Overview

SIR/XS is a comprehensive database management and application development system. The normal way to use SIR/XS is through a set of graphical user menus and dialogs. Simply enter the operating system command *sir* or click on the appropriate icon and the menus are displayed. This gives access to any SIR databases and enables you to create and develop applications. In addition to this, there are a number of features for more complex situations.

The menus allow you to perform all standard tasks but you may wish to provide scripts for other users to use and want to know about commands.

When you execute *sir*, the command line has various optional keywords and parameters known as execution parameters that specify a starting database and various other settings.

The system can also run login scripts that further customise the environment you are using.

If you have a number of regular, routine tasks that are not done by people interactively, but are simply run, then you may wish to use sirbatch facilities.

If you have a number of users who need to update the same database simultaneously, then you may wish to use sirmaster process that controls and co-ordinates updates.

If you want to provide ODBC access to your SIR data for other applications, then you need to install the SIR ODBC driver and use the SirSQL Server.

If you wish to allow one SIR/XS session to send commands to run on another part of the network, then you may wish to use PQLServer facilities.

Introduction

When you start SIR/XS, you see the main SIR window. This consists of a title bar, a main menu and toolbar at the top, a large scrollable output area in the middle and a message and progress area at the bottom.

The title bar displays the name of the application, the default database (if any) and, if concurrent database access is being used, the name of the controlling master process.

The menu (and subsequent sub-menus and dialogs) is the main way of interacting with SIR/XS. (Note. These menus and dialogs are all written in VisualPQL and the source of all programs is included on the system procedure file and can be altered or replaced by custom built menus.)

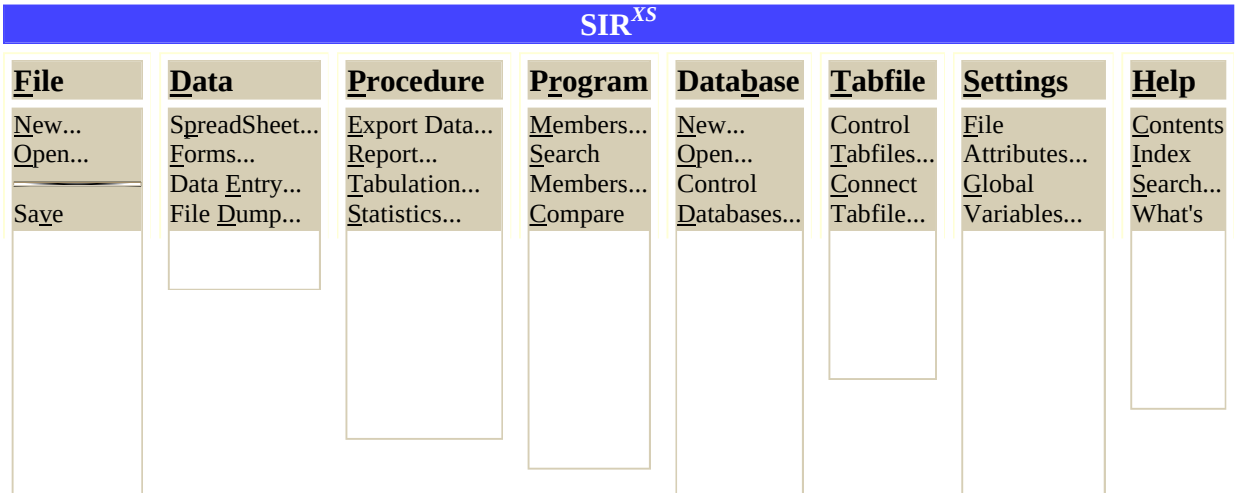
Select a choice from any menu with the arrow keys, with a mouse or other pointing device or with underlined "hot keys" (use Alt-letter in the standard way) or by any indicated control key. Some of the frequently used items can also be activated from the toolbar.

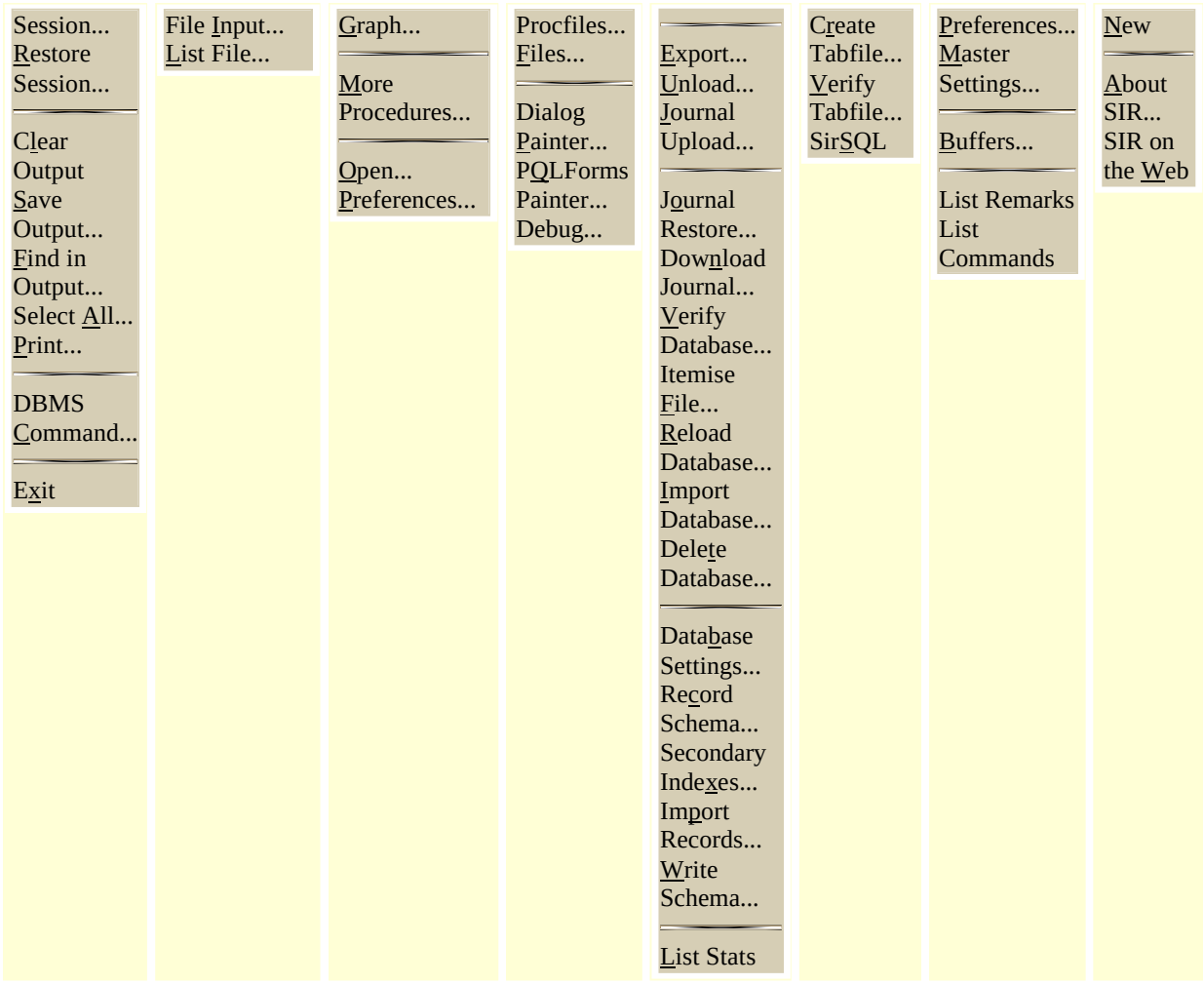
Return to a previous menu by pressing `Esc`.

Menu items may pull down further menus (indicated with a small arrow), may display a dialog that you fill in or may take an immediate action. Some actions may generate output in the Main Output Window. This is a scrollable area (up/down/left/right) that holds remarks, messages and screen listings. The amount of remarks, commands and other internally generated messages can be controlled through session options. You can select and cut from the main output window and can print it, save it or clear it. If your session produces more output than it can hold, earliest lines are discarded.

When you start SIR/XS, you may get a 'Welcome to SIR/XS' dialog asking what you want to do. This allows you to create a new database or to connect to an existing database. Press Close to use SIR/XS without a database. Check "Don't show me this again" to suppress this. It can be set to display again in the preferences dialog.

Default Main Menu





The **File** menu contains commands and utilities for working with the output window and overall session.
The **Data** menu contains commands and utilities for working with the data in a database.
Procedures are simple dialog based utilities for extracting data from a database to various output.
The **Program** menu contains commands and utilities for working with VisualPQL programs and DBMS command procedures.
The **Database** menu contains commands and utilities for managing databases.
The **Tabfile** menu contains commands and utilities for managing tabfiles.
Settings control the current session settings and options.
Help accesses documentation both locally and on the internet.

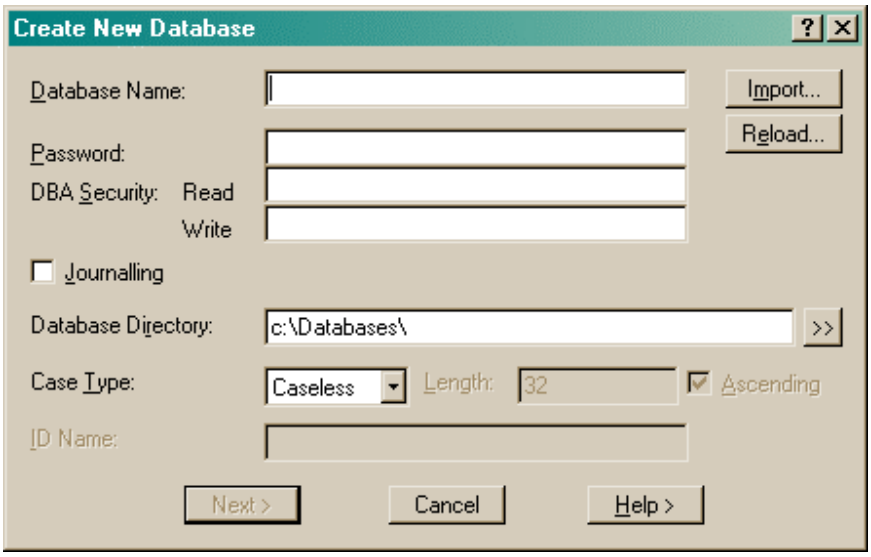
Drag and Drop files

You can select a file and 'drag and drop' it into the SIR/XS main window. The action taken is based on the file extension:
.sr1, .sr2 or .sr3 connect to the database. If password or security are defined then a dialog

is displayed;
.sr4 or .srp set the default procfile to this file;
.pwr invokes the PREAD dialog;
.tbf connects the tabfile;
.unl invokes the Reload dialog;
.exp invokes the Import dialog;
.dlg invokes the Dialog Commander;
Any other file extension invokes the program/files dialog.

New

Use this command to create a new SIR database.
Shortcut: **CTRL+N**



New Database

The New Database dialog requires a database name. You can specify a **Password** and Database Administrator (**DBA**) read and write passwords if you wish (all must be valid SIR names). If a password is assigned, it must be supplied every time anyone connects to this database. If DBA passwords are assigned, then these must be specified when connecting to the database in order to be allowed to perform DBA only tasks such as modifying the database definition.

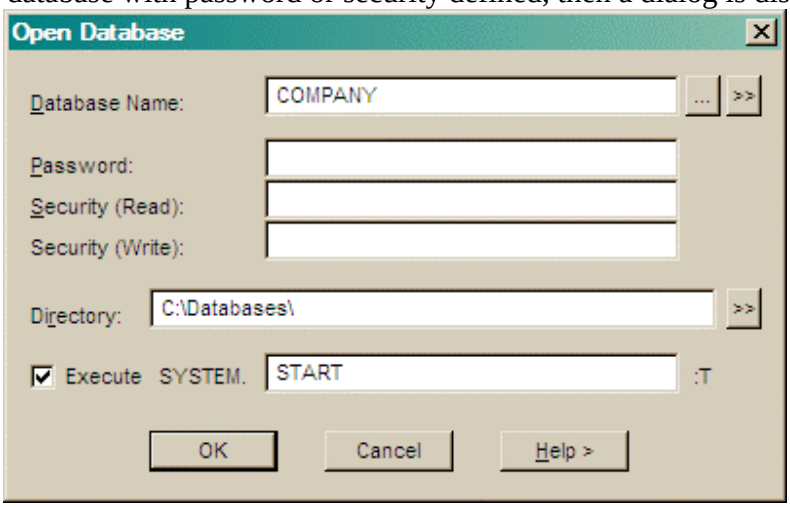
The database directory specifies where the database files are created (defaults to your current directory). Ensure any specification includes a terminating delimiter e.g. a backslash '\' on Windows. You must have write access to this directory. The database filenames are the database name with extensions .sr*n* where n is 1 to 6. Check Journaling to keep a record of changes made to the database and data that can be used to recreate a database if necessary.

A database can have a case structure where each record on the database belongs to an entity and is usually processed as part of this entity. For example, a personnel database might contain various record types (basic data, positions held, reviews, etc.) where all records relate to a single employee with an employee identification that can be used as the case id. If the database is Case structured, select the data type from 1, 2 or 4 byte integers or string. One byte holds values up to 123, two bytes up to 32763 and four bytes over 2 billion. Assign the case id a valid SIR name. Cases are normally held in ascending sequence. Uncheck Ascending to store data in descending sequence.

See also the DBMS `CREATE DATABASE` command.

Open

Use this command to open an existing SIR database or other file.
Shortcut: **CTRL+O**
If the file is a database with password or security defined, then a dialog is displayed:



Open Database

Enter the name of the database or use the browse button [>>] to find the database.

If a database password exists you must enter it to attach to the database (Passwords are not case sensitive).

Enter a **Read Security** and **Write Security** password. If you do not enter any passwords and there are no security passwords defined, then you have full access to the database. If there are passwords defined, then the access you have is restricted to the level matching the passwords you specify.

If you browsed for the database, the directory is filled in automatically. When specifying a directory, ensure any specification includes a terminating delimiter e.g. a backslash '\' on Windows.

