-Requester: Amiga OS 2.1 (38.x)
Online-Help: Amiga OS 3.0 (39.x) or amigaguide.library v34 from FD
Datatypes-Fields: Amiga OS 3.0 (39.x)
Faster project-loading: Amiga OS 3.0 (39.x)

Fiasco 1.1 uses the gtlayout.library by Olaf Barthel for its GUI. The library is included in the archive of Fiasco.

The memory pool functions of Amiga OS 3.0 and Amiga OS 3.1 do not free unused puddles until the pool is deleted. Use SetPatch 40.16 (already included in WB 40.42) to fix this. If you use Amiga OS 2.0 or Amiga OS 2.1 you do not have to care about that.

3.2 Installation

If you have the Installer program from Commodore, simply doubleclick on the install icon of your preferred language in the install drawer. You will be instructed what to do.

If you don‘t own the Commodore Installer, you may simply drag the Fiasco drawer somewhere you want. You may copy the catalogs to locale:catalogs, but they will work at this place, too. You may delete the unused languages in “Documentation” and drag the remaining files in the parent drawers. The files in “Development” and “Install” are not required for normal operation of Fiasco and may be deleted, too. With this configuration, Fiasco will run. If you have a 68020 processor or better, you should delete the file gtlayout.library in the main directory of Fiasco. Then, you should copy the gtlayout.library from the directory libs/68020 into the main directory. If you want to make the gtlayout.library accessible for all programs, you should copy it into the libs: directory.

3.3 Quick Start

These are the most important things, which you have to know while working with Fiasco:

- The program may be started over the Program- or Projecticon
- There are two working-modes: In the record mode you may edit records, search for them etc. The mask mode allows adding or changing of fields. You may control the modes using the menuitems Control/RecordMode and Control/MaskMode
- The service window makes certain operations easier, especially if you don‘t know exactly about shortcuts of the menus. You may open it over Control/ServiceWin. *Attention:* The functions of the gadgets differ in the different modes.
- The List, which can be opened with Control/ListWin, may be changed by clicking in the titles of the list. Clicking one time activates the column. Over the menu List you can do several things with this column. If you click at the right border of a title, you can size the column. The other space can be used to drag the Column to any other place in the list.
- Certain project options may be changed over the menuitem Project/Options
- If you have any problems, you may press the help key while browsing through the menu.

Chapter 4

Basic elements of a Database

A database can be compared with a card file. Most implementations of databases have this scheme as base.

In Fiasco, a database project consists of two components: On the one hand the data, divided in records. On the other hand the mask, which defines the structure of the data.

In the following, the basic general and the basic Fiasco-specific principles of databases are described.

4.1 Records

Records are the file cards of a database. That means, a record is a collection of several points for one main item (e.g. for a person name, address, etc.). In Fiasco, the mask is only able to display one record at the same time. The list displays several records as lines.

4.2 Fields

Fields define, what data may be stored. In Fiasco, the fields are defined in the mask. Fields are the basic elements of the mask and the list.

Fiasco supports several types of fields. More information on the field-types and their features is given in the chapter Fieldtypes.

4.3 Mask

The mask is the way to display data, which Fiasco uses most time. A mask can display only one Record, as opposed to the List. The advantage of the mask is the clarity of the display. In the card file example, the mask defines the structure of the file cards.

The mask is built up of fields, which have several types and images.

If you use normal Amiga programs, you would call these fields "gadgets". Internally, Fiasco uses gadgets (from the gadtools.library) as fields.

Fiasco masks adjust automatically to any non-proportional font. For example, topaz and courier are non-proportional.

To create a mask in Fiasco, you have to be in the mask mode. You may change the position of existing fields using the mouse or make other changes with the Field menu. More on this topic can be found in section 5.1.

4.4 List

Control/ListWindow opens a window, which displays the records in a list. The records are represented by lines, while the fields of a record are represented by columns. The first line of a list shows the ids of each field. If the window is not big enough to display the whole list, you may use the scrollbars in the right and in the bottom border of the window to scroll through the list. The line of the current record is marked using a backfill.

You can select records using the list. Click simply on the line of a record. Changes in the record can only made in the mask.

If a Filter (cf. section 6.6) is active, the list displays only the matching records.

The layout of the list is normally done automatically. Positions and dimensions of the fields in the mask will be used to determine the dimensions in the list. However, you may change the position and the with of each column in the list. To change the width, you have to click in the header-line at the right corner of a column. One line appears, which shows the actual with of the column. Now you may drag the line using the mouse. The place, where you drop the line (that means, you release the mouse button), will be new right border of the column. Columns, which are overlapped by the column, will be shifted to the right.

The position of a column may be changed, too. Click over the middle of the column-header. You may drag now the column in the list. The column will be inserted near as possible at the place, where you drop it.

You may hide columns entirely with the menuitem List/Hide column (cf. section 10.2.38). The columns may be revealed by using List/Show column.

List/Recalc list calculates the positions and dimensions of all columns again. You can compare it with Clean up of the Workbench. Columns, which have been hidden, are kept hidden.

4.5 Stretching of the mask

Normally, the Fields in a Fiasco Mask are placed very tight. This is not very nice, and all other "normal" GUIs leave a few pixels between the gadgets. It would be possible to place one empty line between the fields, but this wastes quickly much place. For this reason, Fiasco makes it possible to leave a few pixels between the gadgets.

These values may be specified in the options requester under Stretch X and Stretch Y.

The owl stretching (ehhhmm – mask stretching %-) makes fields bigger, than specified in the field requesters. This is evident in the lines, because most Fiasco fields only expand to this directions. String fields may be bigger than the number of chars they can hold. The biggest problem are text fields, because the with of them is normally only the required. The stretching makes them wider and the Text has to be centered.

You should specify as X value only zero to avoid these problems and use one column as separator. In Y direction this value 4 is the best.

4.6 Editing Modes in Fiasco

Fiasco divides it's operation into modes. If you want to make changes in the mask, you have to be in the mask mode. If you want to make changes in the records, you have to be in the record mode.

4.6.1 Record Mode

You may add, delete or edit records in this mode. It may be activated with Control/Record Mode. When the record mode is active, the field type cycle gadget in the service window is not selectable and the status gadget displays normally the number of the active record and the number of all records (for instance: 78 / 92).

4.6.2 Mask Mode

This mode makes you able to edit the mask, that means you may create new fields, delete some or change their position or attributes. Relations may also created and changed here. This mode may be activated with Control/Mask Mode. When the mask mode is active, the "tapedeck" gadgets in the service window are ghosted and the status gadget displays normally the coordinates of the cursor in the mask (for instance: X: 10, Y: 5).

Chapter 5

Creating and working with a Database

And now to the practice: If you want to create a database in Fiasco, you will have to create the mask at first, and then the records. Fiasco allows you in most aspects to create a database in an intuitive way.

In the following, the creation of a simple database is described.

5.1 Creating the Mask

To be able to create a mask, you first have to activate the mask mode (Control/MaskMode). After activating this mode, a cursor appears in the mask. With the mouse or the cursor keys you can choose, where the next operation in the mask takes effect. Before creating a new field you have to choose the type of the new field. You can do that using the Field/Type menu or using the lowest gadget in the service window.

After these operations you may use Field/Add Field to create a new field. At first, the field requester appears. The gadgets in the requester depend on the supported attributes of the active field type. They are described in the type documentation for each field. It is not sufficient to click on Ok without any other action. Certain attributes, like the ID have to be specified by the User. Fiasco won't close the requester, if the are any invalid settings in it. After proceeding, the field will appear in the mask.

You may change all attributes in a later operation, with the exception of the fieldtype itself. The position may be changed by mousedragging. The fieldrequester may be opened by doubleclicking on the field or by choosing Field/Edit Field. You should take care, if you want to change the field ID.

Other Fiasco projects or ARexx scripts, which want to access this field, won't find it after the change. If you change the value max chars of string, extern or datatypes fields, you will be informed, whether you could loose data.

With Field/Remove Field you are able to delete Fields. *Attention:* If Settings/Security-Requester is not active, all Data in this Field will be immediately freed. While saving the project the data on disk will be also erased.

In the options requester you may specify further parameters for the current project, like mask stretching, name of the author, etc.

Field/Edit Relations works similar to Edit Field. With this menuitem you are able to control relations of this field. See section 7.1 for more information on relations

If you have completed the mask, you may return to RecordMode. You may create now records.

5.2 Creating and working with Records

If you have a mask with some fields in it, you may create records for storing data. The simplest method of creating a record, is to select Record/Add Record or it's equivalent Add in the service window. This creates — as the name implies — a record and activates it. The fields in the record will contain the values, which have been defined in the mask mode. You may now activate a field using the mouse and edit its contents.

Another way of creating records, is Record/Duplicate Record. This function creates a record, which is an exact clone of the record, which was previously active. All init cont-attributes will be ignored.

If you do not need a record anymore, you may delete it using Record/Remove Record or Delete in the service window. If you have selected Settings/Security-Reqs, you will be asked one time again, before the record is really deleted.

If you want to look through the records you have created, you may use the menu, the service window, the cursor keys or the list window [1]. Because I think, that the GUI parts can be used intuitive, I only explain the cursor-keys. The up-key activates the previous record. The down-key activates the next record. The ordering correspondents to the concept of the list window. The cursor keys paired with the Ctrl key activate the first or the last record, respectively.

[1] Please don't complain, that this is too much ;-)

5.3 Converting Fields

If you have created a project, you may want to change the type of some fields. This may be, because some contents of a field have developed in another direction, you have originally thought of. The convert function may be also useful, if you have imported a file. After a import, all fields are string fields.

To open the convert requester, you must be in mask mode. Activate the field, you want to convert and select Field/Convert Field. The convert requester displays the ID of the field, the current type and the types, the field may be converted to. If you select Alternative format, the convert function may convert the data in a other, often more abstract format. Not all fieldtypes support this option. If you select the the new type and proceed with Ok, the field will be converted. Note, that Fiasco wont warn about any loss of data. If the new fieldtype requires additional attributes (e.g. the extern fieldtype needs a program), the fieldrequester will open after that. Other attributes will use default values. If you convert a field and after that convert it back to it‘s old type, it won‘t have the old attributes.

Information about the results of converting a field from one type to another type can be found in the field documentation. Text and Button fields cannot be converted, in other cases, converting from certain to certain other type makes not much sense (e.g. boolean to datatypes).

5.4 Using marks

In advanced use of a database, marks can become useful. A mark is simply a flag, that may be set or reset for a record. Thus, a record may be marked or unmarked. Marks could be simulated using boolean or other fields, but the marking feature of Fiasco provides some additional advantages. First of all, a marked record can be easily discovered in the list, because it is displayed in a highlighted state. In the mask, marked records can only recognized, if the service window is open. If a marked record is active, a “M” will be displayed at the right of the status gadget of the service window.

Marks can be set using Record/Mark Record and cleared using Record/Unmark Record. If you want to clear all marks in a project, use Record/Unmark all Records. To set all marks, use Record/Mark all Records. If you want to unmark all marked records and to mark all unmarked records, use Record/Toggle all Marks.

Filters (cf. section 6.6) are related to the marks. Thus, Fiasco provides some additional menuitems in the Compare menu. Compare/Mark opens

a search requester, which can be used to mark all records, which match a given pattern. To convert the marks into a filter, use Compare/Marks to filter. To convert a filter into marks, use Compare/Filter to marks. These functions convert set marks to "unfiltered" records and "unfiltered" records to set marks.

Marks are saved in a Fiasco project file and thus will be kept after a new loading of the project.

Chapter 6

Searching in a database

The GUI interface to the Fiasco search function is the search requester. The search requester can be accessed using Compare/Find. You may select, what field will be searched for what pattern. Here are also the controls for the "blurred" search.

The search pattern correspondents to a normal entry. If the a boolean field has to be searched, TRUE correspondents to a selected field and FALSE to a deselected field. Cycle fields take the number of the label (counting from zero) or the real text of the label. Slider-Fields only allow the value. Extern and datatypes fields can only be searched by filename. You cannot search (with the builtin function) for the contents of the files.

The gadgets at the bottom of the requester start the search. If a matching entry is found, the record will be activated. You may search further using the menuitems Compare/Find next and Compare/Find previous.

6.1 Patterns

Besides the plain inputs, you may use patterns. String fields support the use of patterns, similar to AmigaDOS (not the real, because the "blurred Search" is not compatible with them). ? is equal to one unknown character. ?iasco would match Aiasco, Biasco, Ciasco, 1iasco, etc. ???? would match entries, which are 4 chars long. #? stands for an unknown number of unknown characters. A#? would match for example Amiga, Africa, A or ABCD. ?#? searches for all non-empty entries. Similar to AmigaDOS, these characters may be "escaped", if you want to search for entries, which contain these special characters. For escaping such a character, you have

to write an ' before it.

If you want to use a * instead of #?, you may activate it using the menuitem Settings/Use * as pattern (cf. section 10.2.62).

Integer and slider fields support also patterns. These are possible: >, <, >=, <=, !=. The argument has to be given after the pattern. > only search for numbers greater than x, >= only for numbers greater or equal x, < only for numbers less than x, <= only for numbers less or equal x. != searches only for numbers not equal x. There is not pattern like == (equal), because this is represented by the plain number.

The patterns supported by one fieldtype are also documented in the fieldtypes documentation

A summary of all patterns is also available in the appendix.

6.2 blurred Search

"Blurred" search [1] offers you the possibility, to search for entries, which are similar to a pattern. This way you can search for entries, even if you don't know the exact spelling. The tolerance of the function may be set at "factor". 0 matches only entries, which are exactly equal. 100 matches nearly all entries.

The count function is very suitable for experiments with "blurred" search.

6.3 Searching with ARexx

You may also search in a database with ARexx. The commands F_FindNext (cf. section 11.2.15), F_FindPrev (cf. section 11.2.16) and F_FindFirst (cf. section 11.2.14) can be used to search in a database. These commands take the field and the pattern for searching as arguments.

Opposed to the GUI search function, the record won't be activated. Only the number of it will be returned in Result. This number can be used now for example with F_GotoRec (cf. section 11.2.27).

If you call F_FindFirst, F_FindNext or F_FindPrev without arguments, these commands will use the arguments, which have been previously used in the search requester. You may also set these values using F_SetSearchPat (cf. section 11.2.59) and F_SetSearchField (cf. section 11.2.58).

See the documentation for F_FindFirst for an example on searching with ARexx.

[1] The word sounds a little bit strange, if you have a better word for this, write me!

You may create using ARexx a search function, which supports several fields.

6.4 Count

With Compare/Count you can open a requester similar to the search requester. You have also to specify pattern, field and tolerance. If you select Ok, the matches will be counted. This way you can collect experiences with the blurred search.