See also the DBMS `CONNECT DATABASE` command.

If other file types are selected and further information is needed to open that file then an appropriate dialog is displayed.

Save and Restore Session

Use these commands to save and restore the current SIR session's settings and database connections.
The settings are saved in a text file as a set of DBMS commands that recreate the session environment. If any databases require connecting and have passwords, when you restore the session, it prompts for access and security level passwords.

Clear Output

Clear the output window or selected text.

Save Output As

Shortcut: **CTRL+S**
Saves the output window contents or selected text to a chosen text file.

Find

Shortcut: **CTRL+F**
Search the output window for specified text.

Select All

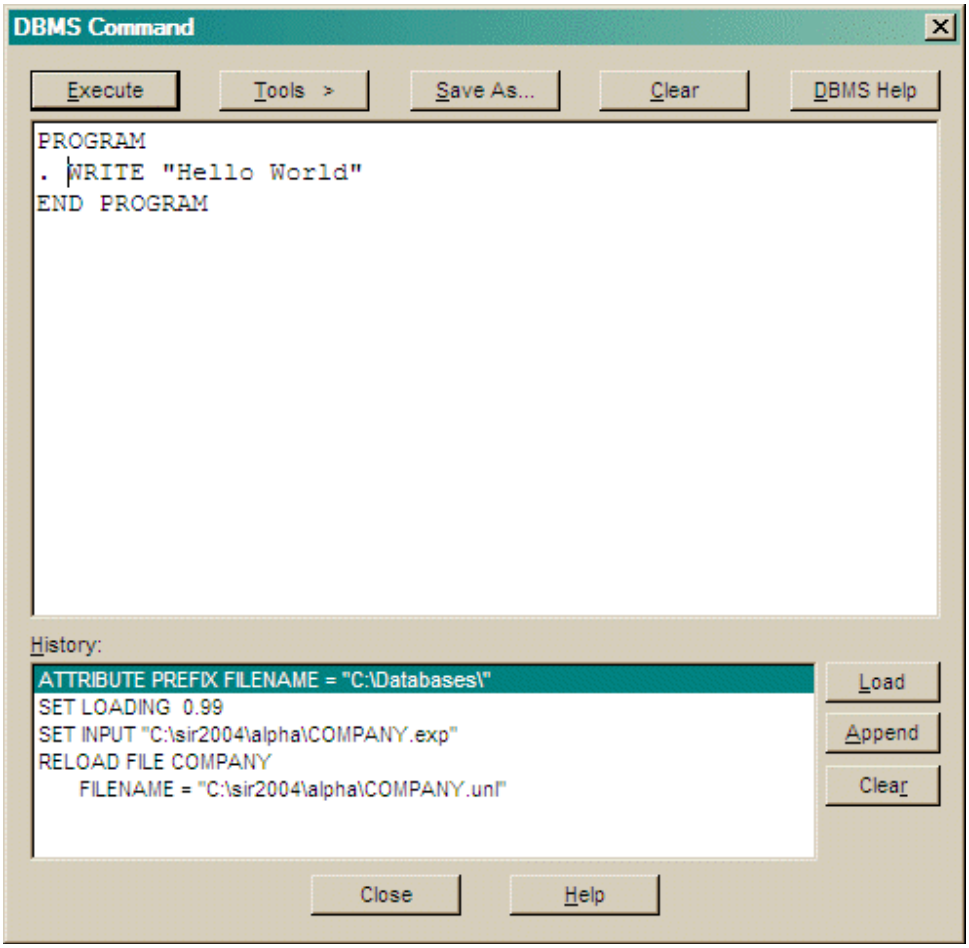
Shortcut: **CTRL+A**
Selects all text in the output window.

Print Output

Shortcut: **CTRL+P**
Prints the output window contents or selected text.

DBMS Command

Shortcut: **CTRL+R**



Run DBMS Commands

Allows entry and execution of DBMS commands and Programs.

Tip

If this command is chosen when the cursor is at the end of a line of text in the output window then that line is taken as a command and executed without opening the DBMS command dialog.

Tip

The default main menus program allows you to enter the name of a menu command in the output window. For example, if you type `MEMBERS`*[ctrl+R] then the Members dialog is displayed.*

The DBMS Command dialog allows entry of DBMS commands in the top text area. Once a command is entered it can be run using the Execute button or cleared using the corresponding Clear button.

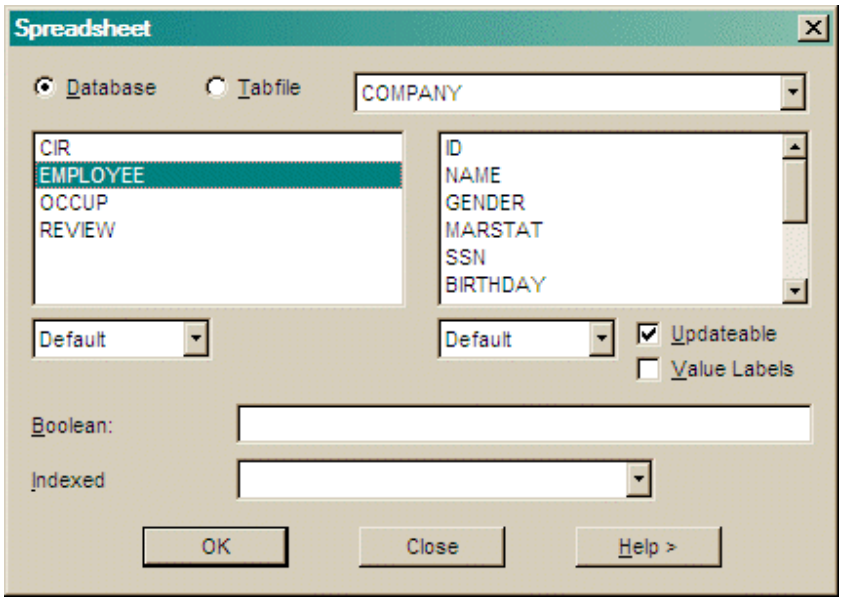
A LIST of previous commands is at the bottom. This list also includes some commands generated by the other menu commands. These commands can be selected and appended to the current command or cleared.

You can change the size of this dialog in preferences.

Exit

Terminates the SIR session.

Spreadsheet



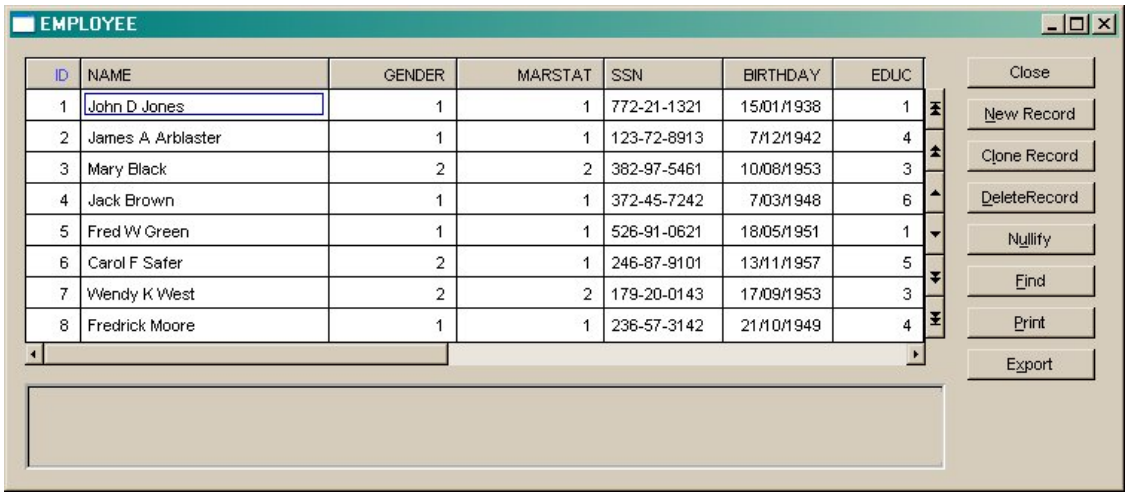
Spreadsheet table selection

The menu interface to the SIR/XS spreadsheet provides a simple and convenient way to examine and modify data in a database record or tabfile table.

The dialog lists all record types in the selected database or tabfile and allows you to select one. By default all variables are displayed or you can select individual variables. By default the spreadsheet is enabled to allow updates to the records providing that all key fields are selected. Uncheck update to disable this. By default, the values of variables are displayed. If you want value labels displayed, check the value labels box; this disables updating.

Enter a Boolean condition to select a subset of records. You may apply conditions based on variables in the record or in the Case information record (CIR).

The SIR/XS spreadsheet can also be invoked by the `SIR SPREADSHEET` command and can be invoked as the `SPREAD SHEET` VisualPQL Procedure.



Spreadsheet

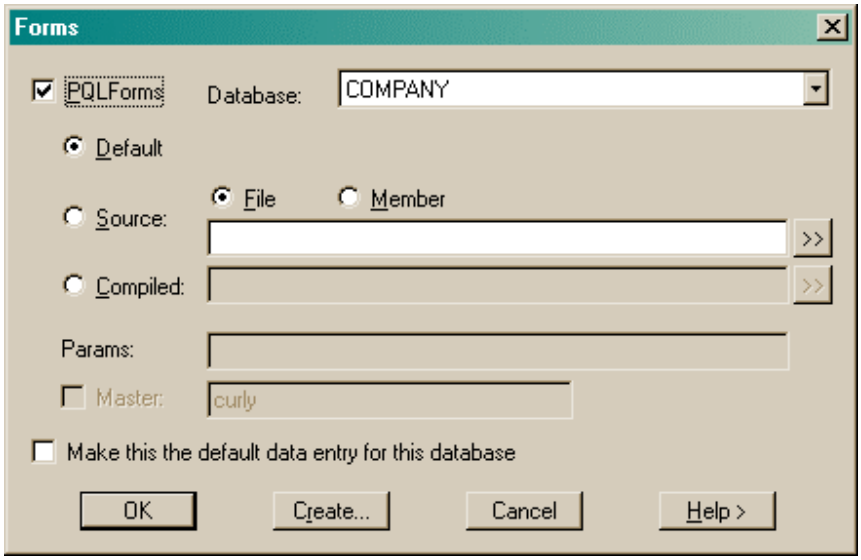
Once the spreadsheet is active, you can move around the displayed data using keys in a similar fashion to other spreadsheet packages. You can edit data by double-clicking or pressing **Enter**. You cannot modify key fields (shown as bold) in existing records.

There are eight action buttons available as follows:

- **Close** Closes the spreadsheet and returns to the main menu.
- **Find** Allows you to search the spreadsheet for specified strings. (Note the find/replace function only updates local data; it does not flag records for database update.)
- **New Record** Creates a single empty row after prompting for keys.
- **Clone Record** Copies the current row to create a new row.
- **Delete Record** Deletes the current row.
- **Export** Creates a copy of the spreadsheet in CSV (comma separated values) format.
- **Print** Prints the spreadsheet.
- **Nullify** Sets the value of the current cell to *UNDEFINED*.

Forms & Data Entry

Forms



Run SIRForms or PQLForms

Lets you create and run data entry forms. SIRForms is a character screen based data entry system. PQLForms is a data entry system based on VisualPQL and has a graphic interface.

Check the **PQLForms** box to create or run PQLforms.
Choose **Default** and press ok to create and execute a form definition for the current database.
Choose **Source File**, and enter the name of a **Text Form** definition file. Use the "[>>]" button to find a file using the operating system file browser.
If you are using SIRForms then you can also enter the name of a file to store a precompiled version of this form.
Press **Create...** to create a default source file with select record types.
Choose **Precompiled** and enter the name of a **Compiled SIRForm** definition file. Use the "[>>]" button to find a file using the operating system file browser.

If a non default form is chosen then **execution parameters** such as `DB=dbname PREFIX=path` etc may also be entered (See SirForms).

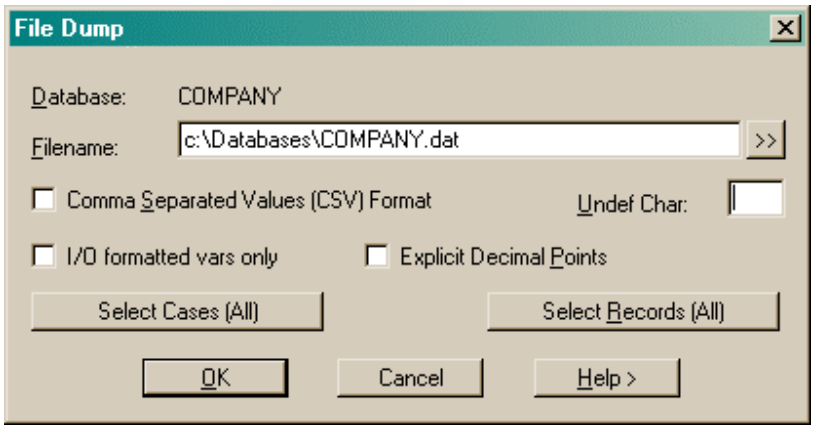
Check **Master** and enter the name[:port] of a master process to start the form in concurrent update mode.

Check **Make this the default data entry for this database** to save this as the method used by the **Data Entry...** menu item.

Data Entry

This option will run a default form for this database. The default will either be the generic PQLForms data entry system created by `WRITE SCHEMA` or it will call the default entry procedure set in the forms dialog above.
The default procedure is stored in the member RUN_DEF_FORM in the database's SYSTEM family.

SIR File Dump



SIR File Dump

The File Dump command writes a text file containing record data from selected record types and cases.

The output file is suitable for input to the File Input utilities. You must have DBA level access to use this utility.

Enter a **filename** to contain the dumped data. Use the "[>>]" button to find a file using the file browser.

Check the **CSV** option to dump the data in *Comma Separated Values* format.

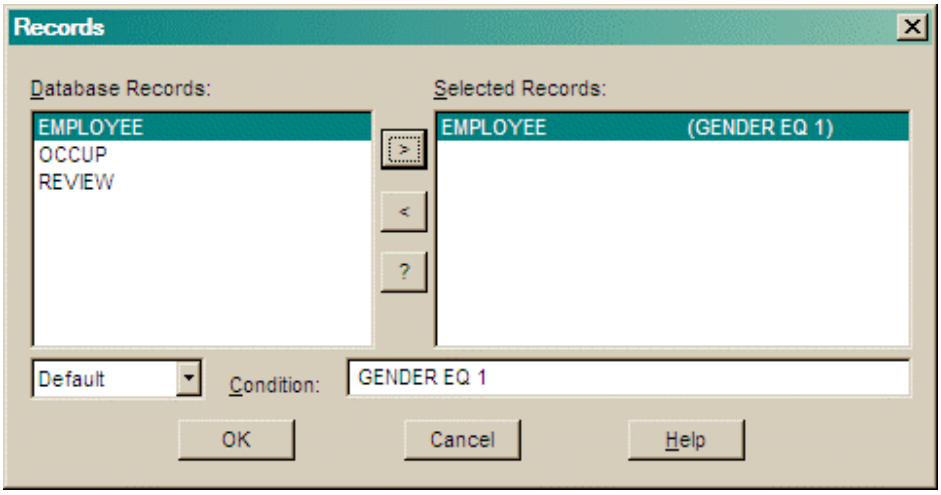
Enter an optional **Undefined Character** to be written to the file when undefined data are found. The default undefined character is BLANK.

If the **I/O formatted variables** option is selected then the format is as defined in the data dictionary. If the **I/O formatted variables** option is not selected then any variables without file formatting defined in the schema are written to the end of each line.

Check the **Explicit Decimal Points** box and real variables with decimal parts will be written with their decimal points. The default is that the decimal point character is not written but implied by the default number of decimal places defined in the schema for that variable. (eg an F4 with a value of 3.1415936 will be dumped as 31416 and input as 3.1416).

Press **Select Cases** to define a subset of case data for the output.
Press **Select Records** to define a subset of record data for the output.
See also the DBMS `FILE DUMP` command.

Select Records



Select Record Types

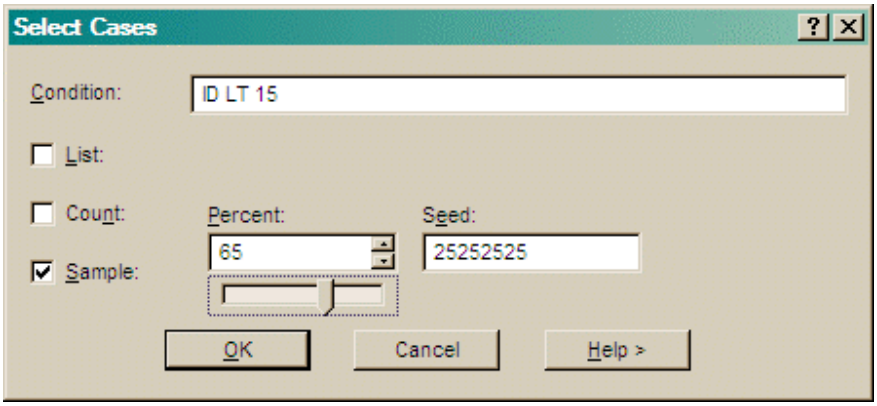
The Select Records dialog is used by several utilities. It is used to restrict the types of records processed by the utility and also filter individual records based on a Boolean expression.

Enter an optional **Boolean** condition based on variables in a record and then select the record type from the list. You do not need to enclose the condition in brackets.

Use the pull down choice to change the order that the record types are displayed in the dialog's record menu.

Press > to add your selection to the **Selected Records** list.
Press < to remove an item from the **Selected Records** list.
Press **OK** to save the selection.

Select Cases



Select Cases

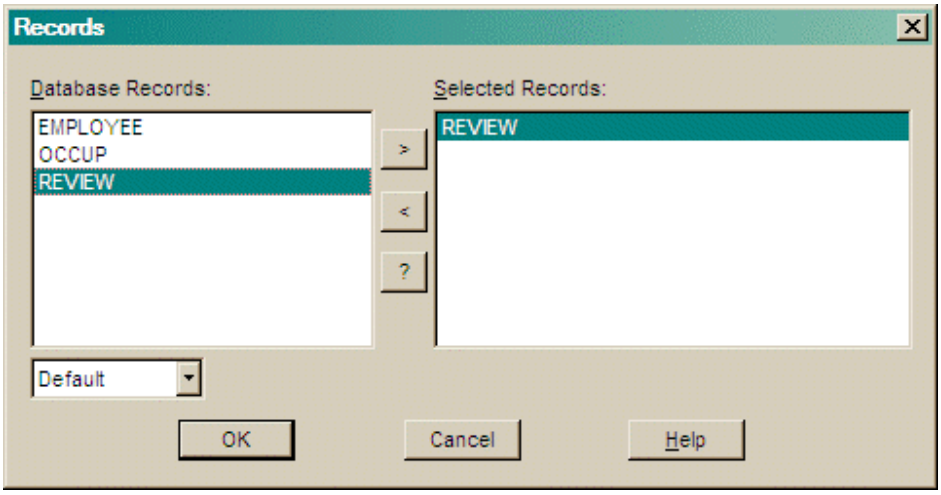
The Select Cases dialog is used by several utilities. It is used to restrict the number of cases processed by the utility and only applies to case structured databases.

Enter an optional **Boolean** condition based on variables in the CIR. You do not need to enclose the condition in brackets.

- Check **List** and type a list of the case ids to be used (e.g. `1,2,4,8 thru 10,15`).
- Check **Count** and type a number of cases, optional increment and optional starting case ordinal (e.g. `5,3,2`).
- Check **Sample** and specify a percentage between 0 and 100 and an optional random number seed (e.g. `.5,18543545`)

Press **OK** to check the syntax of the Boolean and list then save the settings.

Select Records



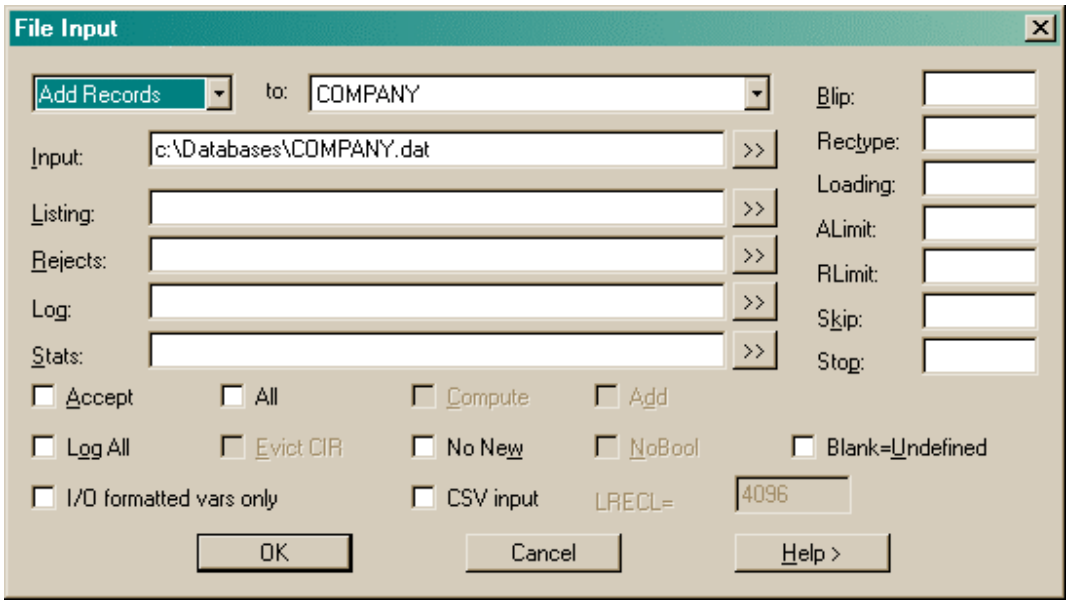
Select Record Types

The Select Records dialog is used by several utilities. It is used to restrict the *types* of records processed by the utility.

Use the pull down choice to change the order that the record types are displayed in the dialog's record menu.

Press **>** to add your selection to the **Selected Records** list.
Press **<** to remove an item from the **Selected Records** list.
Press **OK** to save the settings.

File Input



File Input

Updates the data in the database from a text input file. This file can be created using a SIR File Dump. The format can be defined in the data dictionary or can be a *Comma Separated Values* format (CSV) file.

Choose the type of data updating required from the list:

- **Add records** adds new records if they don't currently exist.
- **Replace records** replaces existing records.
- **Read input data** adds or replaces records.
- **Update records** replaces selected variables on existing records.
- **Evict records** deletes existing records.

Choose the database to update.

Input file

Enter the name of the file containing the input data. Use the "[>>]" button to find a file using the operating system file browser.

Output files

Optionally enter the names of files to contain information on rejected input records.

- The **Listing** file contains error messages for rejected records.
- The **Records** file contains the rejected record in its original format.

Summary files

Optionally enter the names of files to contain information on the input run.

- The **Log** file describes rejected records.
- The **Stats** file contains a summary of the numbers of records added, modified or deleted.

Options

Not all options are applicable to all update types. Options that are not applicable are disabled.

- Check **Accept** to accept records with erroneous data values.
- Check **All** to write records with errors to the error file whether or not they are Accepted.
- Check **Logall** to write records with errors to the log file whether or not they are Accepted.
- Check **NoNew** to specify that no new Cases are created.
- Check **NoSeq** to ignore sequence checking on multiple line input records.
- Check **EvictCir** to delete the Common Information record if all records in this case are deleted.
- Check **Compute** to specify that any data dictionary COMPUTEs are to be reexecuted.
- Check **Add** to allow new records to be created.
- Check **NoBool** to specify that any data dictionary ACCEPT REC IFs are bypassed.
- Check **Blank=Undefined** to specify that numeric variables will be set to *Undefined* when the input file contains blanks. The default is for these variables to be set to zero if BLANK is not defined as a valid missing value.
- Check **I/O Formatted Vars** to only update variables with text file format settings in the schema.
- Check **CSV** to update from a *Comma Separated Values* format input file. If you choose this option then you may need to specify a file record length. It does not matter if this length is longer than the physical record length.

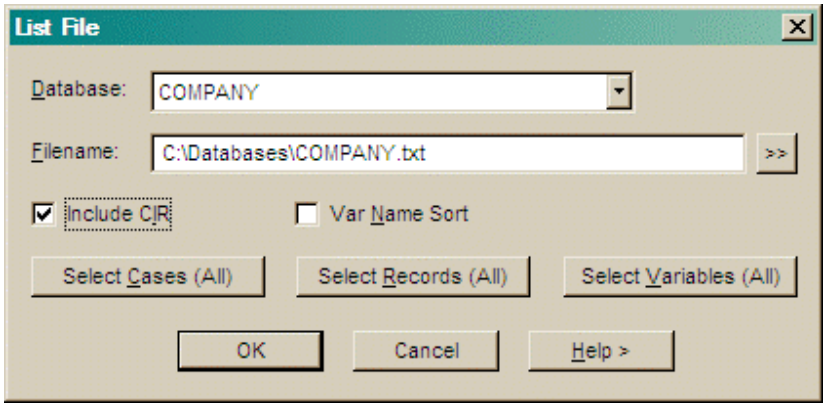
Parameters

- **Blip**. Enter an approximate number of input records to enable the progress bar.
- **Rectype**. Enter the record name or number that is to be updated in this run. If this is not specified it is got from the record type columns on the input file.
- **Loading**. Enter a number between 0 and 99 (the default is 50).
- **ALimit**. Enter a number to limit the number of records accepted with errors.

- **RLimit**. Enter a number to stop the input run when that number of records is rejected.
- **Skip**. Enter a number of lines to skip at the start of the input file.
- **Stop**. Enter a number to stop the input run when that number of records is read.

See `batch data input`.

List File



List File

Produces a report showing the data in the selected database.

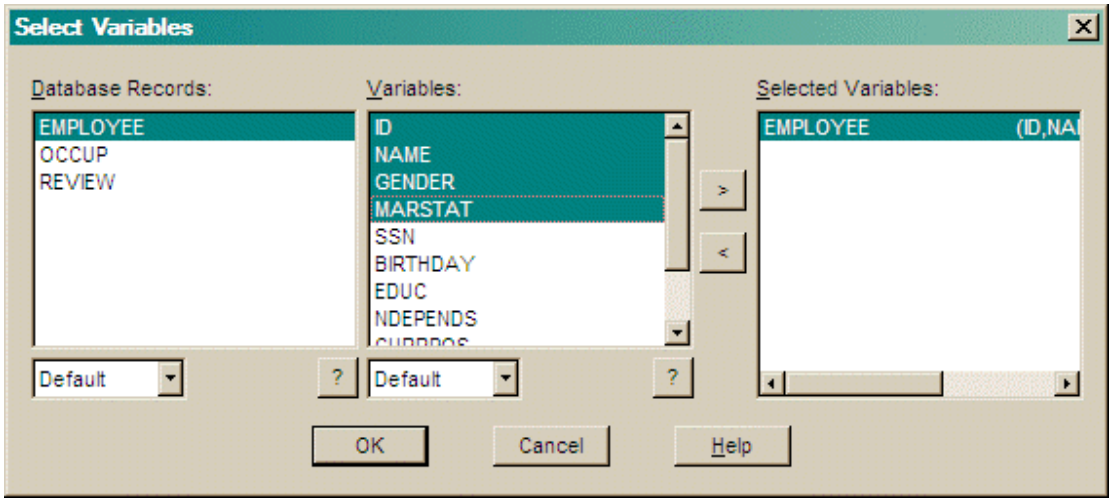
You must have DBA level access to use this utility.

You can select record types and variables to display and also take a subset of the data.

Check **Include CIR** to display data in the Common Information Record.
Check **Sort By Variable Name** to list the data for each record in alphabetical order of variable name. The default is the data dictionary order.
Press **Select Cases** to define a subset of case data for the output.
Press **Select Records** to define a subset of record data for the output.
Press **Select Variables** to define variables to include in the output

See the `FILE LIST` command.

Select Variables



Select Variables

Select Record types and variables to be processed by the utility. If no variables are selected then all variables in the selected record are processed.