6.5 Replace

Compare/Replace is the replace function, with which you are able to replace certain values with others. Here are also patterns possible, but only one value will be inserted. The gadget Replacement takes the value, which has to be inserted. If you select the gadget Confirm, you will be asked for each record, if you really want to replace the value. The record will be displayed while you are asked.

Attention: You can quickly destroy important data with a bad pattern (for example: #?)!!!

6.6 Filter

The filter function in Fiasco gives you the possibility to reduce the displayed records to only these records, which match with a pattern. With Compare/Filter you may open the filter requester, which has the same structure as the search requester. If you proceed with Ok, only these records will be displayed, which match with the specified pattern.

With Record/Next and Record/Previous you may browse through the records. The list also displays only matching records. You may temporarily disable the filter using Compare/Filter On?.

If you create new records while the filter is active, the records will be displayed, regardless if they would match the filter or not. If you change the contents of a existing record, it will be also displayed furthermore. If you want to update the filter, you have to call the filter requester and select Ok.

Chapter 7

Alternative Data Mechanisms

Normally, Fiasco stores the data for the fields of a project directly in the file of the project. However, storing certain kinds of data this way may be very inefficient in terms of disk space.

Currently, Fiasco provides two alternative mechanisms to store data. Relations read the data from an other project into their projects. Several projects may access these data. In contrast, virtual fields store their data nowhere! The data are calculated automatically while loading the project.

Please note, that these mechanisms only help to save disk space, in RAM, they require the same amount of memory as other fields do.

7.1 Relations

Relations are fields, which have not stored their contents in the file of the project of the relations, but in another project file. An additional field with a key is required, which is used to identify the record, from which the data should be taken.

This mechanism prevents the situation, that in many different projects the same data are stored. So it helps to save disk space. Furthermore you have to change only the contents of one field in one of the projects, and all other corresponding fields will recognize that change, too.

7.1.1 Creating Relations

For using relations in Fiasco, you have to create a project, which will be the source for another project, which will read the data from there. The source project will be called later "there" and the project, which will read from it will be called "here".

You have to create in the "there" project at least two fields, one for the data and one for the key. The field for the key should be an integer field. This is the fastest method. However, it is possible to use all other field types as keys, too.

You may use the special field attribute gimme unique key, if you want to get automatically a key, whenever you create a new record. Note that the key is only created when you create a new record. If you activate this attribute later, the already existing entries will keep their old value. If you change the contents of such a field, the change will be left without any checking.

It is up to you to choose the type of the second field. If you create fields, which store strings (string, extern and datatypes), you should remember the max chars value, because you also have to use this value in the second project.

If you want to see any effect after activating the relation, you should create a few records with some content at first.

Now it is time to save the project and to create a new one.

The second project must contain again two fields, which have to match in type and max chars if you use string, extern or datatypes. The key field should *not* use unique key, because you should decide freely, what key do you want to use.

Before you activate the relation, the project should be saved in the directory, in which the other project has been saved, in order to be able to use relative and not absolute paths.

Now you can open the relation requester for the field, which is supposed *not* to contain the key (Field/Edit Relations). Primary, you should select the key "here" in the listview in the upper left edge of the window. After that you should select the other project with the filerequester gadget in the middle of the requester. Now you can select the key and the real field "there". Proceed then with Ok. If everything works correctly, the requester is closed and the relations will be loaded. Otherwise, a requester will inform you of any failure.

A relation checklist, which contains the information in a compressed form, is also available.

7.1.2 Technical notes about Relations

Fiasco 1.1 has increased the speed of accessing relations heavily. This could be reached with certain optimizations, the basic code has not changed. The action, which consumes the most time, when accessing relations, is the searching for correct keys. For each key in your project, you have to go through the whole "there" file and compare the keys. Fiasco 1.1 improves access speed by caching entries, which have been read. However, this consumes much memory. In low memory situations, Fiasco will have to throw out some cached entries to recover some memory. Now, Fiasco has to access the disk again to read these entries, which slows the whole thing again down. To avoid this, you should ensure a amount of free memory.

The second method of improving the access speed, is remembering of Keys, which have no matching Key in the "there" file. This is, of course, only useful, if your project contains some "blind" keys.

The third method would not exist, if the first one does not exist. The problem of this method are big files. It uses an unsorted list, in which the entries are stored with their record number. To get a record, Fiasco will have to go through the whole list and compare each record number with the one, it searches for. If this list contains a bigger number of records, this operation will get very slow (Imagine: If you have 1000 Records here, and 1000 Records there, you will have to examine 1000x1000 (MxN) Records in the worst case). Sorting or special search and optimize methods don‘t solve this problem, these methods have a very high overhead and slow the whole thing even more down. Fiasco 1.1 just remembers the address of the record, which it has previously stopped searching and continues searching next time at this record. This increases the speed with certain files, in which the groth of the keys is roughly the same as in the "there" files. I think (hope), that most files are structured in this way. However, the worst case lies still at MxN plus a small overhead, that is required by this handling.

This information should give you a rough idea, why a particular file does not load it's relations as fast as another. Here is a short list of the factors, which may slow down the loading:

- Low memory
- No "blind" keys; All keys have a matching key in the "there" file
- Bad ordering of the records

Note: If you try to load relations from a floppy disk drive, it will get extremely slow, because Fiasco seeks through the whole file.

7.2 Virtual Fields

The data of virtual fields are not saved on disk; their data are calculated while loading the project. If you want to make a field virtual, you should activate the Virtual option in the field requester.

Fiasco uses for calculating these data the ARexx script of a field. The script will be called for each virtual field in each record.

Because Fiasco is in a special state, you are limited in the commands, which you may call. Currently, you may only call these commands:

- F_SetFieldCont
- F_GetFieldCont
- F_MarkRecord
- F_UnmarkRecord
- F_IsMarked
- F_RequestChoice
- F_RequestFile
- F_GetFieldAttributes
- F_VirtualMode

If you call any of these commands, which refers to a record, the record currently in work will be the active one (which you refer to, if you omit the record argument). If you want to refer to different records, you should be aware of the fact, that you don‘t know, which virtual fields have already been done. However, it is guaranteed, that all normal values and all relations are Ok. If you use F_GetFieldCont and F_SetFieldCont, you don‘t know, whether other virtual fields in the current record have been done, yet.

The ARexx script of a virtual field will be also called, like all other fields, after the contents are changed by the user. To find out, whether you are in normal or virtual state, use F_VirtualMode (cf. section 11.2.68).

Chapter 8

Import and Export

The Import and Export functions of Fiasco are provided to be able to load data from other database programs into Fiasco and to write data with Fiasco, which may be read by other programs.

Such Import/Export-files contain the data in coded in ASCII. The fields or records are marked with special characters, which may be freely defined in the Import/Export function of Fiasco.

To the address of the beginner: To be able to use the Import/Export function of Fiasco, you have to know some details. In the following, the structure of Import/Export files is described. If you are familiar with databases, you do not have to read this. The section after that describes the special escape-sequences used by Fiasco. Although other databases may use a similar scheme, you should read this section carefully, because the whole Import/Export function of Fiasco relies on this.

8.1 Structure of Import/Export files

The names used here refer to the gadget labels in the Import/Export requesters. Note, that some marking characters may be empty. To use the file with Fiasco, you have to define at least either Field Start/Field End or Field Separator and either Record Start/Record End or Record Separator. However, the import functions of other programs may get upset, this structure

is correct.

Record Start	
Field Start	
Field Data	Contents of the field in ASCII format.
Field End	
Field Separator	Separates two fields, *not* used after the last field of a record.
...	
Field Start	
Field Data	
Field End	
Record End	
Record Separator	Separates two records, *not*used after the last record of a file.
...	
Record Start	
...	(see above)
Record End	
End of File	

If you activate First Record contains IDs, the field IDs will be stored in the first record, as if they were fields.

An Example of an Import/Export file

Record start and record end are empty. Record separator is a newline. Field start and field end are double quotes. Field separator is a comma. The first record contains the IDs of the fields. Note the empty field in the last record.

```
"Name","FirstName","Rank","Current"
"Picard","Jean-Luc","Captain","U.S.S. Enterprise"
"Riker","William Thomas","Commander","U.S.S. Enterprise"
"Data","","Lieutenant Cmdr.","U.S.S. Enterprise"
```

8.2 How to specify special characters

You often cannot simply type the characters for marking fields and records as plain text. For example, if you want to use the newline character as a record separator, you cannot simply hit the Return key. Instead, you have to type it in as a escape-sequence. Fiasco supports escape sequences similar to the escape sequences of the programming language “C”. The

escape sequences are introduced by a \. These are supported:

\n	Newline-character, ASCII 10
\f	Formfeed-character, ASCII 12
\r	Return-character, ASCII 13
\t	Horizontal tabulator, ASCII 9
\v	Vertical tabulator, ASCII 11
Number	Character with specified ASCII code
Char	Character directly copied

The last option (\ + *Character*) makes it possible to use a character, which is reserved for escape-sequences.

In Import, you may also specify character-classes. Character-classes are introduced in Fiasco with an #. These are supported:

#p	Printable character.
#a	Printable ASCII-character. Without international chars
#c	Control-character. Not printable

Export supports to insert some additional information in the export-file. These commands are introduced with an %. These are supported:

%f	ID of field.
%r	Number of record

8.3 Importing of Data

The import requester is the GUI interface to the import function of Fiasco. You can open it using Project/Import. The file you want to import must be specified in File. After having done this, you have to specify the structure of the file in the requester. If you have exported the file out of another database just before and still know the structure- parameters, you can simply copy them into Fiasco‘s import requester. If this is not so, you can display the contents of the file using the View button at the right side of the filename. Fiasco will start either more or multiview to display the file. If the file has a standard structure, it should not be too difficult to recognize the parameters.

Usually, Record Start and Record End are empty and Record Separator is \n. Field Start and Field End are often empty or double quotes (”). Usual values for Field Separator are a comma (,) or a tabulator (\t).

Skip Lines defines the characters, which introduce a comment *at the beginning of a line*. If present, specify here the comment introducer. This may also be used to skip any formatting information present in the file. Fiasco‘s import function does not use such information. Using Start Skip,

you may skip any initial comment or similar in the file. Max. Fields can be used to specify a record end mark, if neither Record Separator nor Record End can be used.

Activate First Record contains IDs, if the first record of the input file consists not of real data, but the Field IDs. If you activate this, the IDs will be used by Fiasco either to create Fields with these IDs or the use already existent fields.

The options Append new fields and Overwrite old project control, whether you want to update a project or you want to create a new one. If you want to create a new project, you should activate both options.

If you want to continue using the settings you have made in the future, you may save them using the Save button. The settings may be reloaded using Load. Fiasco already comes with several settings to import data from various sources.

To start the import process, you just have to click on Ok.
Attention: If the input file is too big, or even if the structure parameters are defective, the system may run out of memory! Fiasco has no big problems, if it runs out of memory, but other programs may have problems. For this reason, you should be careful with unsaved data!

If everything went well, the import requester closes and the new project is activated. At first, you will want to improve the formatting of the project using the mask mode. If you had not activated First Record contains IDs, you should change the field IDs according to the contents of the fields. In addition, you should create text fields to comment the existing fields. At this point, you have a nicely formatted project. However, all fields are only string fields. You should investigate, whether some fields may be integer, cycle or other fieldtypes. You may change to type of these fields with the convert (cf. section 5.3) function of Fiasco. In the example used in section 8.1, the rank field may be converted to a cycle field.

If you have done these steps, you should save the project under appropriate name.

8.4 Exporting of Data

Exporting data is from the view of Fiasco much less complicated than importing. Normally, you can use the default parameters of Fiasco (No Record Start and Record End, a newline for Record Separator, double quotes for Field Start and Field End and a comma for Field Separator. If you use these parameters, you must take care, that you data do not contain any double quote. In addition, you have to check, if the program, you want to import the data supports these parameters, and conditionally change them.

If you select First Record contains IDs, Fiasco will create an additional record at the top of the file, which contains the field IDs. The file will contain no other formatting information.

If you select Marked Records only, only the marked records will be written.

To start exporting, click on Ok.

Chapter 9

Fieldtypes

Fields are the stores for data. There are only two basic types: “string” and “number”. All other types are more or less modifications of these types, which make the work with the database easier.

Fiasco supports the following types:

- String
- Integer
- Float
- Boolean
- Cycle
- Slider
- Date
- Time
- Extern
- Datatypes
- Text
- Button

9.1 Standard Attributes

These attributes are normally supported by a field type:

ID: This string serves for identification of a field. It is displayed in the mask mode in the fields, in the list header, in the search and related requesters and in the relation requester. You have to use it also in ARexx scripts, if you want to access a field from there. This string must be unique in the current project.

Width: defines the width of the field in the mask in characters. This value is also used as a default value for the width of the list column. However, this can be changed separately.

Init Cont/Use own value: you may specify a value here, which will be used while creating a new record.

Init Cont/Use old value: If you create a new record, the value, which has been used in the old record, will be used in the new record again.

Script: You may specify a ARexx script here, which will be called, when a new record is created, or the contents of a field has been changed. It is possible, that init cont has not the effect, which has been set in the requester, if the script changes the contents of the field.

Read Only: The fieldcontent will be displayed in a recessed box, which cannot be activated or edited.

Virtual: The value of the field is not saved on disk, but is recalculated everytime when the project is loaded. This is done using the init cont attributes and the ARexx script attribute. Please note, that these fields occupy the same amount of RAM as other fields.

By using mask stretching (cf. section 4.5) it is possible, that the attributes, which specify the dimensions of the field, are slightly influenced.

9.2 Fieldtypes

9.2.1 String fieldtype

A string field takes strings with limited length.

New Attributes:

Max Chars: determines, how many chars may be typed in this field. This attribute has direct effect on the size of the project file.

Search equivalent:

correspondents to the content.

Supported search patterns:

? = One unknown character.
#? = No or more unknown characters.

Conversion into a string field:

All fields can be converted without loss of data into a string field. Alternative formats, if supported are specified in parenth eses.
Additional notes:

Boolean	-	"Checked" is TRUE(1), otherwise FALSE(0)
Cycle	-	Label (labelnumber) converted
Slider	-	Level converted
Date	-	Date in format "DD.MM.[YY]YY" converted
Time	-	Time in format "HH:MM[:SS]" converted

9.2.2 Integer fieldtype

You may enter integer numbers in the range from -2,147,483,348 to 2,147,483,347 in an integer field.

New Attributes:

Max Chars: determines the maximum length of a number in chars.

Init Cont/Gimme unique Key: puts in this field whenever a new record is created, a (in this database) unique number. This Attribute is mutually exclusive to init cont and use old value.