Use the pull down choices to change the order that the record types and variable names are displayed in the dialog's menus.

Press > to add your selection to the **Selected Records** list.
Press < to remove an item from the **Selected Records** list.

Output Procedures

The items under the Procedures menu offer an easy way to generate output from your database. These are related to the PQL Procedures in that they each generate a PQL program that includes one of those PQL Procedures.

The Output can be a report or tabulation, a graph or a file readable by another software package.

The procedures can be saved as either a set of selected options for the dialogs or as a PQL program. You can edit the saved PQL to perform more complex operations than can be specified in the dialog interface.

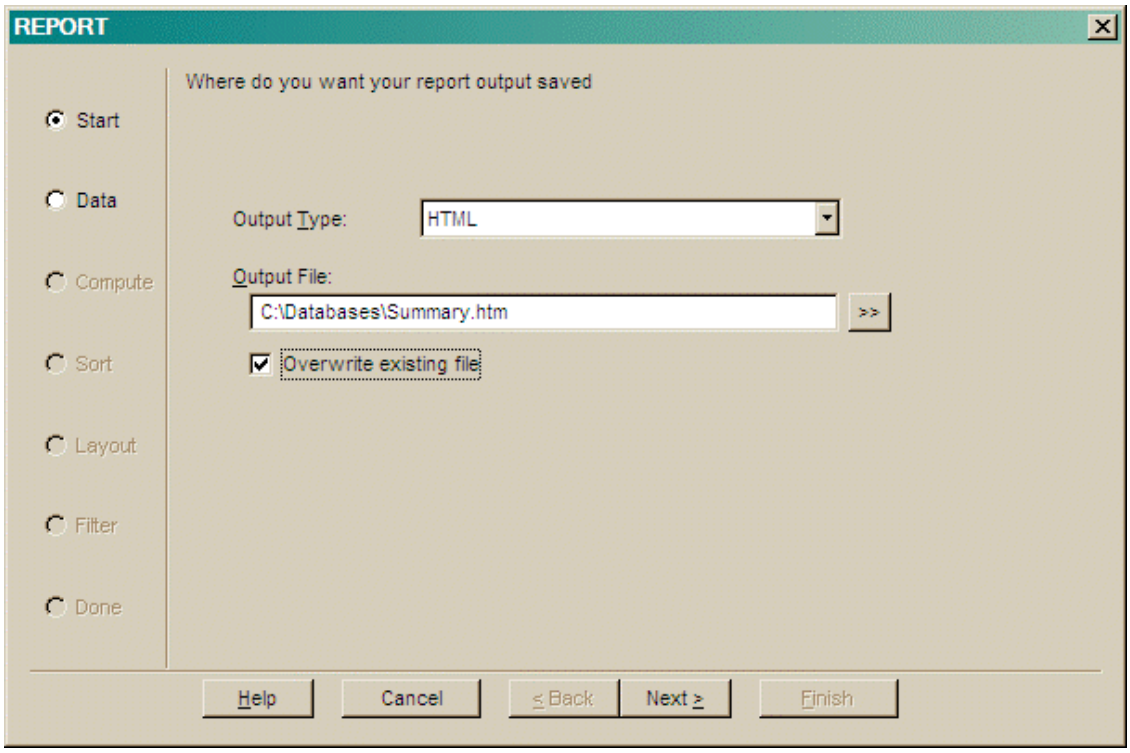
The More Procedures selection in this menu is a place where you can store commonly run reports and other programs. These programs appear by name in a list and you can add new programs and descriptions. The most recently executed program from the procedures above can be stored in the More Procedures list, as can external programs in files or members.

The Preferences selection will open the main preferences dialog on the Procedures page.

The Export, Report, Tabulate, Graph and Statistics options in this menu all initiate a set of dialogs that allow you to select an output file, database and local variables, sort, filters and layout options for the generated output. These dialogs are described below.

1. Output File Type and name

Enter an output filename and select an output type.



Select Output Type

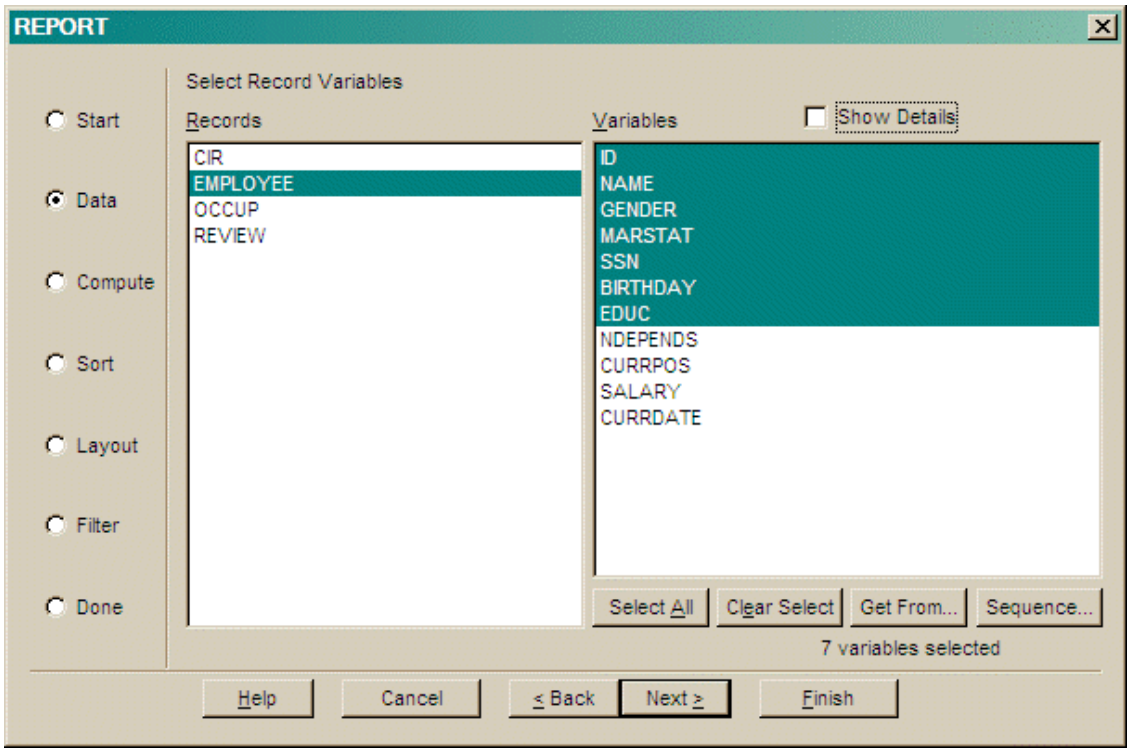
The available output types for Export are:

- **Comma Delimited File** - readable by many packages (text)
- **BMDP® Format** - BMDP Statistical package (binary)
- **DBASE® Format** - readable by DBASE, Excel (binary)
- **Data Interchange Format (DIF)** - readable by many packages (text)
- **Minitab® Portable Format** - Minitab Statistical Software (text)
- **SAS® Export Format** - SAS Statistical Software (text)
- **SPSS® system Format** - SPSS Statistical Software (binary)
- **SPSS® portable Format** - SPSS Statistical Software (text)
- **SYSTAT® Format** - SYSTAT Statistical Software (binary)
- **XML® Format** - eXtensible Markup Language (text)

The first page of the dialog lets you select a more specific type of output. In the case of Export, the type of file format (e.g. SPSS, SAS, Minitab...). For reports or tabulations, you can select HTML or Text output. Enter or browse for the name of the output file.

2. Select Variables

Select record variables from the default database for use in the procedure.



Select Variables

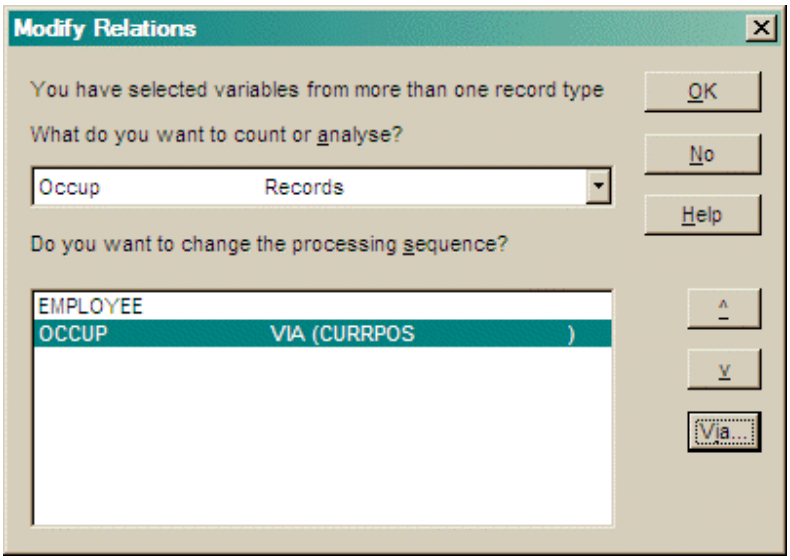
If you are going to create derived variables (eg AGE GROUP - based on BIRTHDAY). If variables are chosen from more than one record type and these record types have different keys then you may want to specify the processing **sequence** to extract the data from these records.

- **Records** Lists the available record types in the database. Select a record type and its variables are shown.
- **Variables** Lists the available variables in the selected record type. Select the variables required.
- Press **Select All** to select all variables in the **current record**.
- Press **Clear Select** to unselect all variables in **this and any other record**
- Press **Get From...** to get the variable selection from a previously saved procedure. ***Note:*** *this also restores derived variables and record filtering options.*

Check the Show Details box to include the variables' type and label in the selection list.

Note: If you have selected variables from more than one record type and there is not a one to one relationship between those records then you may need to specify a path from one record type to the other. A dialog is displayed to specify this path.

Sequence



Modify Relations

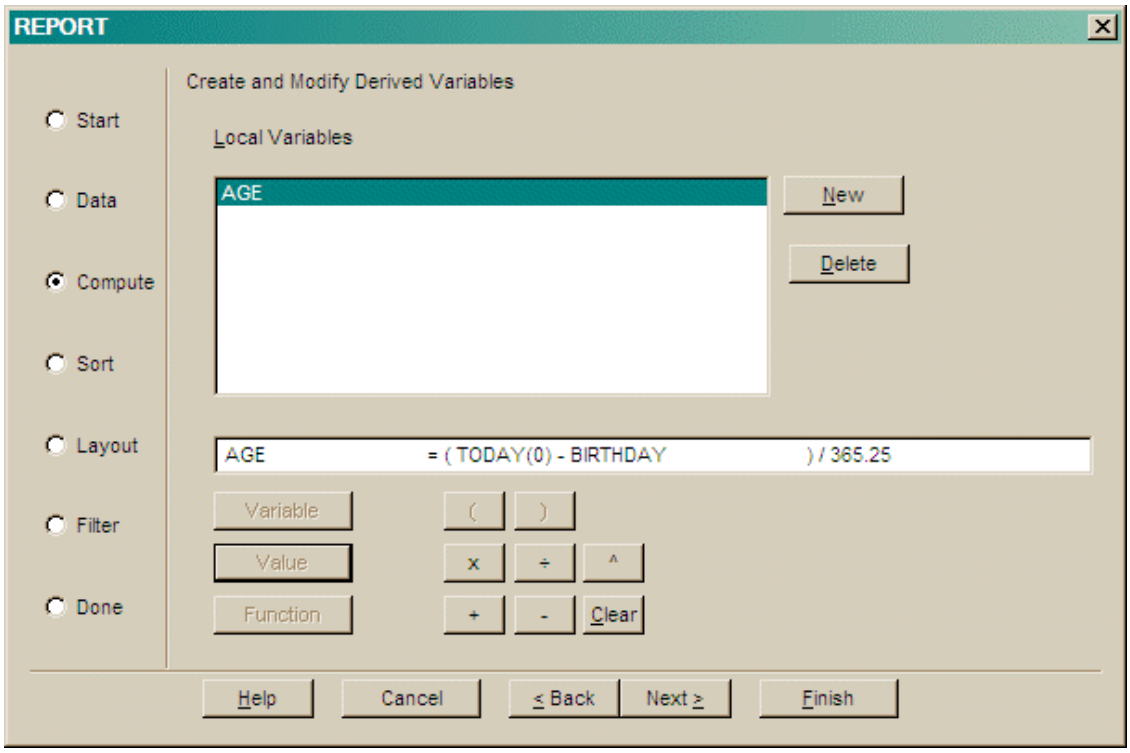
Select the type of record that you want to analyse. For example if you want a report on People then you might choose a demographic record.

Change the record processing sequence and paths using the up/down arrows and the via... button.

In the 'Via' dialog, choose the variables from previous records to use as the key fields for this record.

3. Local Variables

Local variables are not stored in the database but are created in the procedure. They can be based on variables in the schema, functions and constants (such as age from birthday). You must select any database variables used in deriving variables from the previous screen.



Create Derived Variables

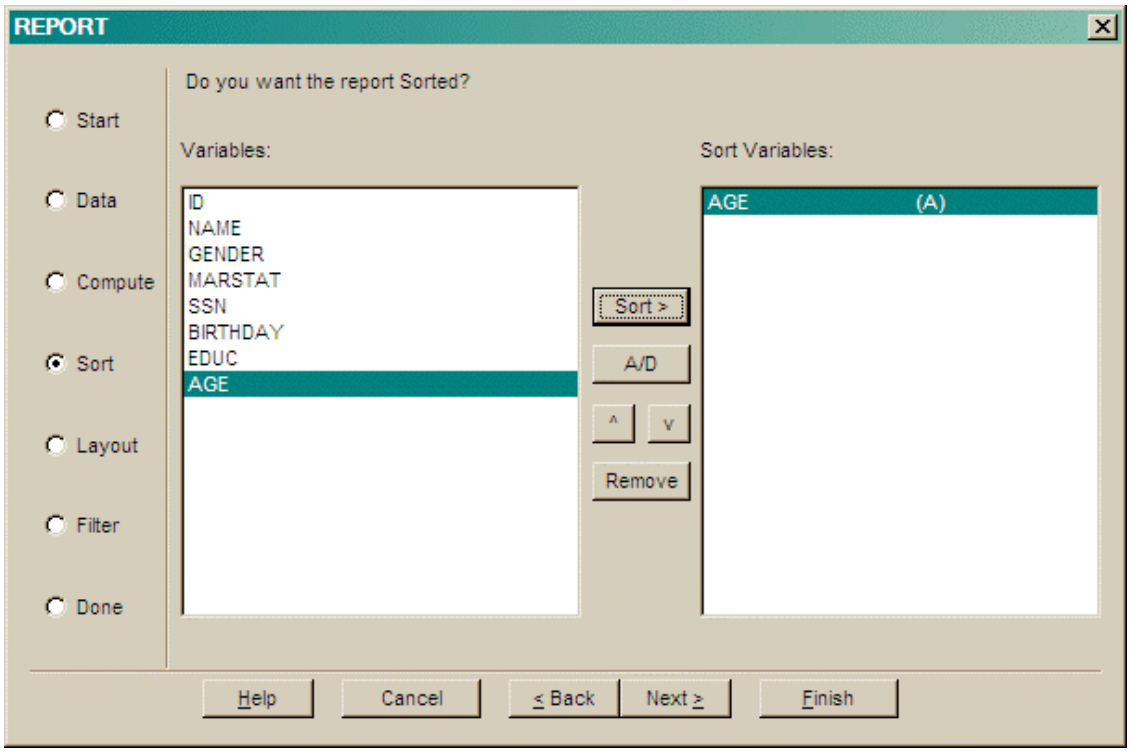
- Press **New** to create a new variable, then select the type of variable and give it a unique name.
- Press **Delete** to remove a local variable.

Use the other buttons to create an expression to define the local variable.

- Press **Variable** to put a variable name into the expression. Only variables of the appropriate type are listed.
- Press **Value** to put a constant value into the expression.
- Press **Function** to use a VisualPQL function in the expression. Only functions of the appropriate type are listed.

4. Sort Output

Select variables to sort the procedure output.



Sort Output

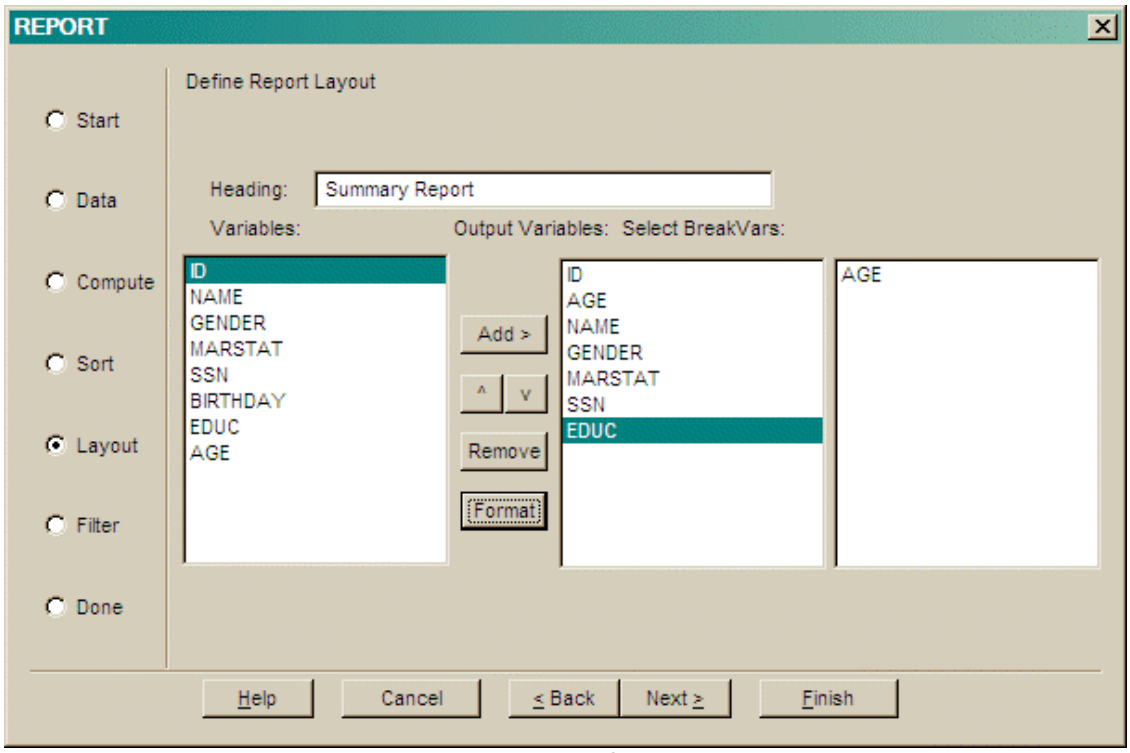
A list of available variables is displayed.

- Press **Sort >:** to append the selected variable to the sort variable list.
- Press **A/D** to toggle between Ascending and Descending sort.
- Press ^ or **v** to move the selected sort variable up or down in the list of variables.
- Press **Remove** to remove the selected sort variable.

If you want to break a report into sections then you must sort the data first. In reports and export files, the output can be sorted. Variables used by sort must be selected or derived variables.

Press **Sort >** to add the selected column to the list of Sort Variables.
Press **A/D** to toggle the sort order of the selected variable (**A**scending or **D**escending).
Press ^ to move the selected variable **up** the list.
Press **v** to move the selected variable **down** the list.
Press **Del** to remove the selected variable from the sort variables.

5. Extra Options



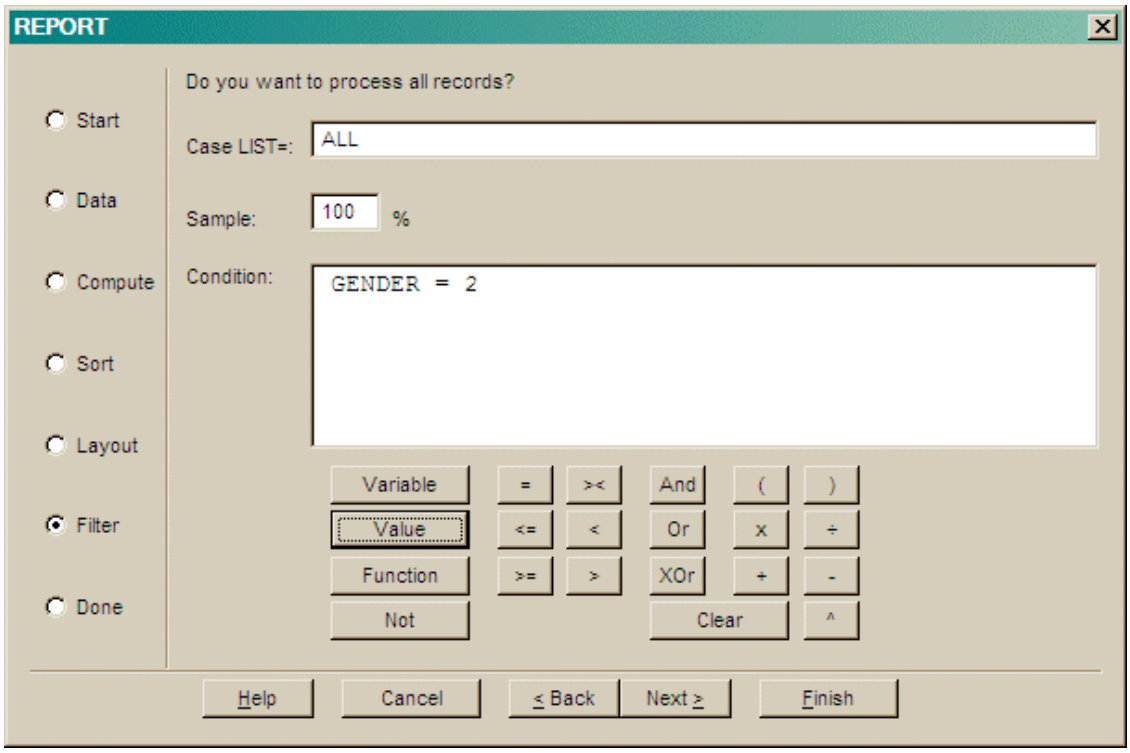
Output type specific options

The following options are available on Report:

- **Heading** Specifies the text displayed at the top of each page of the report.
- **Variables** Lists all the variables available to the procedure.
- **Output Variables** Lists the variables, in report order, that are used in the report. If there are no variables in the list then ALL the variables available are used. Variables can be added to this list from the list of available variables, removed or resequenced.
- **Break On** Lists all Sort variables. Select one or more to break the report into sections.
- **Total** displays totals for all numeric variables at the end of the report.
- **Subtotals** displays totals for all numeric variables at the end of each section, if break variables are selected.

6. Filter Records

Use filter to specify a subset of the data to be used in the procedure.



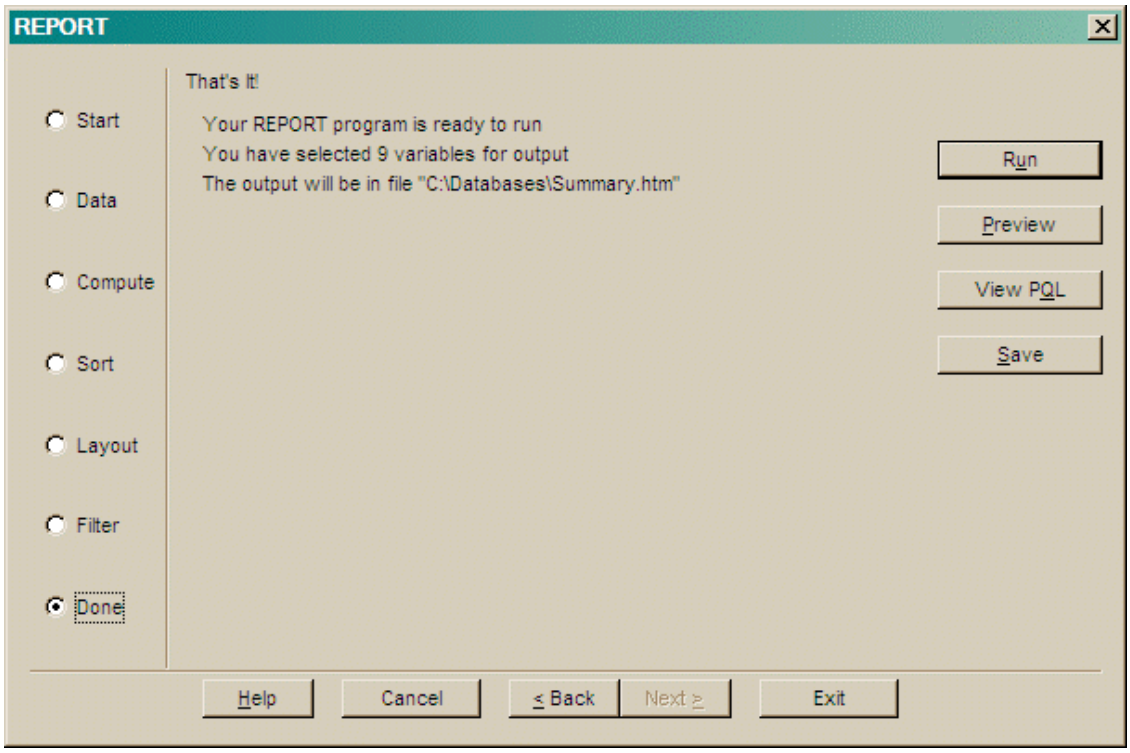
Filter Output

Specify a percentage (0 - 100%) for a proportional **Random Sample** of the data.

Specify a logical **Condition** that records must satisfy to be used.
Use the buttons to create a logical expression.

- Press **Variable** to put a variable name into the logical expression.
- Press **Value** to put a constant value into the logical expression.
- Press **Function** to use a VisualPQL function in the logical expression.

7. Finished



Run & Save Program

- **Run** Runs the procedure and displays the output.
- **Preview** Runs the procedure on a small number of records and displays the output.
- **View VisualPQL** Shows the VisualPQL code that is generated. The options in the procedures represent a fraction of the options available in the VisualPQL language. You can save this code and use it as the basis of a more complex procedure.
- **Save** Saves the options and settings specified in this procedure. Saved files can be restored or run like any other set of SIR/XS commands in a file. The variable selection section can be loaded without loading all the procedure options so that a report, say, can be produced on the same set of data used in another procedure type.

More Procedures

This displays a dialog listing previously stored programs that will run on this database. These programs are stored in the `PROCEDURE` family belonging to the current database. There are also some *SYSTEM* procedures that will run on any database.

A user can add a *USER* procedure by pressing the **Add** button then entering information about the new procedure and selecting the procedure's program source. If you have run one of the other procedures in the main menu (eg Report... Tabulate... Export...) in this session then you can also save its source as a stored procedure.

The program source along with the entered information is then copied to the procedure family.

If you uncheck the **Copy Source** button then the stored procedure will simply CALL the external source code. This way you can put the procedure into several databases but only maintain one copy of the program.

Pressing the **Run** button will execute the selected procedure. What happens after you press run depends on the stored program.