Search equivalent:

is equal with the field content.

Supported search patterns:

>- greater than
<- less than
>=- greater or equal
<=- less or equal

! =- not equal

Conversion into an integer field:

Integer fields only accept the numeric part of the source data. If the source data begin with a non-numeric part, the field will contain 0.
Additional notes:

Float	-	Integer part converted
Boolean	-	"Checked" gets 1, "Unchecked" gets 0
Cycle	-	Labelnumber converted
Slider	-	Level converted
Date	-	First date element (Day) converted
Time	-	First time element (Hour) converted

9.2.3 Float fieldtype

You may enter a real number in a float field.

New Attributes:

Precision: Number of digits after the decimal point.

Search equivalent:

is equal to the field content

Conversion into a float field:

Float fields only accept the numeric part of the source data. If the source data begin with a non-numeric part, the field will contain 0.
Additional notes:

Boolean	-	"Checked" gets 1.0, "Unchecked" gets 0.0
Cycle	-	Labelnumber converted

Note: The precision of the float field type is not very high. It is recommended to use string fields instead. ARexx is also able to make mathematical operations with string fields, if they contain only numerical characters.

9.2.4 Boolean fieldtype

A Boolean field can only contain two values: "True" or "False". It appears in the mask as a "checkbox gadget".

Changed Attributes:

Width: always 3

Search equivalent:

TRUE or 1 - checked field
FALSE or 0 - unchecked field

Conversion into a boolean field:

Boolean fields convert all numbers not equal 0 and TRUE into the checked state. All other values will become the unchecked state.

Under Amiga OS 2.x this field can look a bit strange, because the images are not scalable. Starting with OS 3.0, the size of it is adjusted to the font size.

9.2.5 Cycle fieldtype

Cycle fields have several choices from a freely definable list. It helps to save memory. There is a maximum of 65536 choices. (I hope, thats enough ;-) A cycle field appears in the mask as a "Cycle gadget" (like the name implies).

New Attributes:

Labels: A list of all choices. There must be at least one entry, two entries give the cycle field a meaning.

Search equivalent:

the number of the label counting from zero or the entry itself (type correctly!)

Conversion into a cycle field:

The values will be converted into labels. If there are equal values, they will get the same label. Data are not lost.
Additional notes:
Boolean - "Checked" becomes TRUE(1), otherwise FALSE(0)

9.2.6 Slider fieldtype

A slider is related to a integer field. It can be used to display integer numbers graphically. The numbers may range from -32,768 to 32,767, but may be influenced by several attributes.

New Attributes:

Min. Value: defines the lowest value, which is displayed. It correspondents to the position of the "knob" at the left or at the upper end of the field.

Max. Value: defines the highest value, which is displayed. It correspondents to the position of the "knob" at the right or at the lower end of the field.

Format: is a formatstring in style of the programming language "C". The syntax:

%[-][0][Field][.Maximum][l]Format

- -: The number is left aligned, normally right
- 0: The field is padded with zeroes. e.g.: 1 -¿ 001
- Field: The minimal field width
- Maximum: only for strings, no meaning here.
- l: Says, that the number is 32 bit wide. This is here always the case.
- Format:
  c - Char, the ASCII character for the number is displayed.
  d - The number is displayed.
  u - The unsigned number is displayed.
  x - The number is displayed in hexadecimal format.
  There are also the control characters b and s. These take addresses a arguments and produce only garbage in this case.

The formatting is done with the exec-function RawDoFmt().

MaxFormatLen the maximum length of the format. This region is in the width region. That means that a higher MaxFormatLen makes the field itself smaller.

Search equivalent:

The number itself.

Supported search patterns:

> - greater than
< - less than
>= - greater or equal
<= - less or equal
! = - not equal

Conversion into a slider field:

Slider fields only accept the numeric part of the source data. If the source data begin with a non-numeric part, the field will contain 0. After converting, you should check the range attributes, because they could influence the data.

9.2.7 Date fieldtype

You may enter a date in a date field.

New Attributes:

Init Cont/use current Date: When a new record is created, the current date is copied in this field.

Search equivalent:

is equal to the contents

Conversion into a date field:

Date fields require the data in the format DD.MM.[YYYY]. The single parts must be numbers. If values are non numeric, the part will get ??.
Additional notes:

Integer	-	converted to first element (Day)
Float	-	integer part becomes day, fractional part Month.
Time	-	Hour becomes Day

Currently, Fiasco only displays and reads the date in german format (DD.MM.[YY]YY). No verification of the values is made, this makes values like 65.20.3687 possible.

9.2.8 Time fieldtype

You may enter a time in a time field.

New Attributes:

Init Cont/use current Time: when you create a new record, the current time will be copied in this field.

Search equivalent:

is equal to the content.

Conversion into a time field:

Time field require the data in the format HH:MM:SS. Every element must be a number. If an element is non numeric, it will be 0.
Additional notes:

Integer	-	Converted to hour
Float	-	Integer part converted to hour
Date	-	Day becomes hour

Currently, the time is only displayed with seconds (HH:MM:SS). AM and PM are not supported. No verification of the values is done, that means, that values like 55:66:99 are possible.

9.2.9 Extern fieldtype

A extern field takes a filename, which will be used on request as argument for a user defined program. This makes it possible to define additional data for a record.

New Attributes:

Command: is the name of a program, which is capable to use these files. The characters %s are replaced with the content of the field. If you don‘t use %s, no arguments will be submitted. (e.g. C:ED %s)

Stack: defines the stack size for a command.

Max Chars: defines the maximum length of a filename in chars. This attribute has direct effect on the size of the project file.

FileReq Gadget: select this attribute to have an gadget at the left side of the field, which opens a filerequester to edit the content. Of course, this has only sense, if the command needs filenames.

Search equivalent:

is equal to the content.

Conversion into a extern field:

All fields can be converted without loss of data into an extern field. However, you have to specify a program, which can use these data.
Additional notes:
Boolean - "Checked" becomesTRUE(1), otherwise FALSE(0)
Cycle - Label (Labelnumber) converted
The programs will be called using the AmigaDOS function System(). A console window will be opened for I/O operations.

9.2.10 Datatypes fieldtype

A datatypes field is similar to a extern field. The difference is the use of the datatypes.library. This is the reason, why you can use these fields only starting with Amiga OS 3.0. The major advantage is, that the data will be displayed directly in the mask. A datatypes field is universal usable and freely extensible. A "popup"-gadget at the lower left side of the field makes it possible to edit the contents using an filerequester. If something went wrong, the error will be displayed in the field.

New Attributes:

Max Chars: defines the maximal length of the filename. This attribute has direct effect on the size of the project file.

Scrollbars: Determines, if scrollbars shall be created at the bottom and at the right border of the field. Without a scrollbar, you can view the only the upper left of a file. (That is not completely true. Some datatypes scroll their display, if you click in their area and drag the mouse in the direction of the hidden part. One datatype, which supports that is the picture datatype)

Save gadget: If you activate this option, you will get a second button under the datatypes field. The button will be marked with an S. If you select the button, a file requester will appear, which lets you choose a file, to which the data, which are currently displayed in the field, will be saved to. The data will be written in IFF format.

Display filename: When this option is active, the filename is displayed at the bottom of the field in a string gadget. If you deactivate this option, you cannot edit the value of the field.

Border: If this option is active, Fiasco will Render a border around the field. Do not deactivate this option too often, because there are no visual elements, which mark the beginning and the end of the field.

Defer loading: If you activate this option, the file of the field will not be immediately loaded after activating the record. Instead, the message Deferred will be displayed in the field. Only if you activate the string gadget and hit return, the data will be loaded and displayed.

Immediate play: Select this option to start playing of the data immediately after activating the record. If you activate this option, Defer loading must not be active. Of course, this option has only effect, if the datatype supports playing. These are currently the animation and the sound datatype.

Searchequivalent:

Is equal to the filename; You cannot search the content.

Conversion into a datatypes field:

All fields can be converted without loss of data into a datatypes field. However, the datatypes system requires valid filenames.
Additional notes:
Boolean - "Checked" becomesTRUE(1), otherwise FALSE(0)
Cycle - Label (Labelnumber) converted
The AmigaGuide- and the Animation-Datatype seem to have some problems with relatively small fields.

The changing of records gets slower, because the data have to be loaded each time. To avoid that, use Defer loading.

9.2.11 Text fieldtype

Text fields are no real fields, they only serve to put text in the mask.

Supported Attributes:

Text: will be written in the mask.

Pen: the color, the text shall be written. Normal is normally black and Highlight is normally white. The colors may be influenced by the palette prefs editor.

Bold: Makes the text bold.

Italics: makes the text italic.

Underlined: underlines the text.

No standard attributes are supported!

search equivalent:

You cannot search for a text field

Conversion into a text field:

You cannot convert fields into a text field.

9.2.12 Button fieldtype

Button fields are no real fields, they only serve to put a button for doing a user-definable action in the mask.

Supported Attributes:

Text: will be displayed in the button.

Type: Select here, whether the button shall execute a CLI or an ARexx program. CLI programs may be normal programs, commands or scripts (with the "s" attribute). ARexx programs must be ARexx scripts.

Command: Select here the program, which shall be executed when the button is activated.

Stack: You may specify the stack size for the program here. The standard is 4096. If you specify too less stack for a program, it will crash.

Console Window: lets you specify the I/O stream for the program. It may be a console-window (CON:), the printer (PRT:), a simple file, or, if you don't want any output NIL:.

The button fieldtype only supports the width-standard attribute.

search equivalent:

You cannot search for a button field

Conversion into a button field:

You cannot convert fields into a button field.

Chapter 10

Graphic user interface of Fiasco

Fiasco opens normally after a start only an empty window. You can work in it using pull down menus. The people, who don‘t like pull down menus, may open an additional window using Control/ServiceWindow. This window makes the most important operations accessible via a mouseclick. The third way to make operations are keyboard shortcuts.

The mouse can be used in the mask mode to set the cursor and to drag fields. After a doubleclick on a field, its field requester will be opened (like Field/Edit field...).

Fiasco supports Menuhelp. That means, that if you press the help key, while you browse through the menus, a short description will be displayed in an AmigaGuide window (This feature requires amigaguide.library, which is part of the OS since release 3.0. If you use 2.0 or 2.1, you may get it from the PD).

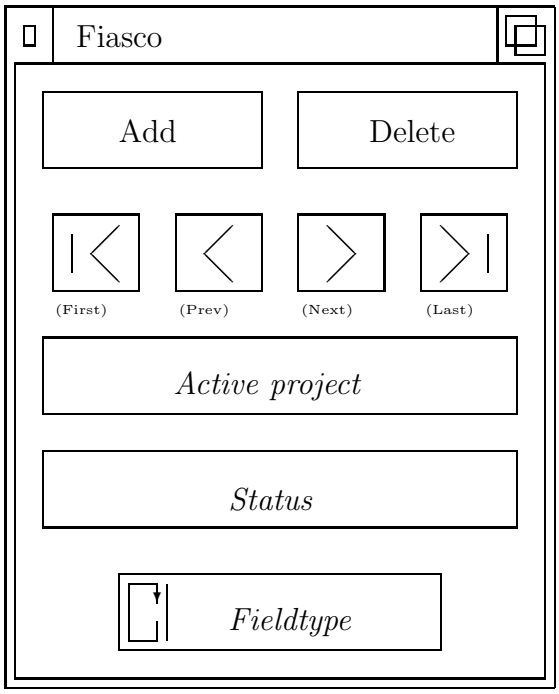
The requesters used by Fiasco have a standard structure. The gadgets at the bottom are for responding. Normally, the left one is a positive response, while the right one is negative. The close gadget of the window is equal to the negative response. Nearly all gadgets in the requesters may be accessed using the keyboard. The positive response may be activated using the Return key, the negative may be activated with Esc.

10.1 Service Window

The service window may be opened or closed with Control/ServiceWindow. If you want Fiasco to open the service window on every program startup,

select the menuitem Settings/Auto-Open ServiceWin (cf. section 10.2.64). Select Settings/Dynamic ServiceWin (cf. section 10.2.65) if you want Fiasco to search for a free place on the screen when Fiasco opens the window. Otherwise, the position of the service window at the time of saving the settings is used.

The service window looks like that:



The gadgets in the ServiceWindow are described in the following:

10.1.1 Add

If the current project is in record mode, a new record will be created. If the mask mode is active, a new field will be created.

Equivalent to:
Record/Add in record mode
resp.
Field/Add field in mask mode

10.1.2 Delete

If the current project is in record mode, the current record will be removed. If mask mode is active, the current field will be remove. *Attention:* This will normally happen without any security request!

Equivalent to:
Record/Remove in record move
resp.
Fields/Remove Field in mask mode

10.1.3 First

If the current project is in record mode, the first record will be activated.

Equivalent to:
Record/First

10.1.4 Previous

If the current project is in record mode, the previous record will be activated.

Equivalent to:
Records/Previous

10.1.5 Next

If the current project is in record mode, the next record will be activated.

Equivalent to:
Records/Next

10.1.6 Last

If the current project is in record mode, the last record will be activated.

Equivalent to:
Records/Last

10.1.7 Active project

The name of the current project is displayed here. If two projects only differ in the path and not in the name, the same name will be displayed.

You may activate another project by activating the window of a project.

10.1.8 Status

Here are status informations displayed.
In the record mode:
number of active record/number of records
A Filter may change these numbers.
In the mask mode:
X: *X position of cursor,* Y: *Y position of cursors*

10.1.9 Fieldtype

If you are in record mode, you can select the fieldtype, which will be used while creating the next new field here. Equivalent to Fields/Field Type

10.2 Menus

The most important interface to the functions of Fiasco are the pulldown-menus of Fiasco's project windows. The functions of all menuitems (from left to right) are described in the following.

10.2.1 Project/New

Shortcut: *A* N

Creates a new project with a mask window. It contains no records or fields. You may create a new database or Open a saved database.
See also: Open

10.2.2 Project/Erase

Shortcut: *A* Z

Erases all data in the current project, the project will be in a status like immediately after calling Project/New. If data have changed since last saving, you will be asked before the data will be erased.

10.2.3 Project/Open...

Shortcut: *A* O

Opens a ASL requester and loads the selected Fiasco project into the current project window. If there are any unsaved data, you will be asked, whether you want to save them first.

If Amiga OS 3.0 is available, Fiasco will increase the buffer-size of the file. This speeds up the load process considerabily.

10.2.4 Project/Options...

Shortcut: *A* $

This menuitem opens the options requester, which can be used for editing projectspecific options. That are:

- Mask stretching (cf. section 4.5)
- Name of author and annotations
- filename of project

The last point makes it possible to change to name of the project without the need to call Save as.