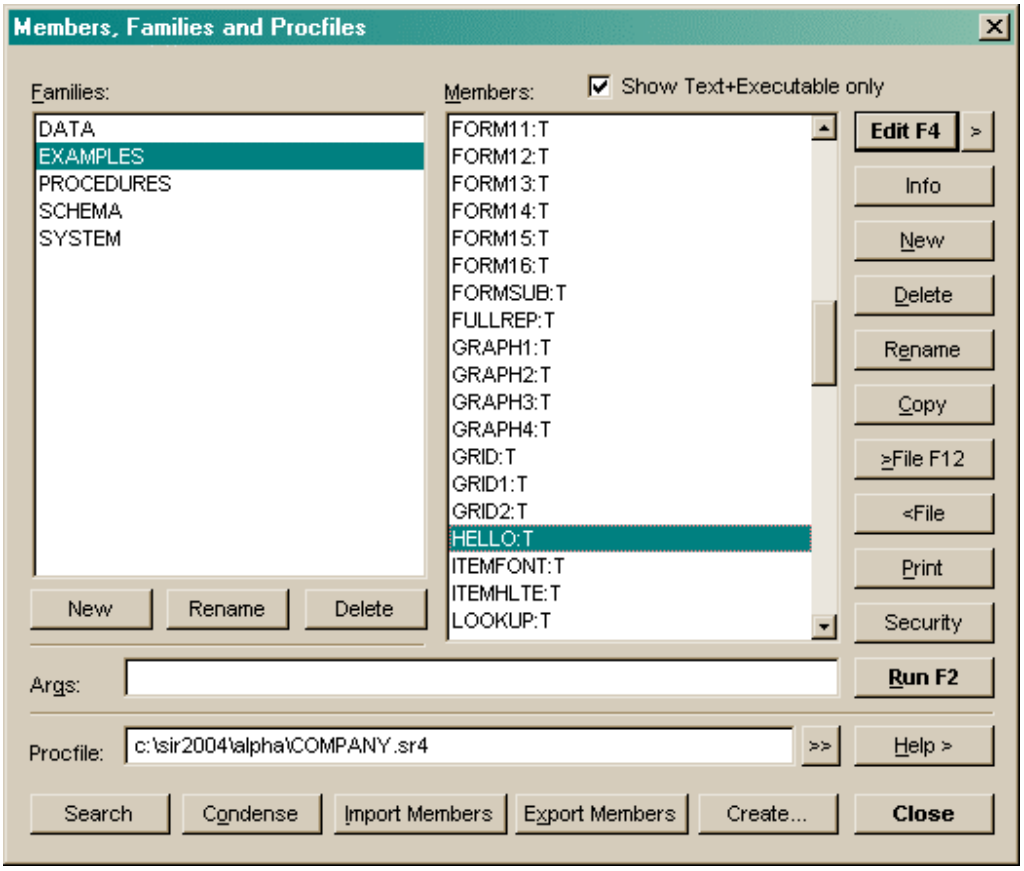
Save and Reload

The selections and options used in a procedure can be saved and restored. There is a Save button in the final dialog that prompts for a filename. Saved files can be restored or run like any other set of SIR/XS commands in a file.

The variable selection section can be loaded without loading all the procedure options so that a report, say, can be produced on the same set of data used in another procedure type.

Program Members

Shortcut: **CTRL+SHIFT+M**



Procedure File Members

Program Members are stored in the Procedure File. Members may be VisualPQL programs, sets of commands or other components of an application.

Families

A family is a directory or grouping of members. Families can only be created, renamed or deleted if you have DBA permissions.

The dialog displays a list of the families in the current procedure file and allows you to select a family. The most recently referenced family is set when the dialog is displayed. There are several family operations:

- Press **New** to create a new family.
- Press **Delete** to delete the current family and all its members.

- Press **Rename** to change the name of the current family.

Members

There are different types of members that are identified by a suffix delimited from the name with a colon (**:**). You can edit and view text members (:T). You can run text (:T) and executable members (:E). The dialog displays a list of the members in the current family. There are several member operations:

- Check the **Show Text+Executable only** box to hide system members (subroutines - :O and external variable blocks - :V) from the display.
- Press **Edit** (or double click on the member) to edit the current text member.
- Press **Info** to see the creation and modification dates and times, the size, type, security and first few lines of the current member.
- Press **New** to create a new member.
- Press **Delete** to delete the current member.
- Press **Rename** to change the name of the current member.
- Press **Copy** to copy the current member to a new member.
- Press **>File** to copy the current member to a file.
- Press **<File** to replace the selected member with a file.
- Press **Print** to print the current member, the current family or the entire file.
- Press **Security** to set Family and Member passwords.
- Press **Run** to execute the current member. If this is a text member, it is expected to contain SIR/XS commands. Optionally specify a list of parameters using **Args:**. These values, delimited by commas, are substituted into any positional parameter variables in the member.

There are several other options:

- Press **Search** to search the current member, the current family or the entire file for a specified text string. All members searched are listed in the output window together with any lines containing a matching string.
- Press **Condense** to condense the file. As members are saved and re-saved in the procedure file, the old members take up disk space. Condense the file to physically delete the copies of old members and recover this space.
- Press **Import Members** to read a file produced by `PWRITE` and store the members from that file. This gives options of notification and for permissions to overwrite existing members of the same name.
- Press **Export Members** to write the current member, the current family or all text members on the procedure file to a file. You require DBA security to write the whole procedure file or selected families.
- Press **Create** to create a new procedure file.

Most members relate to a specific database and are saved in the procedure file that belongs to that database. Normally the procedure file you are working with is the

procedure file of the default database. However, you can select another procedure files to work with.

The System Procedure File

The system procedure file is shipped with the software and contains various programs and procedures to run the system. The system procedure file has the internal name `SYSPROC` and the external name `SirProc.srp`.

Passwords

Procedure files, families and members may have passwords. If a database password is specified when the procedure file is created, the password must be supplied to access that procedure file with DBA permissions. If a password is specified when a family or member is created, the password must be supplied to access that family or member. A password on a saved member prevents those who do not have the password from using it.

Users may need to execute or run procedures stored as members, but not be allowed to see or modify those procedures. Give a member a password and then make it *public* to do this. Public members may be executed but not seen or modified unless proper passwords are provided. Make members public with the `Public` option on the dialog.

Member Types

There are different types of members that are identified by a suffix delimited from the name with a colon (**:**).

There are four current types of members. The member type is identified by a suffix delimited from the name with a colon (**:**). The current member types are:

`Text` **`:T`**
: These contain readable text such as the source code of VisualPQL programs and other command procedures and can be edited. The other member types are compiled and so cannot be edited. The source code and the compiled version are frequently stored as members with the same name with a different suffix. If a member name without a suffix is `RUN`, and both a :T or :E version exist, the system determines which to run based on dates and times of last update.

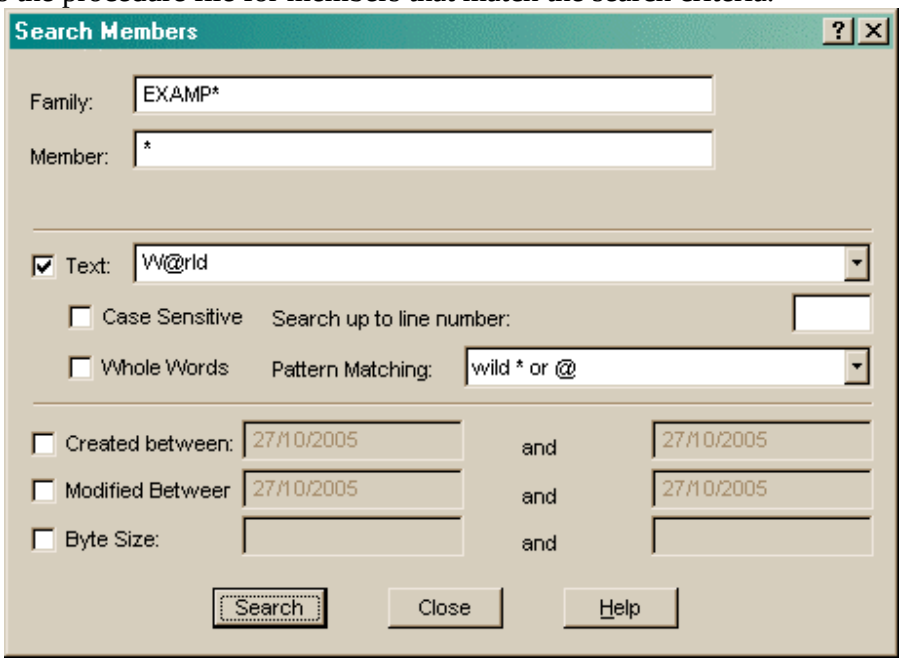
`Executables` **`:E`**
: These are compiled versions of VisualPQL programs and retrievals. Executable members are created by the VisualPQL compiler.

`Object Code` **`:O`**
: These are compiled versions of VisualPQL subroutines. Compiled subroutines are created by the VisualPQL compiler.

`External Variables:`**`V`**
: These are compiled versions of VisualPQL external variable blocks. Compiled external variable blocks are created by the VisualPQL compiler.

Search Members

Searches the procedure file for members that match the search criteria.



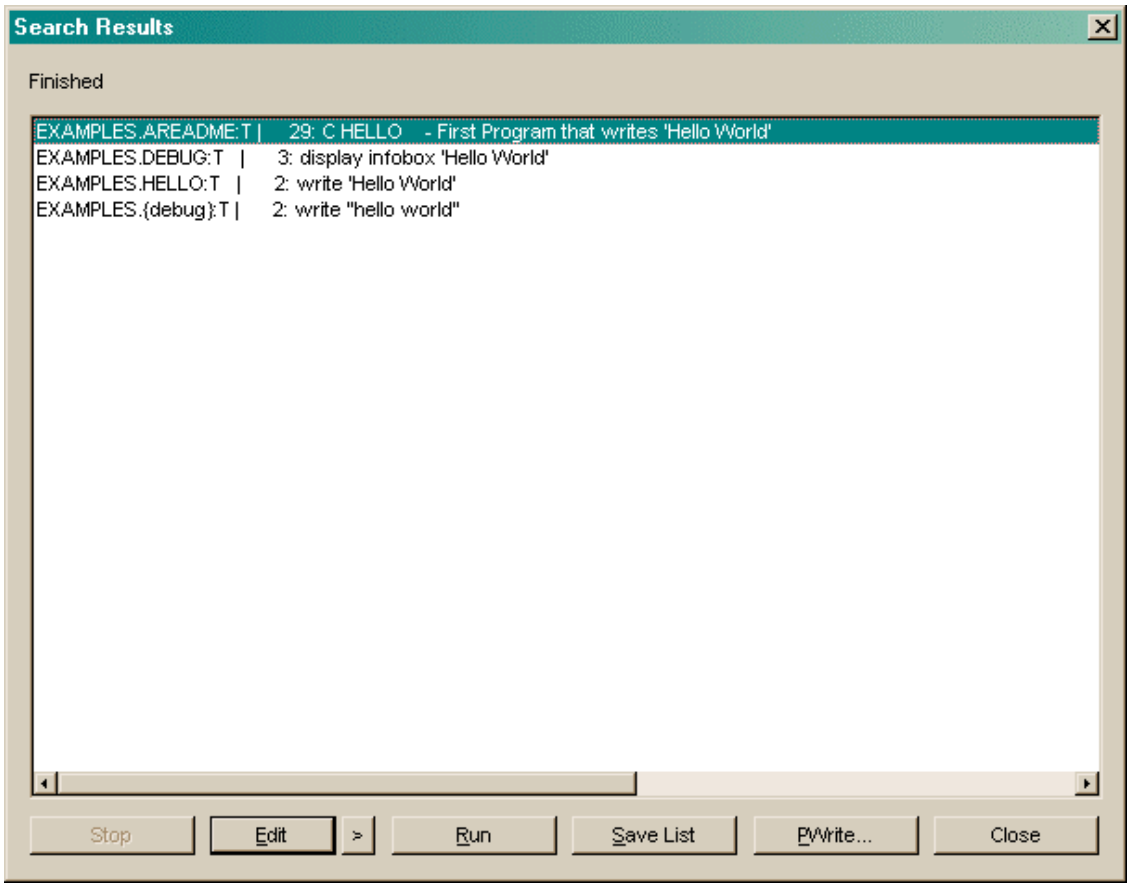
Search Members

Use the **Family** and **Member** expressions to find or search members with names that match the pattern.

Check the **Text** box and enter text to search for within the selected members.
The search can be **Case sensitive** and the text must match exactly. The search can be limited to **whole words**, for example the string "EXAM" is not found in "This is an example" if the Whole Words box is checked.
You can limit the number of **lines** that are read in each member. If you wanted to find all members that included the word "GENERATED" on the first line then enter "1" in the Lines box. If this is blank then all lines are searched.

Use the **Created**, **Modified** and **Byte Size** fields to search for members with these attributes.

Press the **Search** button to start searching. If members are found matching the search criteria then a dialog is displayed:

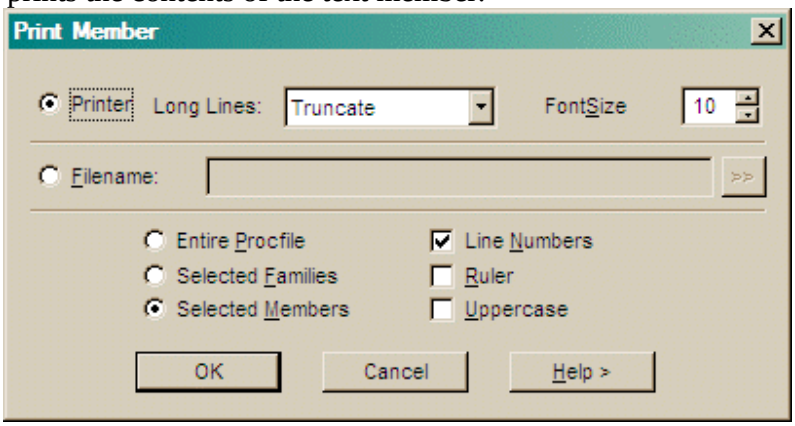


Found Members

From here you can **edit** or **run** selected members in the list or **save** the text of the list to a file.

Print Member

Formats and prints the contents of the text member.



Print Member(s)

Check **Line Numbers** to print line numbers;
Check **Column Numbers** to print a heading showing column numbers at the top of each page;
Check **Uppercase** to convert the text to uppercase before printing.

Select **Truncate** , **wrap** , or **page** to specify how long lines are printed. Truncate does not print any text that does not fit on the printed line; wrap prints text on the next line; page prints another page (or pages) with the long text.

You can optionally save the formatted output to a named file.

Import Members

Import Members (also known as PREAD) processes a file as generated by Export Members or the PWRITE command, creating or replacing text members in the procedure file.

Type in the name of the input file.
Use the "[>>]" button to find a file using the operating system file browser.

Check **NoInform** to suppress remarks in the output window;
Check **Replace** to replace existing Members;
Check **Public** to allow password protected Members to be Run without prompting for passwords;
Check **Confirm** to prompt for confirmation before replacing a Member;
Check **Report** to produce a summary report at the end of the process.