10.2.5 Project/Statistic...

no Shortcut

Shows a few informations for the current project. Example:
One Record requires about 100 Byte RAM. 200 Records of this project need about 19 KByte RAM. There is space for about 2300 more Records.

The memory required for the project and the basic fields is not included.

10.2.6 Project/Reload Rels

Shortcut: *A* !

This item reloads all relations in the current project, like it happened while loading the project. This is particularly useful, if you have deactivated Settings/Update Relations? (cf. section 10.2.61) and changed some keys and want to see the result.

10.2.7 Project/Save

Shortcut: *A* S

Save writes the data of the current project under the same name to disk. If you want to save the project under a different name, you have to use Save as (cf. section 10.2.8) or Options (cf. section 10.2.4) to change the name and then Save.

If Amiga OS 3.0 is available, Fiasco will increase the buffer-size of the file. This speeds up the save process considerably.

10.2.8 Project/Save As...

Shortcut: *A* A

Here you may save the current project under a new name. The name will be requested using an ASL requester and will be kept after saving.

If Amiga OS 3.0 is available, Fiasco will increase the buffer-size of the file. This speeds up the save process considerably.

10.2.9 Project/Import...

Shortcut: *A* I

Opens the import requester (cf. section 10.3.15), the GUI interface for the import function of Fiasco. You can use import to load data from foreign databases into Fiasco.

10.2.10 Project/Export...

Shortcut: *A* E

Opens the export requester (cf. section 10.3.16), the GUI interface for the export function of Fiasco. You can use export to save data in a format, which can be read by other databases.

10.2.11 Project/About...

Shortcut: *A* ?

This item shows a small requester, which gives informations about version, copyright and some system internal data.

10.2.12 Project/Quit

Shortcut: *A* Q

This item closes the current project. If it has been changed and has not been saved yet, you will be asked, whether you want to do this. If this project is the last one, which Fiasco has currently open, Fiasco will exit.

10.2.13 Record/Add Record

Shortcut: *A* +

Adds a new record to the record list of the current project. Each Field contains then it's init cont, which is normally nothing. If the list is open, a new line will be inserted.

If a Filter is active, the new record will be automatically declared valid. If you want that new Records are filtered correctly, you will have to select Compare/Edit Filter again and simply click on Ok.

This menuitem may only be selected in the record mode.

See also: Record/Remove Record

10.2.14 Record/Duplicate Record

Shortcut: *A* 2

Creates an exact copy of the current record. All init cont attributes will be ignored. Even a field with gimme unique Key will contain the old value. That means, that two records with the same "unique" Key will exist.

10.2.15 Record/Delete Record

Shortcut: *A* -

Removes the current record and the data in it. If the are relations, which search for a key, which was defined in this record, they will find in the future nothing.

This menuitem may be selected only in the record mode. If you have selected Setting/Security-Requester?, you will be asked before proceeding.
See also: Record/Add Record, Record/Delete all Records

10.2.16 Record/Delete all Records

Shortcut: *A* Removes all records in the current project. The mask will not be deleted by this function.

Record/Delete all Records may be only called in record mode.
See also: Record/Delete Record

10.2.17 Record/Cut Record

Shortcut: *A* X

Copies the current record to the clipboard and removes it from the record list of it's project. After that, you may use Record/Paste Record (cf. section 10.2.19) to insert it in the project, again.

This function may only be called in record mode.
See also: Record/Copy Record, Record/Paste Record, Section Clipboard support of Fiasco

10.2.18 Record/Copy Record

Shortcut: *A* C

Copies the current record to the clipboard. After that, you may use Record/Paste Record to insert it in the project again.

This function may only be called in record mode.
See also: Record/Cut Record, Record/Paste Record, Section Clipboard support of Fiasco

10.2.19 Record/Paste Record

Shortcut: *A* P

Creates a new record and pastes the contents of the clipboard into that record. Normally, you should call Record/Cut Record (cf. section 10.2.17) or Record/Copy Record (cf. section 10.2.18) before calling this function.

This function may only be called in record mode.

See also: Record/Cut Record, Record/Copy Record, Section Clipboard support of Fiasco

10.2.20 Record/Previous

Shortcut: Cursor up

Activates the record, which is the predecessor of the current. If the current record is the first one, the display will be "beeped". Please note, that Filters change the behavior of this item. In this case, the previous matching record will be activated.

The keyboardshortcut correspondents to the structure of the list, which displays the previous record over the current.

This menuitem may be only selected, if the record mode is active.

See also: Next, First, Last, Goto, Find previous

10.2.21 Record/Next

Shortcut: Cursor down

Activates the record after the current record. If the current record is the last in the list, the display will be "beeped". Please note, that Filters change the behavior of this item. In the case of an active filter, the next matching record will be searched.

The keyboardshortcut correspondents to the structure of the list, which displays the next record under the current.

This menuitem may be only selected, if the record mode is active.

See also: Previous, First, Last, Goto, Find next

10.2.22 Record/First Record

Shortcut: Ctrl Cursor up

Activates the first record of the current project. In the case of an active Filter, for the first matching record will be searched.

This item is only in the record mode selectable.

See also: Next, Previous, Last, Goto

10.2.23 Record/Last Record

Shortcut: Ctrl Cursor down

Activates the last record of the current project. In the case of an active Filter, for the last matching record will be searched.

This item may be only selected in the record mode.

See also: Next, Previous, First, Goto

10.2.24 Record/Goto...

Shortcut: *A* G

Opens the goto requester, which can be used to activate a record using it's number. Please note, that the record number may be changed by adding or deleting records or by using filters.

This item can only be selected in the record mode.

See also: Next, Previous, First, Last

10.2.25 Record/Mark Record

Shortcut: *A* .

Marks the current record. If a record is marked, it will displayed highlighted in the list and the character "M" will be displayed in the service window.

This item can only be selected in record mode.

See also: Unmark Record, Mark all Records, Unmark all Records

10.2.26 Record/Unmark Record

Shortcut: *A* :

Deletes the mark of the current record. It won‘t be displayed highlighted anymore.

This item can only be selected in record mode.

See also: Mark Record, Mark all Records, Unmark all Records

10.2.27 Record/Mark all Records

Shortcut: *A* ,

Marks all records in the current project. Note, that the previous marking of all records will be overwritten.

This item can only be selected in record mode.

See also: Mark Record, Unmark Record, Unmark all Records, Toggle all Marks

10.2.28 Record/Unmark all Records

Shortcut: *A* ;

Clears the marks of all records in the current project. Note, that the previous marking of all records will be overwritten.

This item can only be selected in record mode.

See also: Mark Record, Unmark Record, Mark all Records, Toggle all Marks

10.2.29 Record/Toggle all Marks

No Shortcut

Toggles the marks of all records in the current project. That means, that a marked record will get unmarked and an unmarked will get marked. You can restore the previous marking of the records by calling this menuitem again.

This item can only be selected in record mode.

See also: Mark Record, Unmark Record, Mark all Records, Unmark all Records

10.2.30 Field/Fieldtype

Select in this submenu the current fieldtype. It will be used, if you create fields. The gadget at the bottom of the service window (cf. section 10.1) has the same function.
These fieldtypes are available (with Shortcut):

String	Ctrl S
Integer	Ctrl I
Float	Ctrl F
Boolean	Ctrl B
Cycle	Ctrl C
Slider	Ctrl S
Date	Ctrl A
Time	Ctrl M
Extern	Ctrl E
Datatypes	Ctrl D
Text	Ctrl T
Button	Ctrl U

10.2.31 Field/Add Field...

Shortcut: Return

Opens the field requester (cf. section 10.3.1) for the current field type and inserts the created field at the current cursorposition.

This item can only be selected in the mask mode.

If there is already a field at the current cursor position, nothing will be done.

Please note, that Return is also shortcut for Edit Field. Enter creates a new field, if no field is currently active, otherwise, it opens the requester for editing the current field.
See also: Edit Field, Edit Relations, Remove Field

10.2.32 Field/Edit Field...

Shortcut: Return

Opens the field requester (cf. section 10.3.1) for the selected field. The field requester can be used to change several attributes of the field. If certain changes would cause loose of data (e.g. changing max chars of a string field

to a low number), you will be informed about the problem and have to possibility to cancel the change. Field types may not be changed this way. You have to use Convert Field (cf. section 10.2.37).

Please note, that Return is also a shortcut for Add Field. Return calls Add Field, if no field is active, and otherwise Edit Field.

This item can only be selected in the mask mode.
See also: Add Field, Edit Relation

10.2.33 Field/Duplicate Field

No shortcut

Makes an exact copy of the active field. It will be placed as near as possible to the original field. The ID will be copy_of_*FieldID*.

10.2.34 Field/Remove Field

Shortcut: Del

Removes the selected field. All data in this field will be also lost. Relations or ARexx scripts, which refer to this field, will be not functional. *Attention:* The relations or ARexx scripts will not complain immediately after removing the field, but at the first activation.

This item can only be selected in the mask mode.
See also: Edit Field, Edit Relations, Add Field

10.2.35 Field/Edit Relation...

Shortcut: *A* &

This item opens the relation requester (cf. section 10.3.12), which adds a relation (cf. section 7.1) to the current field.

This item can only be selected in the mask mode.
See also: Field/Remove Relation

10.2.36 Field/Remove Relation

Shortcut: *A* 0

This item deletes all relation informations of the active field. After that, the data in this field will be written into the normal file.

This item can only be selected in the mask mode.

10.2.37 Field/Convert Field...

Shortcut: *A* "

Opens the convert requester (cf. section 10.3.2) for the selected field. Using convert, you may change the type of a field in a smart way.

This item can only be selected in mask mode.
See also: Add Field, Edit Field

10.2.38 List/Hide column

Shortcut: *A* [

Hides a activated column of the List. You activate a column by clicking in the topmost line of the list, which contains the field IDs. After hiding a column, the columns at the right side of it will be shifted to the left. The column may be made visible again by using Show column (cf. section 10.2.39).

This item may only be selected, if the list window is open.

10.2.39 List/Show column...

Shortcut: *A*]

This item opens a requester, which may be used to reveal the columns, which have been hidden with Hide column (cf. section 10.2.38). Fiasco tries to place the columns as near as possible at their old positions.

This item may only be selected, if the list window is open.

10.2.40 List/Show all columns

no shortcut

Makes all columns, which have been hidden using Hide column (cf. section 10.2.38), visible again.

This item may only be selected, if the list window is open.

10.2.41 List/Recalc List

Shortcut: *A* %

This menuitem calculates all positions and dimensions of the columns in the list. Hidden columns are not revealed.

This item can be compared with Clean up of the Workbench.

This item may only be selected, if the list window is open.

10.2.42 Compare/Find...

Shortcut: *A* F

Opens the search requester (cf. section 10.3.3), which can be used to define search criterions.

This item is only selectable, if the record mode is active and if the current project contains at least one record.
See also: search requester, Find next, Find previous

10.2.43 Compare/Find next

Shortcut: *A* >

Activates the *next* record, which matches with the search criterions, which have been specified using the search requester (cf. section 10.3.3). If no matching record is found, you will be informed about that.

This item is only selectable, if the record mode is active and if the current project contains at least one record.
See also: Search requester, Find, Find previous

10.2.44 Compare/Find previous

Shortcut: *A* <

Activates the *previous* record, which matches with the searchcriterions, which have been specified with the search requester (cf. section 10.3.3). If no record is found, you will be informed about that.

This item is only selectable, if the record mode is active and if the current project contains at least one record.
See also: Search requester, Find..., Find next

10.2.45 Compare/Replace...

Shortcut: *A* R

Opens the replace requester (cf. section 10.3.4), which can be used for replacing data.

This item is only selectable, if the record mode is active and if the current project contains at least one record.

10.2.46 Compare/Count...

Shortcut: *A* #

Opens the count requester (cf. section 10.3.5), which can be used to determine the number of the records matching with the specified pattern.

This item is only selectable, if the record mode is active and if the current project contains at least one record.

See also: Find

10.2.47 Compare/Sort...

Shortcut: *A* =

Opens the sort requester (cf. section 10.3.6), which may be used to sort the records of the current project.

This item is only selectable, if the record mode is active and if the current project contains at least one record.

10.2.48 Compare/Edit Filter...

Shortcut: *A*

Opens the filter requester (cf. section 10.3.7), which can be used to create Filters.

This item is only selectable, if the record mode is active and if the current project contains at least one record.

10.2.49 Compare/Use Filter?

Shortcut: *A* ‘ (Gray key at the upper left of the keyboard)

This item can be used to switch the filter (cf. section 6.6) on or off. If no filter has been created yet, the filter requester (cf. section 10.3.7) will be opened.

This item is only selectable, if the record mode is active and if the current project contains at least one record.

10.2.50 Compare/Mark...

Shortcut: *A* K

Opens the mark requester (cf. section 10.3.8), which can be used to mark specific records, which match a pattern. This works much like the creation of filters.

Existing marks will be overwritten; marked records will be unmarked, if they not match.

This item is only selectable, if the record mode is active and if the current project contains at least one record.

10.2.51 Compare/Filter to Marks

No Shortcut

Converts the current filter (cf. section 6.6) (active or inactive) to marks. Records, which match the filter, will be marked and records, which don‘t match the filter, will be not marked. If the filter is active, it will be deactivated.

See also: Compare/Edit Filter, Compare/Marks to Filter

10.2.52 Compare/Marks to Filter

No Shortcut

Converts the marking of the current project into a filter (cf. section 6.6). Each marked record will be declared as valid record, every record, which is not marked, will be filtered out. This filter will not be copied into the filter requester. If you open the filter requester and proceed with Ok, the filter created with Marks to Filter will be overwritten.
See also: Compare/Filter to Marks

10.2.53 Control/Record Mode

Shortcut: *A* D

This item switches the current project to record mode (cf. section 4.6.1), in which records and the contents of records can be changed. If this mode is *active* , a checkmark will be set at the left side of the item.
See also: Record mode, Mask mode

10.2.54 Control/Mask Mode

Shortcut: *A* M

This item switches the current project to mask mode (cf. section 4.6.2), in which the mask can be changed. If this mode is *active* , a checkmark will be set at the left side of the item.
See also: Mask mode, Record mode

10.2.55 Control/ServiceWindow

Shortcut: *A* W

This item controls the service window (cf. section 10.1). If it is checked, the service window is open. The service window makes the most important record- and mask-operations easier and displays some status information.

The service window serves globally for all projects.

10.2.56 Control/ListWindow

Shortcut: *A* L

This item controls the list window, if it is checked, the list is open.
Each project may have it's own list window.

10.2.57 Control/ARexx-Debug

Shortcut: *A* B

This activates a special debug mode of Fiasco for the ARexx interface. If Fiasco commands fail, Fiasco will create an requester, which contains more detailed informations about the error.