See the PREAD command.

Export Members

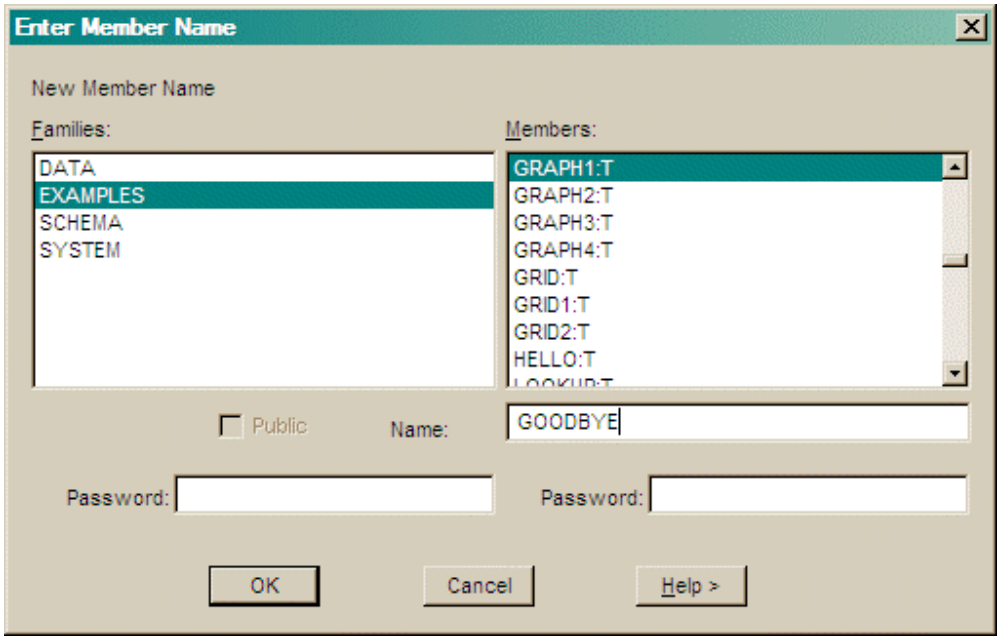
Produces a text file containing the chosen Member(s) in a format suitable for input to the PREAD command or Import Members.

Type in the name of a file to contain the Member(s).
Leave the filename blank and the information is written to the output window.
Use the "[>>]" button to find a file using the operating system file browser.

Check **NoInform** to suppress remarks in the output window.

See the `PWRITE` command.

Member Name



New Member

This generic dialog requires a member name for use with the previous dialog.

Type the member **Name**, that can include a family name (FAMILY.MEMBER).
Type an optional **Password** for this member.

Check **Public** to allow this member to be executed without a password.

Family Name

Supply a new Family name and optional password.

Compare Procfiles

Compare Procfiles lets you compare members between two procedure files or PWRITE files.

Selected members can be copied from one procedure file to another or from a PWRITE file to a procedure file.

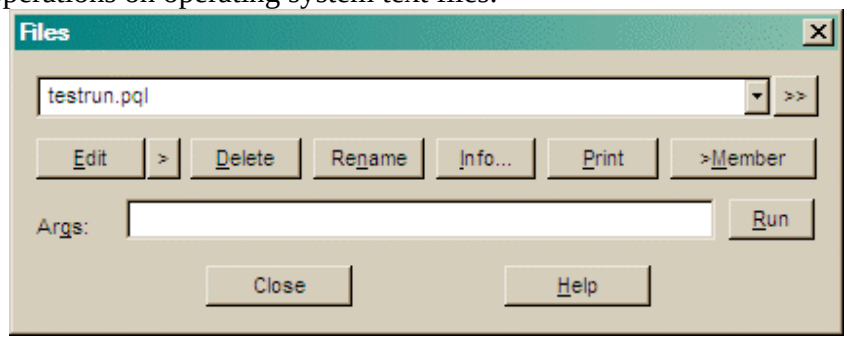
Select two procedure files or a procedure file and a `PWRITE` (export version of a procedure file);
Select **type** of members to compare (Text, or compiled executables).
Check the **size**, **date** or **contents** boxes to compare these attributes. If none of these are checked then members that do not exist in the one of the files are selected. **Size** selects members that differ by size; **Date** selects members that differ on modification date; **Contents** selects members if they differ by size or, if they are the same size, then the text is compared and the member is selected if different.

Press **Compare** to start comparison process.

After comparison the different members selected are in each list. Use the < or > buttons to copy selected members from one side to the other.

Files

Shortcut: **CTRL+SHIFT+F**
Performs operations on operating system text files.



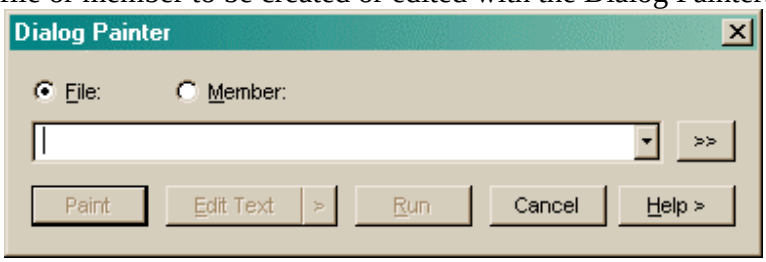
Files

Type in the **Filename**. Use the "[>>]" button to find a file using the operating system file browser.

- Press **Edit** to edit the file using your current text editor.
- Press **Delete** to remove the file. It cannot be undeleted.
- Press **Rename** to change the name of the file.
- Press **Info...** to display size and history information on the file.
- Press **Print** to print the contents of the file, maintaining any SIR/XS report formatting.
- Press **>Member** and you are prompted for the member to save this file in.
- Optionally enter **Arg**uments for substitution parameters in the file and
- Press **Run** to execute a set of SIR/XS commands stored in the file.

Dialog Painter

Shortcut: **CTRL+SHIFT+D**
Prompts for a file or member to be created or edited with the Dialog Painter.



Select Painter Input

PQLForms Painter

Prompts for a file or member to be created or edited with the PQLForms Painter.

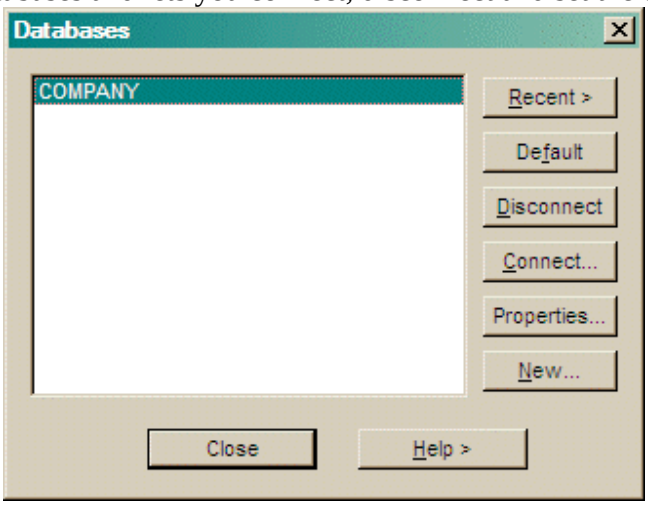
Use the create button to generate a default form from the current default database.

PQL Debugger

Starts the GUI PQL Debugger allowing a previously debug-compiled program to be selected and debugged. The use of the debugger is described in the a VisualPQL Reference manual.

Database Connection

Shortcut: **CTRL+D**
Lists connected databases and lets you connect, disconnect and set the default database.



Databases

Press **Recent** for a list of recently connected databases.

Select a connected database and press **Default** to set that database as the default.

Select a connected database and press **Disconnect** to disconnect that database.

Press **Connect...** to get the Open Database dialog.

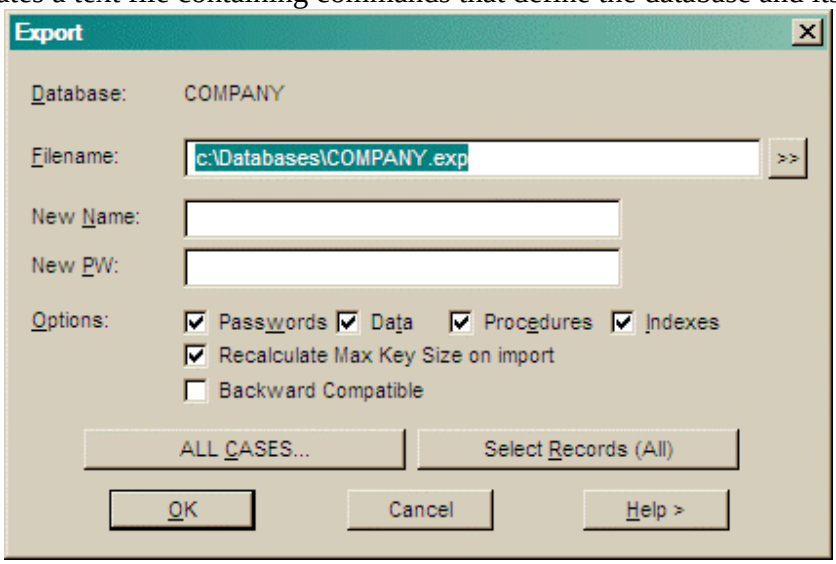
Press **Properties...** to display some information about the selected database.

Press **New...** to create a new database.

Press **Close** to close the dialog.

Export

Export creates a text file containing commands that define the database and its contents.



EXPORT Database

You must have DBA level access to the database.

An Export can be used to transfer a database from one operating system to another.

An Export file can be used as the input to the Import dialog, or simply run as a text file containing SIR/XS commands.

Enter a **Filename** to contain the Export. Use the "[>>]" button to find a file using the operating system file browser.
Optionally specify a **New Database** name and **New Password** for the exported database. The default is that of the current database.
Options

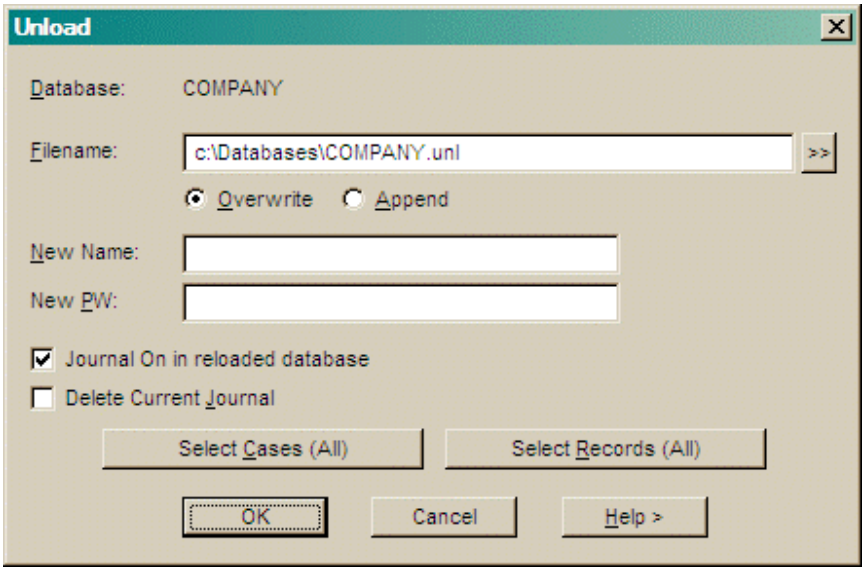
- Check **Passwords** to include the current database passwords to the export file.
- Check **Data** to include the data section of the database.
- Check **Procedures** to include the procedure file section of the database.
- Check **Indexes** to include the secondary index creation commands the database.
- Check **Recalculate Max Key Size on import** to suppress the generation of the MAX KEY SIZE command. This command is set from subsequent record definitions. If the current database has had a record type with a large key defined and then subsequently deleted then the new database inherits the large MAX KEY SIZE setting that may cause records and indexes to be larger than necessary.
- Check **Backward Compatible** to produce an export that is readable by earlier versions of SIR.

Press **Case Filter** to define a subset of case data for the output.
Press **Records** to define a subset of record data for the output.
See also the DBMS `EXPORT` command.

Note

There is no `IMPORT` command as such. The export file is a set of commands that, when run, rebuilds the database. An export file can be run like any file with SIR/XS commands.

Unload



UNLOAD Database

Creates a binary file containing a subset or a full copy of the database and data.

You must have DBA level access to the database.

The output file can be used to create a new database. A unload file can be used as input to the Reload dialog.

Enter a **Filename** to contain the Subset. Use the "[>>]" button to find a file using the operating system file browser.
Optionally specify a **New Database** name and **New Password** for the exported database. The default is that of the current database.
Check **Delete Journal** to remove the database journal file.

To unload a subset of the database, select cases and records using the following buttons:
Case Filter to define a subset of case data for the unload.
Records to define record types used in the unload.
See the `SUBSET` command. See the `UNLOAD` command.

Journal Upload

Creates a text copy of the journal file.

You must have DBA level access to the database.

The output file can be applied to a mirror copy of this database using the Journal Download dialog. This file is transportable across operating systems. Enter a **Filename** to contain the Upload. Use the "[>>]" button to find a file using the operating system file browser.
Enter the name of the **Journal** to upload. The default is the current database journal file.
Enter the database **Update Levels** to limit the upload. The default is the most recent update level.
Leave the first level blank to start from the first update in the journal. Leave the last blank to get all updates to the end of the journal.
Enter an optional **Title** for the upload. This title is ignored by the download procedure.
Press **Select Variables** to define the records and variable updates to upload.
See the `UPLOAD` command.

Journal Restore

Restores the current database to a different update level using a journal file.

You must have DBA level access to the database.

The current database could be a reload of an earlier version of your database for which you have subsequent updates stored on a database journal file (`JOURNAL RESTORE`), or you can restore the database to and earlier update level by undoing the journalled changes (`JOURNAL ROLLBACK`).
Note that changes to the schema cannot be undone. It is possible to rollback data changes from before some schema changes but this should be done with caution.