10.2.58 Settings/Create Icons?

If this item is checked, Fiasco will create icons while saving projects.

10.2.59 Settings/Create Backups?

This item determines, whether Fiasco creates backups of old projects while saving new projects. The backup file will be named *oldname*.bak.

10.2.60 Settings/Write Relations?

If this item is checked, Fiasco will also write relations back in their "there" projects. Otherwise, changes made in these fields will be lost. This item should be only active, if Update Relations? (cf. section 10.2.61) is also active, or if you call Project/Reload Rels (cf. section 10.2.6) before saving. Otherwise you risk, that data in the "there" project will be overwritten by invalid data in some fields of the "here" project.

10.2.61 Settings/Update Rels?

This item determines, whether relations are updated immediately after the input of a new key. This requires disk accesses, which may become annoying if Fiasco has to read the data from a floppy disk. If you deactivate this item, you should also deactivate Write relations? (cf. section 10.2.60), because there may be invalid data in the project, which would be written into the

"there" file. If you want to see the changes, you can update the relations using Project/Reload Rels (cf. section 10.2.6).

10.2.62 Settings/Use * as Pattern?

Activate this item to activate the support of the asterisk as a valid search-pattern. The * has then the same meaning as #?.

10.2.63 Settings/Security-Reqs?

If this item is checked, Fiasco will warn you before deleting any fields or records. This can prevent erroneous deleting and loss of data.

10.2.64 Settings/Auto-Open ServiceWin?

If this item is checked, the service window (cf. section 10.1) will be opened on every program start automatically.

10.2.65 Settings/Dynamic ServiceWin?

If this item is checked, the service window will be opened at an free place. Otherwise, fixed coordinates will be used.

10.2.66 Settings/Talking?

Check this item, if you want Fiasco to use the narrator.device to "speak" certain messages.

10.2.67 Settings/Display...

no shortcut

Opens the display requester, which can be used to specify display options for Fiasco. You can select here, whether Fiasco shall open it's windows on a public screen or on an own custom screen. Furthermore, you may select fonts for the screen and the mask.

The latter replaces the menuitem Settings/Choose Font of Fiasco 1.0.

10.2.68 Settings/Editor...

no shortcut

Opens a requester, which lets you specify a editor program, which will be called by Fiasco, if you select the Edit Script button in the fieldrequesters.

10.2.69 Settings/Save Settings

Saves the current program settings in the files "env:fiasco.prefs" and "envarc:fiasco.prefs". The settings "survive" rebooting.

10.2.70 Settings/Save Settings as...

Saves the settings in a file, which has been specified with an ASL requester. If you save the file in "env:", the settings won't survive an reboot. If you save them only in "envarc:", they will get first active after rebooting, because Fiasco searches for it's current settings only in "env:".

10.2.71 Settings/Load Settings...

Loads a specified settings file and uses them. To use them also after reboots, you should select Save Settings to write them to "env:" and "envarc:".

10.2.72 User/Edit...

Shortcut: **A** U

Opens the Usermenu requester (cf. section 10.3.9), which can be used to define Usermenus.

10.3 All requesters

Requesters are used by Fiasco to get information, that is required for certain operations. Normally, the requesters are created after selecting a menuitem of Fiasco. So called EasyRequesters, which are used by fiasco to request a simple choice are not explained here, because they are mostly easy to understand and described in function specific sections.

Most requesters can be controlled by using the keyboard. The shortcuts, which are marked with an underscore, are usually single characters without a qualifier.

The gadgets at the lower bottom of a requester are usually for proceeding. Normally, the leftmost is a positive response (Ok), while the rightmost is a negative response (Cancel). The positive response, which has an additional frame, has Enter as shortcut. The negative has Escape as shortcut.

10.3.1 Field requester

The field requester can be used to change the attributes of a field. Each fieldtype has a different field requester, because the gadgets of the field requester represent the supported attributes of each fieldtype. The supported attributes are listed with the field documentations.

The field requester will show up, if you call Add Field (cf. section 10.2.31), Edit Field (cf. section 10.2.32) or doubleclick on a field.

If you proceed with Ok, all values will be checked for validity. If one value cannot be used by Fiasco, a requester will explain the problem.

A small summary of the conditions: (presumed, that these attributes exist)

- There must be an Id.
- MaxChars must be ¿ 0.
- Width must be ¿ 2.

If dimensionvalues cannot be used, because other fields are too near at the field, another requester appears, which has the gadgets Shift, Squeeze and Cancel. Cancel does nothing, it only returns to the field requester. Squeeze makes the field fitting in the space. Shift shifts the field to the left, to make it fitting. It is not always possible to Shift.

If you change a already existing field, which stores it‘s contents in strings and supports MaxChars (currently String, Extern and Datatypes), an additional control is implemented. If you change "MaxChars" to a value, which does not allow to keep all strings in their original length (that means, some strings are longer), you will be asked, whether you really want to cut these strings or to keep the old value.

10.3.2 Convert Field requester

The convert field requester can be used to change the type of one field without the need of an ARexx script.

Field ID: This text gadget displays the ID of the field, which will be converted. Please check here, whether you have called Convert Field for the correct field.

old Type: Displays the current type of the field.

new Type: Select here the new type of the field. Please note, that conversions between certain fieldtypes may loose data. Consult the field documentation for more information on this topic.

alternative format: Select this checkbox to active an output-format, which differs from the normal format. Please see the field docs, whether this gadget has any effect and if, what effect.

Ok: Starts the conversion and exits then.

Cancel: Simply closes the requester.

10.3.3 Search requester

Field: Select here the field, which will be searched. The listview displays only the ids of real fields, buttons and textfields are not displayed. Only one field can be selected.

Pattern: Enter here the pattern to search. It may be a simple value or one with Patterns. This value will be also used in the count and replace requesters.

Blurred search/Activated: If you want to use "blurred search" (cf. section 6.2), you will have to activate this gadget.

Blurred search/factor: You can control here the tolerance of the "blurred search". 0 searches only for exactly matching entries, 100 searches for almost all entries.

Next: Proceeds with searching for the next matching entry and activates it.

First: Searches for the first matching entry.

Previous: Searches backwards for the next matching entry.

Cancel: Closes the requester without any further action.

10.3.4 Replace requester

If you already know the search requester, you should have no problems with this one.

Field: Select here the field, which will be searched. The listview displays only the ids of fields, textfields are not displayed. Only one field can be selected.

Pattern: Enter here the pattern to search. It may be a simple value or one with Patterns. This value will be also used in the count and replace requesters.

Replacement: Enter here a value, which will be copied in the matching entries. No patterns are possible.

Confirm: If you want to be asked for every replacing operation, you should select this gadget.

Blurred search/Activated: If you want to use blurred search, you will have to activate this gadget.

Blurred search/factor: You can control here the tolerance of the blurred search. 0 searches only for exactly matching entries, 100 searches for almost all entries.

10.3.5 Count requester

This requester lets you count records, which match with a patterns. More on counting in section 6.4.

You can open this requester using Compare/Count (cf. section 10.2.46).

If you already know the search requester, you should have no problems with this one.

Field: Select here the field, which will be searched. The listview displays only the ids of fields, textfields are not displayed. Only one field can be selected.

Pattern: Enter here the pattern to search. It may be a simple value or one with Patterns. This value will be also used in the search and replace requesters.

Blurred search/Activated: If you want to use blurred search, you will have to activate this gadget.

Blurred search/factor: You can control here the tolerance of the blurred search. 0 searches only for exactly matching entries, 100 searches for almost all entries.

Ok: proceeds and counts the matching records. The number will be displayed at the end.

Cancel: closes the requester without any further action.

10.3.6 Sort requester

You can sort a Fiasco project in the sort requester in respect to one field. It can be opened using Compare/Sort (cf. section 10.2.47).

Sort by: A list of all fields is displayed here. You should select a field here, which will be used as orientation while searching.

Descending: Select this gadget to sort the data from high values to low values (i.e. Z, Y, X, ..., C, B, A)

Ok: begins with sorting. The previously active record will be kept active, but it is highly probable, that the number of the record will change.

Cancel: Closes the requester without any further action.

10.3.7 Filter requester

Filters (cf. section 6.6) offer the possibility to create an overview over a group of records. A filter creates the impression of a database, which consists only of the matching records. Filters are not created during the normal program functions, but it is created immediately after confirming the filter requester. That means, records which will be added to project, during a filter is active, will be displayed regardless of their contents. The same rules for changes records.

The filter requester may be reached using Compare/Edit Filter.

If you already know the search requester (cf. section 10.3.3), you should have no problems with this one.

Field: Select here the field, which will be searched. The listview displays only the ids of fields, textfields are not displayed. Only one field can be selected.

Pattern: Enter here the pattern to search. It may be a simple value or one with Patterns.

Blurred search/Activated: If you want to use blurred search, you will have to activate this gadget.

Blurred search/factor: You can control here the tolerance of the blurred search. 0 searches only for exactly matching entries, 100 searches for almost all entries.

Ok: creates the filter. The project will seem, as if would consist only of matching records.

Cancel: closes the requester without any further action.

10.3.8 Mark requester

The mark function of Fiasco provides a possibility to remember about a specific record. The mark requester has the purpose to mark all records, which match with a specific pattern. This requester is highly related to the filter and search requesters (cf. section 10.3.3).

The mark requester may be opened with Compare/Mark.

Field: Select here the field, which will be searched. The listview displays only the ids of fields, textfields are not displayed. Only one field can be selected.

Pattern: Enter here the pattern to search. It may be a simple value or one with Patterns.

Blurred search/Activated: If you want to use blurred search, you will have to activate this gadget.

Blurred search/factor: You can control here the tolerance of the blurred search. 0 searches only for exactly matching entries, 100 searches for almost all entries.

Ok: marks the matching records. The old marking of the records will be lost!

Cancel: closes the requester without any further action.

10.3.9 Usermenu requester

Fiasco has the ability to create own menuitems and to put CLI programs or ARexx scripts behind them. The defined items may be selected apart from the classic way with the mouse with the F-Keys. F1 to F10 correspondent to the first ten items, Shift and F1 to F10 correspondent to the items 11

to 20. If you want to define more than 20 items, you will have to select the additional items with the mouse.

Furthermore, Intuition limits the number of definable items to 63. If you try to define more items, they will be cut.

The items may be saved using Settings/Save Settings (cf. section 10.2.69).

Items: This is the list of all existing menu items. With Add you can add one, with Del you can delete one and < and > serve to change the position of the item.

Type: Select here, whether you want to call a program or a ARexx script.

Command: Select here the program or the ARexx script respectively, which shall be executed.

10.3.10 Option requester

The option requester contains settings, which concern the project. It may be opened using the menuitem Project/Options or the ARexx command F_OptionsReq.

Name: Here you can change the filename of the project. The project will be saved under this name in future. After saving, all direction-relative operations will use the new directory.

Author: You can use this field to enter your own Name! It will be stored at the beginning of the project file.

Annotations: Yet another gadget for free use. You may store any notes here, for example a version string (with $VER: at the beginning). It will be written to the project file just before the author.

stretch X / Y: These values are added to the with or height of the cursor. The effect of this operation is a stretching of the mask in X- or Y-direction. More on stretching in section 4.5.

10.3.11 Goto requester

The goto requester is one of the simplest requesters in Fiasco at all. It may be opened with Record/Goto and offers the possibility to activate a record using it's number.

Please note, that Filters change the record numbers.

go to: Takes the number of the record.

Ok: Proceeds and activates the record with the number.

Cancel: Oh sorry, i just forgot... %-)

10.3.12 Relation requester

This is the main interface for relation (cf. section 7.1) handling in Fiasco. It may be opened using Field/Edit Relations.

Key here: Select here the key in the current project.

Real here: displays the ID of the field, whose relations are just now edited.

Key there: Select here the field of the project, which has been specified under Related File, which contains the Key. This listview displays only fields, which can contain the Key (with same type).

Real there: Select here the field of the project, which has been specified under Related File, which is supposed to be the counterpart of Real here. This field is used to read the Data, which will be displayed in Real here. This listview displays also only fields, which look as if they could contain the data (type and max chars must be equal).

Related File: Select here the project file relative to the directory of the current project, which contains the informations.

Ok: Proceeds with loading the relations. If any errors occur while loading, the requester will be activated again, otherwise it will return to the main window.

Cancel: closes the requester without any further action.

10.3.13 Show column requester

This requester, which may be reached with List/Show column, displays the currently hidden columns in the list. If you select one and click on Ok, the column will be inserted in the list at it‘s old position.

Field: All hidden columns are displayed here. Select here the column, which you want to be revealed.

Ok: Inserts the column and redisplays the list.

Cancel: Closes the requester.

10.3.14 Display Options Requester

This requester controls the display elements of Fiasco. You can open an own screen for Fiasco and choose the fonts for the custom screen and for the mask (Fiasco 1.0 had the menuitem Choose Font for this purpose).

Screen

Screentype: Select here, whether you want to use a public screen or an own custom screen.

PubScreen Name: Specify here the name of the public screen, you want Fiasco to open it's windows on it. This has only effect, if you select at Screentype the choice PublicScreen. If you leave this gadget empty, Fiasco will use the default public screen.

Screen Mode: You may select here the display mode for the custom screen. Clicking on the popup gadget will open an ASL screenmode requester. This requires asl.library version 38 or higher.

Screen Font: This gadget controls, whether you want to use an own font for the custom screen or the Workbench screen font, which is controlled by the Font Preferences.

Custom Font: If you want to use an own font for the custom screen, you may select it here.

Mask Font

Mask Font: This gadget controls, whether you want to use an own font for the mask or the System default font, which is controlled by the Font Preferences.

Custom Font: You may select here an own font for the mask. It must be fixed width.

Ok: Proceeds with resetting the display of Fiasco. If required, a new screen is opened, and so on.

Cancel: Closes the requester without any further action.

10.3.15 Import requester

The import requester is the GUI interface to the import function (cf. section 8) of Fiasco. Import makes Fiasco able to read data from other databases programs. Most times, this cannot be done directly, but the foreign database

has to "export" the data. You may specify various parameters for importing, so you should be able to read nearly all import-/export-formats into Fiasco.

The Fiasco distribution contains several predefined import formats, which can be loaded using the Load button at the bottom of the import requester.

The values, that may be typed in the gadgets of the import requester, are described in the Import/Export section of this document.

File: Specify here the file, which contains the data to import. You may use the picker button at the right side to select it using an ASL requester.

View: Click here, if you want to view the contents of the file. Fiasco will start asynchronously More or MultiView, if available.

Records/Start: Enter here the start characters for records. Default: Empty.

Records/End: Enter here the characters at the end of a record. Default: Empty.

Records/Separator: Enter here the characters between two records. Default: \n.

Fields/Start: Enter here the characters, fields start with. Default: ".

Fields/End: Enter here the characters, fields end with. Default: ".

Fields/Separator: Enter here the characters between two fields. Default: \t.

Misc/Skip Lines: Enter here introducing characters for remarks. Default: Empty.

Misc/Start skip: Enter here the number of lines, which shall be skipped at the start. Default: 0.

Misc/Max fields: Enter here the maximum number of fields in a record. Can also be used, if record separators are missing. Default: 100.

Options/First record contains IDs: Activate this gadget, if the first record of the file contains the IDs of the fields in the project. They will be used by Fiasco then instead of generic IDs.

Options/Append new fields: Activate this, if you want Fiasco to create new fields for your data and not to use existing ones. If you have an entirely empty project, you should activate this option.

Options/Overwrite old project: Removes the old data in the current project window. If you do not select this, you data will be appended somehow to the existing project.

Ok: Starts the import process. Note that Fiasco may run out of memory due to bad structure parameters and too big files. Programs, which have problems with low memory should not run during this process.

Save: Saves the current settings in a specified file.

Load: Reads the settings from a specified file and sets them up in the requester.

Cancel: Closes the requester without any further action.

10.3.16 Export requester

The export function provides a capability to share the data, which were created using Fiasco with other databases, which cannot read the normal format of Fiasco databases. See the Import/Export section of this document for more information about this mechanism.

File: Specify here the name of the file, the data shall be written to. If a file already exists with this name, it will be overwritten.

Records/Start: Enter here the start characters for records. Default: Empty.

Records/End: Enter here the characters at the end of a record. Default: Empty.

Records/Separator: Enter here the characters between two records. Default: `\n`.

Fields/Start: Enter here the characters, fields start with. Default: ".

Fields/End: Enter here the characters, fields end with. Default: ".

Fields/Separator: Enter here the characters between two fields. Default: `\t`.

Options/First record contains IDs: Activate this gadget, if you want Fiasco to write the field IDs in the first record.

Options/Marked records only: Activate this gadget, if you want Fiasco to write only records, which are marked.

Ok: Click here to start the export process.

Save: Saves the structure parameters in a selected file.

Load: Loads the structure parameters in a selected file.

Cancel: Closes the requester without any further action.

Chapter 11

ARexx

ARexx is a macro programming language, which is capable to connect different programs. ARexx has been developed by William S. Hawes and is part of the system software since OS 2.0.

The ARexx port of Fiasco may be accessed externally out of a script, or ARexx scripts can be called by Fiasco. For example. this happens if you specify in the field attribute Script an ARexx script and you change the contents of a field. These scripts may react on the change and can adjust the value of another field or can do something else.

To be able to communicate with Fiasco, you have to add the line Address FIASCO to the script.

Nearly all operations, which can be used with the GUI of Fiasco, can be used with the ARexx commands. Additional, the functions of Fiasco may be extended with ARexx. There are many ARexx commands, which do exactly the same as their GUI "brothers". That means, that certain commands may open a requester under certain conditions. It is often possible to circumvent this problem. It will be fixed sometimes in the future. There are also commands, which open a requester in any case. This may be also useful for scripts, but has been primary implemented to give Fiasco a second menu (Iconbars). I have experimented already with ToolManager-Docks. Unfortunately, this was too slow for fast browsing in a database.

11.1 ARexx and Fiasco in general

A Fiasco command returns in the case of success in RC 0. If a command had problems, because it's environment was not proper, 5 is returned. More serious errors, like missing arguments, return 10. Fatal errors return 20.

Parameter are separated by white spaces. If single arguments are supposed to contain spaces, it does not work to enclose them simply in quotation-marks. This is caused by ARexx, which swallows all quotation-marks. To avoid this, you should enclose the marks in the other marks. (e.g. F_Open ' "Test Datei" ') You have to use the single quotes at the outer position, because Fiasco only handle double quotes. Be sure, not to use variables inside of any quotations. To use them, you have to close the quotation, write the variable and open the quotation again, if required. These issues do not apply for argument, which have the /F modifier. If a command returns a value, this is stored in RESULT. To use RESULT, you have to put an OPTIONS RESULTS at the beginning of a script.

The debugging of ARexx scripts is a bit problematic. Scripts, which have been activated using the user menu or fields, have no output stream. All errormessages will be swallowed. If you want to test ARexx scripts, you should run the scripts from the shell (using rx *filename*; Fiasco must be in the correct status). To get more information, why a command failed, which has been sent to Fiasco, you should activate the item Control/ARexx Debug. Fiasco will show a requester with a explanation for the reason of the error. The script won't continue until the requester has been closed. You have two choices there: Continue returns the correct error code, Ignore Error returns 0 in RC, which looks like the command has succeeded. An additional choice is Help, which won't proceed, but display the help text for the command, which failed.

The style of the documentation of the commands is similar to the Amiga OS Autodocs. Synopsis defines a template.

11.2 ARexx Commands

11.2.1 F_AboutReq

Name: F_AboutReq – Open the "About" requester

Synopsis: F_AboutReq

Function: Does exactly the same as Project/About (cf. section 10.2.11).

Inputs: none

Results: none

11.2.2 F_ActivateField

Name: F_ActivateField – activate the field in the GUI

Synopsis: F_ActivateField Field/A
rc = *Success*

Function: Activates the field with the specified ID in the mask. Only fields, which appear as a string/longint gadget may be activated. If project or window is not active, the field cannot be activated. This command may be only called in mask mode.

Inputs: Field - ID of field to activate

Results: rc = 0, if field has been activated

See also: intuition.library/ActivateGadget()

11.2.3 F_AddFieldReq

Name: F_AddFieldReq – open the add field requester

Synopsis: F_AddFieldReq

Function: This command does exactly the same as Field/Add Field (cf. section 10.2.31). It may be only called in mask mode.

Inputs:

Results:

See also:

11.2.4 F_AddRecord

Name: F_AddRecord – Add a new record.

Synopsis: F_AddRecord

Function: Add to the current project a new record. This record will get active.

This function may only be called in record mode.

Inputs: none

Results: none

See also: F_RemRecord, Record/Add Record

11.2.5 F_ClearProject

Name: F_ClearProject – clear the active Project

Synopsis: F_ClearProject Force/S

Function: Deletes all data in the current project. It will be in a state much like after a New.

If you do not specify Force, this command does exactly the same as Project/Erase (cf. section 10.2.2). That means, it is possible, that a requester opens, which asks you whether you want to save the current project before proceeding or cancel.

To prevent this, specify the Force parameter. This will suppress all warnings. To find out, wheter the project is not saved, you may use F_IsVirgin (cf. section 11.2.31).

Inputs: Force – suppress all warnings

Results: none

See also: Project/Erase, F_IsVirgin, F_MakeVirgin

11.2.6 F_CloseList

Name: F_CloseList – close the list window

Synopsis: F_CloseList

Function: This command is equal to deactivating the menuitem Control/List.

Inputs:

Results:

See also: Control/List

11.2.7 F_CloseServiceWin

Name: F_CloseServiceWin – close the service window

Synopsis: F_CloseServiceWin

Function: Closes the service window. If the window is not open, nothing happens.

Inputs: none

Results: none

See also: F_OpenServiceWin

11.2.8 F_ConvertField

Name: F_ConvertField – change the type of a field

Synopsis: F_ConvertField Field/A,NewType/A,AltFormat/S

Function: Changes the type of the named field. You cannot convert text or button fields. May be only called in mask mode.

Inputs: Field - ID of field
NewType - New Type of field. (e.g. String)
AltFormat - Specify, if you want an alternative Format

Results: none

See also: Chapter Converting Fields

11.2.9 F_CountRecs

Name: F_CountRecs – count the records

Synopsis: F_CountRecs
Result = *Number_of_Records*

Function: Counts the records, which are currently in the current project. May be only called in record mode.

Inputs: none

Results: Number_of_Records - The number of records, may be zero. Note, that Filter influence this value.

See also:

11.2.10 F_CountReq

Name: F_CountReq – Open the count requester

Synopsis: F_CountReq

Function: Does exactly the same as Compare/Count (cf. section 10.2.46). May be only called in record mode.

Inputs:

Results:

See also: Compare/Count...

11.2.11 F_DupRec

Name: F_DupRec – Clone the active record

Synopsis: F_DupRec

Function: This command duplicated the active record exactly. All the Init Cont attributes are ignored. This command does exactly the same as Record/Dup Record (cf. section 10.2.14). May be only called in record mode.

Inputs:

Results:

See also: Records/DupRecord

11.2.12 F_Export

Name: F_Export – export ata out of Fiasco

Synopsis: F_Export File/A,RecStart/K,RecEnd/K,RecSep/K,FieldStart/K, FieldEnd/K,FieldSep/K,FirstRecIDs/K,MarkedOnly/S
rc = *Success*

Function: Calls the export function of Fiasco. See the Import/Export chapter for more information about exporting. If you do not specify a parameter, it will be empty.

Inputs: File - File to write
RecStart,RecEnd,RecSep,FieldStart,FieldEnd,FieldSep - structure parameters
FirstRecIDs - First Record will contain field IDs
MarkedOnly - Exports only marked records

Results: rc = 0, if everything went well.

See also: F_Import, Chapter Import/Export

11.2.13 F_FilterReq

Name: F_FilterReq – open the filter requester (cf. section 10.3.7)

Synopsis: F_FilterReq

Function: Does exactly the same as Compare/Filter (cf. section 10.2.48). May be only called in record mode.

Inputs:

Results:

See also: Compare/Filter

11.2.14 F_FindFirst

Name: F_FindFirst – Search for a pattern

Synopsis: F_FindFirst Field,Blur/K,Pattern/F
Result = *Number_of_Record*

Function: Searches for the first matching with the pattern, which has been either set with F_SetSearchPat or using the arguments. If rc is equal zero, Result is equal to the number of the found record. This may be accessed using F_GotoRec. If nothing is found, 5 is returned.

Inputs: Field - ID of the Field to search
Blur - Factor for blurred search. Specifying activates it.
Pattern - Standard search pattern.
If you don‘t specify Field or Pattern, the values will be used, which have been previously used in the search requester or have been set by F_SetSearchPat and F_SetSearchField.

Results: rc = 0: result = Number of matching record.
rc = 5: nothing found or no pattern.

Example:

```
/* Find-Example.rexx */
options results
address FIASCO

count = 0

F_FindFirst "Test" "?#?"   /* search for the first record in
```

```
                                * which the field with the ID Test
                                * is not empty */

    do while rc = 0            /* Continue searching until
                                * nothing is found */

       F_GotoRec Result        /* activate the found record */

       count = count + 1

       F_FindNext "Test" "?#?" /* search for next */

    end

    /* All records done */
```

See also:

11.2.15 F_FindNext

Name: F_FindNext – Search for a pattern.

Synopsis: F_FindNext Field,Blur/K,Pattern/F
Result = *Number_of_next_Record*

Function: Searches for the next matching with the pattern, which has been either set with F_SetSearchPat (cf. section 11.2.59) or using the arguments. If it succeeds (rc = 0), Result contains the number of the found record. The record may be activated using F_GotoRec (cf. section 11.2.27).

Note: The active Record is not searched by F_FindNext and F_FindPrev. If you want to write a program, which searches all records, you have to call at first F_FindFirst (cf. section 11.2.14) and then F_FindNext.

Inputs: Field - ID of the Field to search
Blur - Factor for blurred search. Specifying activates it.
Pattern - Standard search pattern.
If you don‘t specify Field or Pattern, the values will be used, which have been previously used in the search requester or have been set by F_SetSearchPat and F_SetSearchField.

Results: If rc = 0, result = recordnumber of next matching.
If rc = 5, nothing found or no pattern

Example: see F_FindFirst

See also:

11.2.16 F_FindPrev

Name: F_FindPrev – Search for a pattern backwards.

Synopsis: F_FindPrev Field,Blur/K,Pattern/F
Result = *Number_of_prev_Record*

Function: Searches for the previous matching with the pattern, which has been either set with F_SetSearchPat (cf. section 11.2.59) or using the arguments. If it succeeds (rc = 0), Result contains the number of the found record. This may be activated using F_GotoRec (cf. section 11.2.27).

Inputs: Field - ID of the Field to search
Blur - Factor for blurred search. Specifying activates it.
Pattern - Standard search pattern.
If you don‘t specify Field or Pattern, the values will be used, which have been previously used in the search requester or have been set by F_SetSearchPat and F_SetSearchField.

Results: if rc = 0, result contains the recordnumber of previous matching
if rc = 5, nothing found or no pattern

Note: F_FindPrev is not very handy in ARexx scripts. You should use combinations of F_FindFirst (cf. section 11.2.14) and F_FindNext instead.

See also:

11.2.17 F_FindReq

Name: F_FindReq – open the search requester

Synopsis: F_FindReq

Function: Opens the search requester (cf. section 10.3.3). This command does exactly the same a Compare/Find (cf. section 10.2.42). The command may be only called in record mode.

Inputs: none

Results: none

See also:

11.2.18 F_GetFieldAttributes

Name: F_GetFieldAttributes – Read the attributes of a field

Synopsis: F_GetFieldAttributes Field/A,X/S,Y/S,W=Width/S,H=Height/S, Rexx/S,Type/S,ListX/S,ListW/S,MaxChars/S,InitCont/S,OwnInit/S, Labels/K/N,Commands/S,Stack/S
rc = *Success*
Result = *Attribute_Value*

Function: Reads one attribute of the specified field. The value of the attribute is returned in Result. Not every fieldtype supports all attributes, if a type does not support a particular attribute, rc will be not equal 0. You may only specify one attribute while calling this command. For convenience, this command may be called both in record mode and in mask mode.

This command may be also called in virtual state.

Input: Field - ID of a field. Always required.
X - I want to know the top edge of field in cursors
Y - Left edge of field in cursors
W - Width of field in cursors
H - Height of field in cursors
Rexx - Name of ARexx script assigned to field
Type - Type of field (e.g. string, integer, etc.)
ListX - Left edge of field in list, -1 if field is hidden
ListW - Width of field in list, -1 if field is hidden
MaxChars - MaxChars attribute
InitCont - InitCont attribute. One of own, old, key OwnInit - Own initial content
Labels - Returns the label of the specified number
Command - Command attribute of field
Stack - Stack attribute of field

Results: rc - zero, if successful.
Result - contains requested attribute, if rc = 0

See also: Field documentation

11.2.19 F_GetFieldCont

Name: F_GetFieldCont – Read the content of a field

Synopsis: F_GetFieldCont Field/A
rc = Success
result = Content

Function: Reads the content of the specified Field in the active record and returns it in result.

May be only called in record mode.

This command may be also called in virtual state.

Inputs: FieldId - Id of Field

Results: rc = 0 - everything Ok, result will be the content
rc = 5 - no record active
rc = 10 - arg missing, or unknown ID.

result - is equal to the current content of the field, if rc = 0.
The format:
String - the string itself.
Integer - the number itself.
Float - the fp number.
Slider - the value of the slider.
Cycle - the number of the active label.
Date - the date in the format DD.MM.[YY]YY.
Time - the time in the format HH:MM:SS.
Extern - the string itself.
Datatyp.- the string itself.

See also:

11.2.20 F_GetProjFullName

Name: F_GetProjFullName – get the name of the current project

Synopsis: F_GetProjFullName
Result = *Name*

Function: Returns the filename of the current project incl. path.

Note: The path is relative to the current directory of Fiasco.

Inputs:

Results: Name - Name of project incl. path.

See also: F_GetProjName

11.2.21 F_GetProjName

Name: F_GetProjName – read the filename of the current project

Synopsis: F_GetProjName
Result = *Filename*

Function: Returns the filename of the current project without path. This value may be used for F_SelectProj (cf. section 11.2.55).

Inputs: none

Results: Result - Filename of the current project without path. A file must not necessarily exist. This is possible, if the name has been changed using Options and the project has not been saved.

See also: F_GetProjFullName

11.2.22 F_GetRecNum

Name: F_GetRecNum – Get the number of the current record.

Synopsis: F_GetRecNum
Result = *Number_of_record*

Function: Returns the number of the active record in result. May be used to save the initial status of the project and to restore it at the end using F_GotoRec (cf. section 11.2.27).

Inputs: none

Results: Result = Number of the record. Note that filters and other operations may change the record numbers.

See also:

11.2.23 F_GotoFirstRec

Name: F_GotoFirstRec – activate the first record

Synopsis: F_GotoFirstRec

Function: activates the first record. If the current project does not contain any records, nothing will happen.

Equivalent with Record/First (cf. section 10.2.22). May be only called in record mode.

Inputs: none

Results: none

See also:

11.2.24 F_GotoNextRec

Name: F_GotoNextRec – activate the next record

Synopsis: F_GotoNextRec

Function: Activates the record after the active one. If the active record is the last record or the current project contains no records, nothing will happen.

Equivalent with Record/Next (cf. section 10.2.21). May be only called in record mode.

Inputs: none

Results: none

See also:

11.2.25 F_GotoLastRec

Name: F_GotoLastRec – activate the last record

Synopsis: F_GotoLastRec

Function: Activates the last record. If the current project does not contain any records, nothing will happen.

Equivalent with Record/Last (cf. section 10.2.23). May be only called in record mode.

Inputs:

Results:

See also:

11.2.26 F_GotoPrevRec

Name: F_GotoPrevRec – activate the previous record.

Synopsis: F_GotoPrevRec

Function: Activates the record, which precedes the active record. If the active record is the first record, nothing will happen.

Equivalent with Record/Previous (cf. section 10.2.20). May be only called in record mode.

Inputs: none

Results: none

See also:

11.2.27 F_GotoRec

Name: F_GotoRec – activate a record.

Synopsis: F_GotoRec Record/A/N

Function: Activate the record, whose number has been given as arg. If the number was invalid, do nothing.

Inputs: RecordNumber - The number of the record. Please note, that sorting, adding or removing records or filters may change the record numbers.

Results:

See also:

11.2.28 F_GotoRecReq

Name: F_GotoRecReq – open the Goto-Requester (cf. section 10.3.11)

Synopsis: F_GotoRecReq

Function: Does exactly the same as Records/Goto (cf. section 10.2.24). May be only called in record mode.

Inputs: none

Results: none

See also: F_GotoRec Record/Goto

11.2.29 F_Import

Name: F_Import – Import data

Synopsis: F_Import File/A,RecStart/K,RecEnd/K,RecSep/K,FieldStart/K, FieldEnd/K,FieldSep/K,SkipLines/k,StartLine/N/K,FirstRecIDs/S, AppendFields/S
rc = *Success*

Function: Calls the import function of Fiasco. The specified file will be imported into the current project using the specified parameters. For more information on import and export see section 8. You may also use the escape sequences of Fiasco. If you do not specify a parameter, it will be empty.

Inputs: File - Name of File
RecStart,RecEnd,RecSep,FieldStart,FieldEnd,FieldSep - the structuring characters
SkipLines - Comment introducer
StartLine - Length of initial comment
FirstRecIDs - First Record contains IDs
AppendFields - Append new fields

Results: rc = 0, if everything went well

Notes: The option Overwrite old project of the import requester is not directly supported. You have to emulate it using F_ClearProject (cf. section 11.2.5).

See also: F_Export, Chapter Import and Export

11.2.30 F_IsMarked

Name: F_IsMarked – Is the record marked?

Synopsis: F_IsMarked Record/N
rc = *IsMarked*

Function: Looks, whether the current or the specified record is marked. If it is not marked, 5 is returned.

This command may be also called in virtual state.

Inputs: Record - Number of record, if not specified, current record is used.

Results: rc = 0: Record marked, = 5: Record not marked, ¿ 5: other error

See also:

11.2.31 F_IsVirgin

Name: F_IsVirgin – Is the project unchanged?

Synopsis: F_IsVirgin
rc = *Is_Virgin*

Function: Tests, whether the current project has been changed since the last saving. If it has been changed, Quit, Erase, Load and so on, will put an requester.

Inputs: none

Results: rc = 0 - Unchanged
rc = 5 - Changed

See also: F_MakeVirgin

11.2.32 F_LoadDTObject

Name: F_LoadDTObject – Load the contents of a datatypes field

Synopsis: F_LoadDTObject Field/A

Function: Loads the contents of a datatypes field, which was “deferred”.

Inputs: Field - ID of datatypes field

Results: The contents are loaded

See also:

11.2.33 F_Locate

Name: F_Locate – Locate the Cursor

Synopsis: F_Locate X/A/N,Y/A/N

Function: Sets the cursor at the given position. At this place the next mask operation will happen. May be only called in mask mode.

Inputs: X - X-Coordinate
Y - Y-Coordinate

Results:

Bugs: Currently not particularly useful, because there are no direct commands for manipulating the mask.

See also:

11.2.34 F_LockGUI

Name: F_LockGUI – Make the GUI not accessible by the user.

Synopsis: F_LockGUI

Function: Locks the GUI of Fiasco. The pointer will appear as a "wait clock". After locking the GUI, the ARexx script can run, without the danger of being influenced by the user. Before the script ends, F_UnlockGUI must be called in order to give the control back to the user. F_LockGUI and F_UnlockGUI may be nested.

Inputs: none

Results: none

Note: Make sure, that your scripts unlock the GUI in every case before exiting. Use signal commands to catch errors or breaks. For example:

```
/* test.rexx */

address FIASCO
options results

signal on syntax
```

```
signal on halt

F_LockGUI        /* Lock the GUI */

/* your code */

F_UnlockGUI      /* Unlock the GUI */

exit             /* And finish */

Syntax:
Halt:

F_UnlockGUI
exit
```

However, if a script leaves Fiasco locked, you may the following script, which is also available in the file ARexx/UnlockGUI.rexx:

```
/*
 * Fiasco will complain once,
 * if ARexx-Debug is activated
 */

address FIASCO

do forever

    F_UnlockGUI

    if rc ~= 0 then break

end
```

See also: F_UnlockGUI

11.2.35 F_MakeVirgin

Name: F_MakeVirgin – Say Fiasco, that the current project is unchanged

Synopsis: F_MakeVirgin

Function: Pretends, that the current project is unchanged. This prevents certain procedures (Erase, Load, Quit,...) to put up a requester.

Inputs: none

Results: A project, that thinks, it has not been changed since the last saving.

Note: The ARexx commands of Fiasco 1.1 provide direct arguments to suppress these warnings. Because of that, this function has no real meaning. It is recommended not to use this command, in order not to confuse the user.

See also: F_IsVirgin

11.2.36 F_MarkAllRecords

Name: F_MarkAllRecords – Mark all Records

Synopsis: F_MarkAllRecords

Function: Marks all records in the current project. They will be displayed highlighted in the list. Does exactly the same as Records/Mark All (cf. section 10.2.27).

Inputs:

Results:

See also: F_UnmarkAllRecords F_MarkRecord

11.2.37 F_MarkMatch

Name: F_MarkMatch – Mark records, which match with a pattern

Synopsis: F_MarkMatch Field/A, Blur/K, Pattern/F/A

Function: Marks all records, which match with the given pattern in the given field. Operates similar to the filter. F_MarkMatch clears the marks of the records, which don‘t match.

Inputs: Field – The ID of the field, which will be examined
Blur – Takes the blurfactor of the comparison. Specify only, if you want to do blurred search.
Pattern – The pattern to search for.

Results:

See also: F_MarkRecord F_ToggleAllMarks

11.2.38 F_MarkRecord

Name: F_MarkRecord – Mark a record

Synopsis: F_MarkRecord Record/N

Function: Marks a record in the current project. It will be displayed highlighted in the list.

This command may be also called in virtual state.

Inputs: Record/N – Optional, if given the record specified by it's number will be marked. Otherwise, the current record will be marked.

Results:

See also: F_UnmarkRecord F_MarkAllRecords

11.2.39 F_NewProject

Name: F_NewProject – Open a new project window

Synopsis: F_NewProject

Function: Opens a new project. A new window is opened and activated. It is then entirely empty. Does exactly the same as Project/New (cf. section 10.2.1).

Inputs: none

Results: none

Bugs: Should claim on error.

See also:

11.2.40 F_OpenList

Name: F_OpenList – Open the list window

Synopsis: F_OpenList

Function: This command is equal to activating the menuitem Control/List (cf. section 10.2.56).

Inputs:

Results:

See also: Control/List F_CloseList

11.2.41 F_OpenProject

Name: F_OpenProject – Load a project

Synopsis: F_OpenProject File/A
rc = *Success*

Function: Tries to read a fiasco project into the current project window. The data, which are currently in the window will be freed *without any request*.

Inputs: Name - Filename of the project

Results: rc = 0, if everything went Ok,
= 10, if argument is missing or file cannot be loaded.

See also: F_OpenProjectReq

11.2.42 F_OpenProjectReq

Name: F_OpenProjectReq – Open the "Open Project" ASL requester

Synopsis: F_OpenProjectReq

Function: Does exactly the same as Project/Open (cf. section 10.2.3). I'm too lazy to write this here again. :-)

Inputs: none

Results: none

Note: The user may have canceled the request

See also: F_OpenProject Project/Open

11.2.43 F_OpenServiceWin

Name: F_OpenServiceWin – Open the service window

Synopsis: F_OpenServiceWin

Function: Opens the service window (cf. section 10.1), if it is not already open.

Inputs: none

Results: none

See also: F_CloseServiceWin

11.2.44 F_OptionsReq

Name: F_OptionsReq – Open the options requester (cf. section 10.3.10) for the current project

Synopsis: F_OptionsReq

Function: Does exactly the same as Project/Options

Inputs: none

Results: none

See also:

11.2.45 F_Progress

Name: F_Progress – give the user a sense of the duration of a operation

Synopsis: F_Progress Done/A/N, Max/A/N

Function: Displays a nice progress bar in the service window, as known of Sort or Open Project. You should reset the status gadget with F_ResetStatus (cf. section 11.2.51) when the operation has completed.

Inputs: Done – the number of data items currently processed.
Max – the number of all data items.

Results:

See also: F_SetStatus

11.2.46 F_Quit

Name: F_Quit – close the current project

Synopsis: F_Quit Force/S

Function: Closes the current project. If you do not specify Force, this command does exactly the same as Project/Quit (cf. section 10.2.12). That means, it is possible, that a requester opens, which asks you whether you want to save the current project before proceeding or cancel.

To prevent this, specify the Force parameter. This will suppress all warnings. To find out, wheter the project is not saved, you may use F_IsVirgin (cf. section 11.2.31).

Inputs: Force – suppress all warnings.

Results: none

Notes: If the current project is closed, another project will be activated, or, if there is no other project, Fiasco will be shut down. An ARexx script should not rely on the order, in which the next project will be activated.

See also:

11.2.47 F_RemAllRecords

Name: F_RemAllRecords – Delete all records of project

Synopsis: F_RemAllRecords Force/S

Function: Removes all records of the current project. If you do not specify the Force parameter, this command does exactly the same as Record/Remove all. That means, that a requester may show up, which will ask you, whether you really want to remove all records. To prevent this behavior, specify Force.

This command may only be called in record mode.

Inputs: Force - suppress all warnings

Results: A project without any records.

See also:

11.2.48 F_RemRecord

Name: F_RemRecord – Delete the active record

Synopsis: F_RemRecord Force/S

Function: Removes the active record and activates the next. If you do not specify the Force parameter, this command does exactly the same as Record/Remove. That means, that a requester may show up, which will ask you, whether you really want to remove this record. To prevent this behavior, specify Force.

This function may only be called in record mode.

Inputs: Force – suppress all warnings.

Results: none

See also: F_AddRecord, Record/Remove Record

11.2.49 F_RequestChoice

Name: F_RequestChoice – request a choice

Synopsis: F_RequestChoice Body/A,Gadgets/A,Title/K
result = *Selection*

Function: Creates an intuition easy-requester with the specified parameters. Works very similar to the CLI command Requestchoice. The differences: Slightly different parameters, puts the requester up on Fiasco's screen.

This command may be also called in virtual state.

Inputs: Body - Main text of requester.
Gadgets - Gadgets at the bottom of requester. Each choice must be separated by a —.
Title - Title of requester.

Results: result - Number of selected gadget, 0 for the rightmost one.

See also:

11.2.50 F_RequestFile

Name: F_RequestFile – request a file

Synopsis: F_RequestFile File,Pattern/K,Title/K,Savemode/S,Drawersonly/S, Noicons/S
rc = *Success*
result = *SelectedFile*

Function: Puts up an ASL file requester. Works very similar to the CLI command Requestfile. The differences: Slightly different parameters, puts the requester up on Fiasco's screen.

This command may be also called in virtual state.

Inputs: File - Initial File including path for the requester
Pattern - Initial Pattern
Title - Title for the requester
Savemode - Activates savemode: Black background, no selection via doubleclick
Drawersonly - Displays only Drawers
Noicons - Filters Icons

Results: rc = 0, if user selected a file, otherwise user canceled.
result = selected file, if rc = 0

See also:

11.2.51 F_ResetStatus

Name: F_ResetStatus – restores the normal informations in the status gadget

Synopsis: F_ResetStatus

Function: Sets the status gadget of the service window to the normal contents. This is RecNum / AllRecs in the Record Mode or X / Y in the mask mode. You should use this call to reset the status informations set with F_SetStatus (cf. section 11.2.60).

Inputs:

Results:

See also:

11.2.52 F_SaveProject

Name: F_SaveProject – Save the current project

Synopsis: F_SaveProject

Function: Save the current project under the old name on disk. Does exactly the same as Project/Save (cf. section 10.2.7).

Inputs: none

Results: none

Bugs: Does not inform the script about errors.

See also: F_SaveProjectReq

11.2.53 F_SaveProjectReq

Name: F_SaveProjectReq – Open filereq and save project under new name

Synopsis: F_SaveProjectReq

Function: Does exactly the same as Project/Save As... (cf. section 10.2.8).

Inputs: none

Results: none

Note: The user may have canceled the request

See also: F_SaveProject

11.2.54 F_SaveSettings

Name: F_SaveSettings – Save the current program settings.

Synopsis: F_SaveSettings

Function: Does exactly the same as Settings/Save Settings (cf. section 10.2.69).

Inputs: none

Results: none

See also: Settings/Save Settings

11.2.55 F_SelectProj

Name: F_SelectProj – activate an already load project

Synopsis: F_SelectProj Name/A
rc = *Success*

Function: Activates a project, which stays already in memory. The filename without path is used to identify the project. This may be the name, which has been obtained using F_GetProjName (cf. section 11.2.21). All following commands refer to the new project.

Inputs: Name - Name of project without path.

Results: rc = 5, if project is already active.
= 10, if argument is missing, or there is no such project.

See also: F_GetProjName

11.2.56 F_SetFieldCont

Name: F_SetFieldCont – Change the content of a field.

Synopsis: F_SetFieldCont Field/A,Cont/A/F
rc = *Success*

Function: Sets the content of the specified field in the active record to the specified content.

May be only called in record mode. This command may be also called in virtual state.

Inputs: Field - Identificationname of the field
Cont - New content of the Field. This arg takes the whole input inclusive spaces. The Interpretation of this arg depends on the fieldtype:
String - is copied directly
Integer - Numbers are read directly, other things are 0
Float - dto.
Boolean - 1 or TRUE = selected, 0 or FALSE = not selected
Slider - Number is read. Bad numbers will be adjusted.
Cycle - Number or name of label is taken.
Date - Date in Format DD.MM.[YY]YY is taken.
Time - Time in Format HH:MM:SS is taken.
Extern - is copied directly
Datat. - is copied directly

Results: rc = 0 - no error
rc = 5 - no record is active
rc = 10 - missing arg or bad FieldId

See also:

11.2.57 F_SetMode

Name: F_SetMode – Select the editing mode

Synopsis: F_SetMode Mask/S, Records/S
rc = *Success*

Function: Activates the specified mode for the current project.

Inputs: Mask - activates mask mode.
Records - activates record mode.
Mask and Records are mutually exclusive.

Results: rc = 0 - no error
= 5 - the project was already in the specified mode
= 10 - missing or bad arg

See also:

11.2.58 F_SetSearchField

Name: F_SetSearchField – Set the field to search

Synopsis: F_SetSearchField Field/A
rc = *Success*

Function: Sets the field, which will be searched by F_FindFirst and F_FindNext.

Inputs: Field - The Id of the field, which shall be searched.

Results: returns 10 in rc, if the argument is missing or the id is unknown.
otherwise 0 is returned.

Notes: You don‘t need this function for simple searching, because Fiasco 1.1 allows passing these parameters directly with the search functions.

See also: F_SetSearchPat

11.2.59 F_SetSearchPat

Name: F_SetSearchPat – Set the pattern to search for

Synopsis: F_SetSearchPat Pattern/A/F
rc = *Success*

Function: Sets the searchpattern for the active project. If you also have specified a search field using F_SetSearchField, you may search for matching entries using F_FindFirst and F_FindNext. The value will be also used in the search requester.

Inputs: searchpattern - a string to search for

Results: rc = 0 - everything ok.
rc = 10 - no argument
rc = 20 - systemfailure (no memory, etc.)

Notes: You don‘t need this function for simple searching, because Fiasco 1.1 allows passing these parameters directly with the search functions.

See also:

11.2.60 F_SetStatus

Name: F_SetStatus – display a status string to the user

Synopsis: F_SetStatus String/A

Function: Displays the given string in the status gadget of the service window (cf. section 10.1). If the service window is not open, nothing will be done.

Inputs: String – the string to be displayed

Results:

See also: F_ResetStatus

11.2.61 F_Sort

Name: F_Sort – Sort the records of the current project

Synopsis: F_Sort Field/A,Descending/S

Function: Sorts the records of the current project according to the alphabetical or equivalent priority of the contents of a specified field.

Inputs: Field – The ID of a field to sort after
Descending – Specify this, if you want the sorting to be backwards.

Results:

See also: F_SortReq, Compare/Sort

11.2.62 F_SortReq

Name: F_SortReq – Open the sort requester

Synopsis: F_SortReq

Function: Does exactly the same as Compare/Sort (cf. section 10.2.47). May be only called in record mode.

Inputs:

Results:

See also: Compare/Sort...

11.2.63 F_ToggleAllMarks

Name: F_ToggleAllMarks – toggle the marks of all records.

Synopsis: F_ToggleAllMarks

Function: Clears the marks on records, which were marked and sets the marks on previously unmarked records. Does exactly the same as Records/Toggle All Marks (cf. section 10.2.29).

Inputs:

Results:

See also: F_MarkRecord

11.2.64 F_UnlockGUI

Name: F_UnlockGUI – Unlock the GUI of Fiasco

Synopsis: F_UnlockGUI
rc = *Success*

Function: Unlock the GUI, which has been previously locked using F_LockGUI (cf. section 11.2.34). The user has again access to Fiasco. F_LockGUI and F_UnlockGUI may be nested.

Inputs: none

Results: rc not equal 0, if no lock was present.

See also: F_LockGUI

11.2.65 F_UnmarkAllRecords

Name: F_UnmarkAllRecords – Clear the mark on all records.

Synopsis: F_UnmarkRecord

Function: Clears all marks of the records in the current project. They will be rendered in a normal appearance in the list. Does exactly the same as Records/Unmark All (cf. section 10.2.28).

Inputs:

Results:

See also: F_UnmarkRecord F_MarkAllRecords

11.2.66 F_UnmarkRecord

Name: F_UnmarkRecord – Clear the mark on a record

Synopsis: F_UnmarkRecord Record/N

Function: Clears the mark on a record, which has be set previously by F_MarkRecord (cf. section 11.2.38) or using the GUI. The record will be rendered in a normal appearance in the list.

This command may be also called in virtual state.

Inputs: Record/N – Optional number of record to unmark. If not given, the current record will be unmarked.

Results:

See also: F_UnmarkAllRecords F_MarkRecord

11.2.67 F_UserCommand

Name: F_UserCommand – Calls a userdefined command.

Synopsis: F_UserCommand Command/N/A

Function: Calls a command, which has been defined in the "User" menu.

Inputs: Command - Number of command, counted from zero. If this number does not exits, nothing will be done.

Results: none

Note: This command is *only* for implementing a icon bar or similar things. It should not be used in normal scripts, because the command may change freely.

See also:

11.2.68 F_VirtualMode

Name: F_VirtualMode – Is this script called from virtual mode?

Synopsis: F_VirtualMode rc = *Virtual*

Function: Tests, whether the running script is called by Fiasco in virtual mode or in normal mode.

Inputs:

Results: rc = 0: virtual status
rc ¡¿ 0: normal status

See also: Section virtual fields

Chapter 12

Example Projects

The directory databases of the Fiasco distribution contains several Fiasco projects. Some of them may be also used for own purposes.

12.1 Addresses

The Address project can be used as a simple addressbook. It contains fields for Name, Address, Phone, etc. The project uses relations to translate the abbreviations of country names (like "I" for Italy) to the long names.

The fields for Phone, Fax or Zipcode are string fields, because they also have to take characters like "/" or must have a leading "0" (which would be swallowed by a integer field).

An additional Idea would be to use relations to search for the name of the city using the zip code.

12.2 Datatypes Demo

This project is a easy Demonstration of the Datatypes fieldtype, which requires the datatypes.library. For this reason, it is only available for users of Amiga OS 3.0 or higher. The mask contains three fields, which can be used to display all Data, which have the correct datatypes installed.

Two fields have scrollbars at the bottom and at the right side. You can use these scrollers to move the contents of the field. The stringgadget below the display contains the name of the file. The gadget with the arrow down at the left side of the string gadgets can be used to open a filerequester for editing the filename.

One field has a button marked with an 'S'. This button opens a filerequester, which allows you to select a file, in which the currently displayed data are saved in. Fiasco writes the data in IFF format.

The field at the upper right has the 'immediate play' attribute, which plays the data — if playable — directly after loading the data.

The browsing between records may get a bit slower, because the data are stored in an external file and must be loaded first.

12.3 FamilyTree

The family tree consists of the projects "persons.fdb" and "families.fdb". "persons.fdb" contains all persons, which are used in the family tree. You may also enter sex, date of birth, etc. here.

These data are used by "families.fdb" with relations, to get names of spouses, children, etc. Additionally, there are fields for marriage and divorce. Caused by the intensive use of relations, this project only contains 10 "real" fields, which are stored on disk. The other 12 fields are loaded from "persons".

12.4 Videos

The video database can be used to manage your homevideo collection. The database consists of two projects: "movies.fdb" and "tapes.fdb". "Movies" takes the informations for each movie (Genre, Director, etc.). The field "Tape" connects each film with one tape, which can be found in "tapes". Here is the play length of each tape defined. An ARexx script calculates the left free space on the tapes.

12.5 Picture Database

This databases uses the Datatypes fieldtype and is only usable, if you have Amiga OS 3.0 or better. The Datatypes field has the "defer" attribute, which means, that Fiasco won't load the data immediately. To read the data, you have to click in the string gadget of the field and press Return.

The string field under the datatypes field takes a description of the picture.

The button "Scan directory" can be used to read a freely selectable directory in the database. "Show on screen" displays the picture on an own screen. This button currently only works, if you use absolute paths for the graphics.

12.6 FAQs Database

This database manages textfiles. For displaying the files, it uses the program Most by Uwe Röhm. If you prefer another textdisplay program, which has an ARexx port, you may adopt the ARexx scripts.

The string field at the top of the mask takes the name of the textfile. Under it there are three buttons to control the database. "Scan dir" reads a selectable directory into this database. "View" displays the currently active text file. The most complex button is "Search". It can be used to search through the all files in the database for a string. If you click on it, a window opens, which asks you for an string to search for. Then it asks for a record to start the search. If you simply hit enter, it will begin at the first record. Then you are asked, whether you want to search all records or only the marked ones. After that Fiasco asks you, whether you want to write the results to a file. Then the last option comes, "Interactive searching". If you activate this, you will be asked for every found string, if you want to display the file at this place. After that the searching starts. If you want to break the searching, simply hit Ctrl-C.

The database contains the data for the FAQ (Frequently Asked Question) files on the Meeting Pearls II CD-ROM. If you don't own the CD ROM, use Record/Delete all Records to get rid of the data.

Appendices

Appendix A

All Searchpatterns

`#?`	String, Extern, Datatypes	An unknown string with undefined length
`?`	String, Extern, Datatypes	An unknown character
`> x`	Integer, Slider	A number, which is greater than x
`< x`	Integer, Slider	A number, which is less than x
`>= x`	Integer, Slider	A number, which is greater or equal x
`<= x`	Integer, Slider	A number, which is less or equal x
`!= x`	Integer, Slider	A number, which is not equal x

Detailed descriptions are available with the field documentations.

Appendix B

Relation Checklist

- create key field “there”. Optionally activate “unique key”.
- create real field “there”. In case of string, extern or datatypes, remember “max chars”.
- save project.
- create key field “here”. Must be the same type as “there”.
- create real field “here”. Must be the same type as “there”. In the case of string, extern or datatypes, “max chars” must be equal.
- save project.
- open relation requester for real field “here”.
- select key “here”
- select relation file
- select key and real field “there”. If the correct field is not displayed, check type and in case of string, extern or datatypes max chars.
- select Ok

Appendix C

Implementation of the Clipboard support

The menuitems Cut Record, Copy Record and Paste Record use the clipboard to store data temporarily. The clipboard of the Amiga OS is meant to provide a interface for different programs to share certain types of data. To make this possible, the clipboard may only contain IFF data.

Fiasco uses unit 0 of the clipboard and stores its data in IFF-FTXT files with a specific format. Each field gets a separate chunk. In this chunk the field content is stored in ASCII format.

The order of the chunks depends on the internal field list of Fiasco. Fiasco also uses this order to find out, which data belongs to which field while pasting the clipboard-contents.

With most other programs, you cannot create such structured IFF-FTXT files. The pasting in other programs is better supported. For example the conclip- program pastes the data correctly, while MultiView displays only the first chunk.

Appendix D

Bugs

If you find some bugs in Fiasco, send a detailed description to me. Please include information about your processor, OS version and other configuration.

These bugs are currently known:

- The frame of the list window flashes sometimes in a weird way under Kickstart 37.x
- ARexx seems to have problems with filenames, which contain spaces. The name is only interpreted to the the first space. This affects the full path, because Fiasco expands the name to the full path before calling ARexx scripts.
- Seems to leave sometimes some memory allocated.

Appendix E

To do

Fiasco is of course not perfect, at all. Here is a list of all things, which will be perhaps added at a later point (no guarantee!). If you have an Idea, send it to me!

- Better scrolling in the mask window. I currently use GadTools gadgets, which have to be recreated if you want to change their positions. I plan to emulate the used gadgets. This would not require any more GZZ windows.
- New searchfunction.
- Sorting should be take several fields into account and should get faster.
- New ARexx commands: ReadRecord and WriteRecord. Should read all fieldcontents and put them in ARexx variables with the field id as names.
- New searchpatterns.
- Hiding of fields in the mask.
- Print function; Reports
- Stringfields, which support multiple lines
- Reversed logic while searching and counting
- AppWindows for Datatypes and Extern Fields

- "Packing" of projects: search for unused fields and make used as small as possible.
- Checking, whether a similar record already exists (automatically)
- Ability to specify the order, how the fields are activated after a Return.
- Save status of list window.
- Fiasco should not have to read the whole file. (To save memory)
- Iconify projects
- Extend ARexx commands, that they can refer to other records than the current.
- Separator bars for Mask
- ServiceWindow hides the unused gadgets instead of disabling them
- List fieldtype with "Add" and "Del"
- "ARexx hook" for extending relations
- Better support of mask mode in ARexx

Index