Use the "[>>]" button to find a **Journal File** using the operating system file browser.
Check **Rollback** and select an earlier update level to restore the database to;
Or check **Restore** and select a later update level to restore the database to;
If there are no appropriate journalled updates then the relevant option will be disabled.
See the `JOURNAL RESTORE` command.

Download Journal

Applies the changes made to one database to the current database.

Enter the name of the **Upload File** to apply. Use the "[>>]" button to find a file using the operating system file browser.
Check **Messages** if you wish to view any informative messages as the download is applied. See the `DOWNLOAD` command.

Verify Database

Check the database for integrity and corrects any errors if possible.

The Options can only be used by the Database Administrator (DBA).

- Check **CIRkey** to list index values in the CIR.
- Check **CIRdata** to list all variables in the CIR.
- Check **Check** to check each variable against its schema definition.
- Check **Ccf** to switch the corruption flag off.
- Check **Reckey** to list index values in each record.
- Check **Recdata** to list all variables in each record
- Check **Patch** to repair any repairable problems.
- Check **Rcf** to list record counts from the CIR.
- Use **Count** to examine a subset of cases in the database. Enter one, two or three numbers specifying a total number of cases, an increment or number of cases to skip and a starting case. (e.g. `4,2,5` checks cases 5,7,9 & 11)

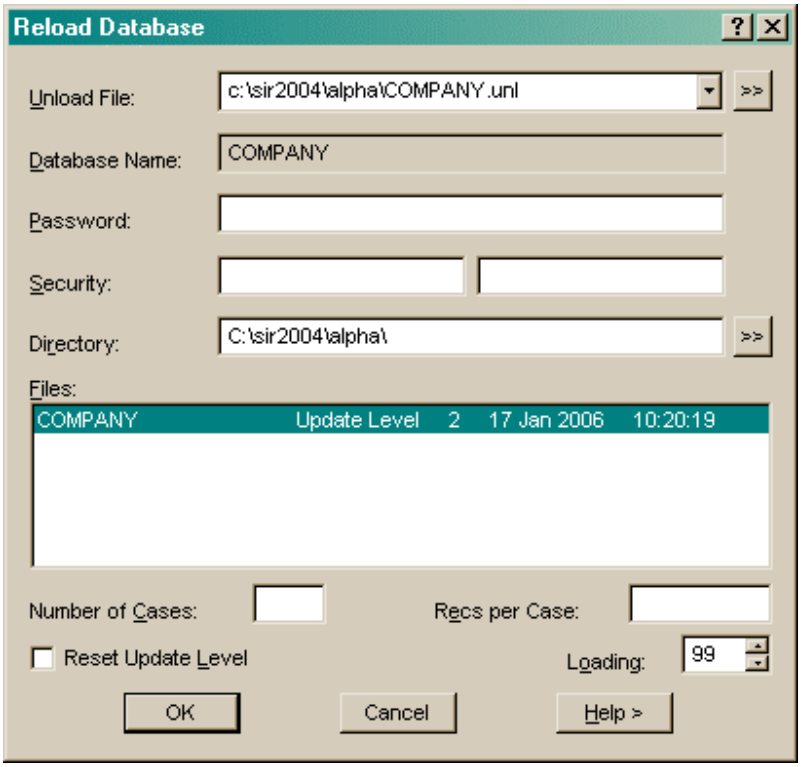
See the `VERIFY` command.

Itemise File

Reports on the contents of a SIR/XS binary file. That is an Unload, Subset or Journal file. Enter the name of the **File** to be itemised. Use the "[>>]" button to find a file using the operating system file browser.

See the `ITEMIZE` command.

Reload Database



RELOAD Database

Creates a new database from a SIR/XS Unload file.

Database

Enter the **Filename** that contains the SIR/XS Database Unload(s).

Use the "[>>]" button to find a file using the operating system file browser.
There may be more than one copy of the database in an unload file. By default the last (most recent) unload of a database is reloaded. Press **Itemize** to see if there is more than one unload on the file, if there are then you can change the **File Number** to select the update level to reload.

Password

Enter the database password if it exists. A valid password must be entered to attach to the database.
Enter a **Read Security** and **Write Security** password. These passwords determine the security level at which you access the database.

Enter a **Database Directory** where the database is to be created. If a database with the same name already exists here then the RELOAD fails.

Uncheck the **most recent database copy** box to select an earlier copy of the database if one exists on the file. You can use the **Itemize** button to view details of the unload file.

Optionally enter new values for `N OF CASES` and `RECS PER CASE` in the new database. These default to that of the original database.

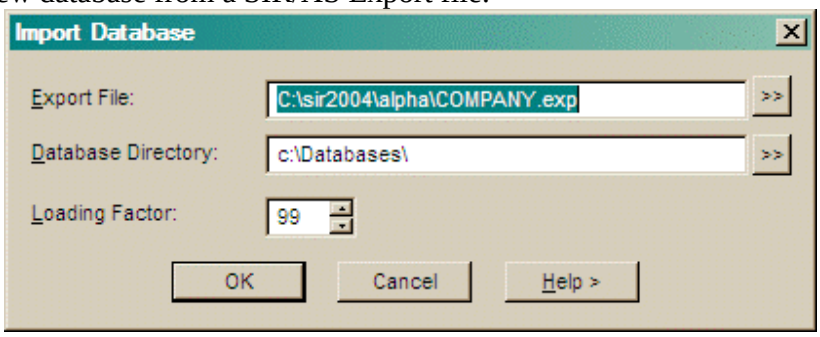
Check the **Reset Update Level** box to set the update level of the created database to one. The default is the update level of the unloaded database.

Enter a **Loading** factor between 0 and 99 (the default is 99).

See the `RELOAD` command. See the `ITEMIZE` command.

Import Database

Creates a new database from a SIR/XS Export file.



IMPORT Database

Enter the **Import file** name that contains the SIR/XS Export file. Use the "[>>]" button to find a file using the operating system file browser.
Enter a **Database Directory** where the database is to be created. If a database with the same name already exists here then the IMPORT fails.

Enter a **Loading** factor between 0 and 99 (the default is 99).

See `IMPORT`.

Delete Database

Deletes the current database data and dictionary from the disk.

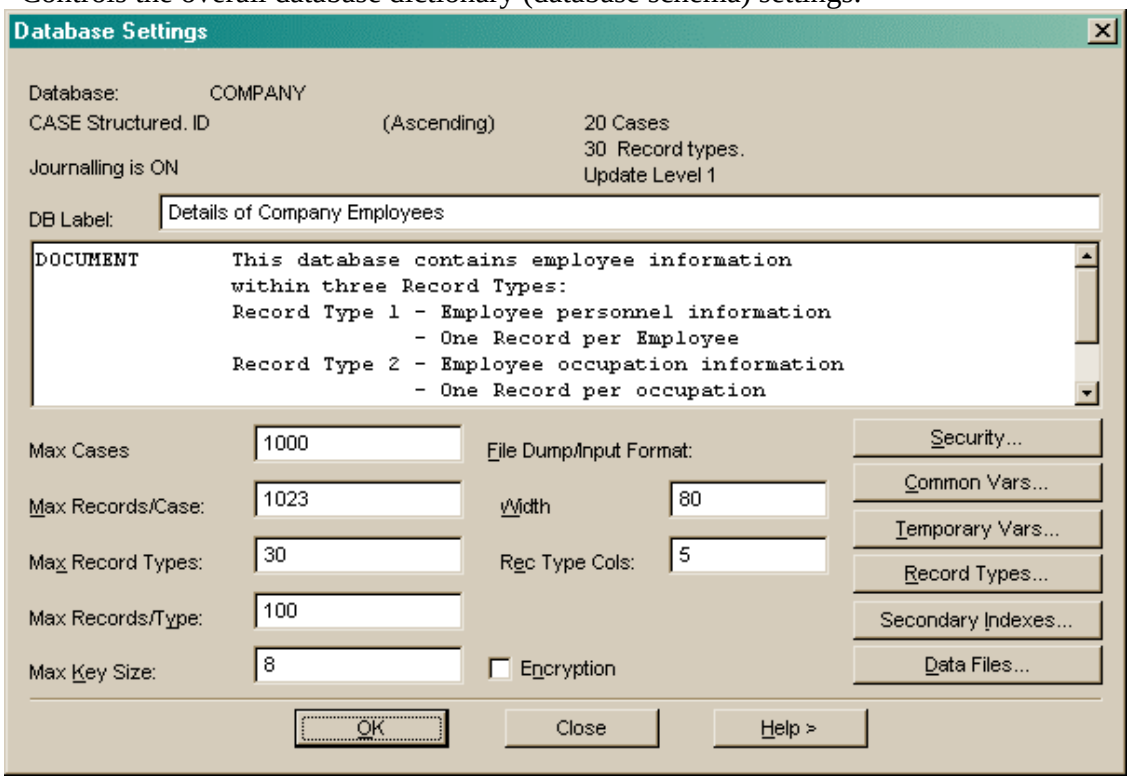
You must have DBA level access to the database.

Check **Delete Journal** to delete the current database journal.
Check **Delete Procfile** to delete the current database procedure file.

See the `PURGE` command.

Database Settings

Controls the overall database dictionary (database schema) settings.



Database Schema Settings

Documentation

The **DB Label** specifies a label of up to 78 characters to describe the database. The text area below can be used to modify more detailed commentary documentation stored with this database.

Maximums

These limits control data integrity and also determine the size of the common information record. If the size of the CIR is changed the database must be Unloaded and Reloaded.

- **Cases/Records** limits the total number of cases in a case structured database or records in a caseless database.
- **Recs per Case** limits the total number of records of any type in a case.
- **Recs per Type** limits the number of records of any one type in a database.
- **Record Types** limits the number of record types that can be defined in a database. Also no record type number may exceed this value.

- **Key Size** specifies the number of bytes required to store the largest record key.

External File Format

These control the format of the data when it is dumped to a text file or read using the input file utilities. They do not effect the structure of the database.

- **Max Input Cols** determines the record length of the dumped data.
- **Rectype Cols** specifies the columns containing the record number.

Press **Security** to modify database access security settings.
Press **Common** to update the list of database common variables.
Press **Temporary** to update the list of temporary variables.

See overall commands.

Database Security

The overall database access password is shown and can be modified here.

Database security is controlled by up to 31 levels of read and write access.

Specifying a read and write password associated with a particular level, at database connection time means you have that level of access to the database.

The Database Administrator (DBA) must have the highest level access (Read 30 / Write 30).
System Level is the minimum Read level of access required to use the UNLOAD FILE Utility. The default is 30.
Common Read Level is the DEFAULT level of access required to read data stored in Common Variables. The default is 0.
Common Write Level is the DEFAULT level of access required to update data stored in Common Variables. The default is 0.

The current read and write passwords are listed.
To define new passwords and associate them with access levels Type the **Level** (0 to 30), the new **Password** and press the **Read** or **Write** button. To remove an existing password, type the level and leave the password blank.

Common Variables

Common Variables are a set of variables stored in the Common Information Record (CIR).

A list of the current Common Variables is displayed using the standard record schema dialog.
Note that many options on this dialog do not apply to common variables or the CIR and have been disabled.

You can enter a **new** common variable and press **Add**, or **Delete** existing variables. You can also modify the variables labels, ranges and other meta data using the **Detail...** button.

Press **OK** or **Apply** to update the common variable definition. Press **Save As...** to save the common variable definition commands to a file without applying the changes to the schema.

See the `COMMON VARS` command.

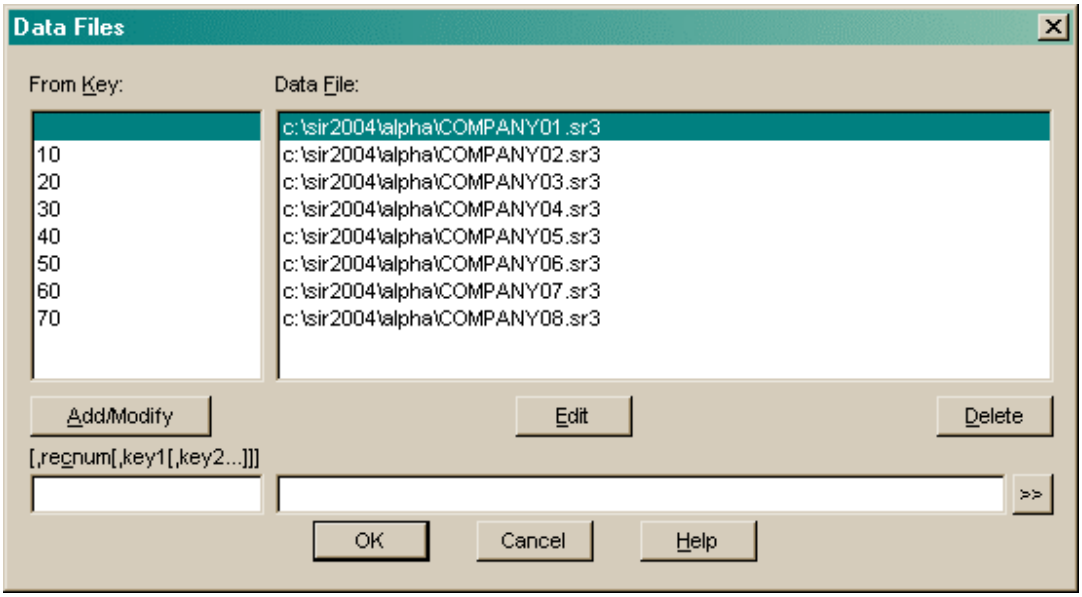
Temporary Variables

Temporary Variables are used in computations during batch data input.

The computations are defined in the record schema definition and the values of these variables are not stored in the database.

A list of the currently defined Temporary Variable is displayed.
Type a new **Name** and press the **Add** button to define a new Temporary Variable.
See the `TEMP VARS` command.

DATA FILES



Data Files

Specifies that the data file for this database is not a standard data file. It may have a different name, be in a different directory or may be split across multiple data files.

Type in a Key made up of the Case Id and optionally a record type and part of a record key. For a caseless database, just enter a record type and part of a record key. Enter a filename and press **Add/Modify**. Any records with keys after the entered one will be stored in that file. If the key is blank then any records before the first entered key. If data files are specified then there must be exactly one blank key.

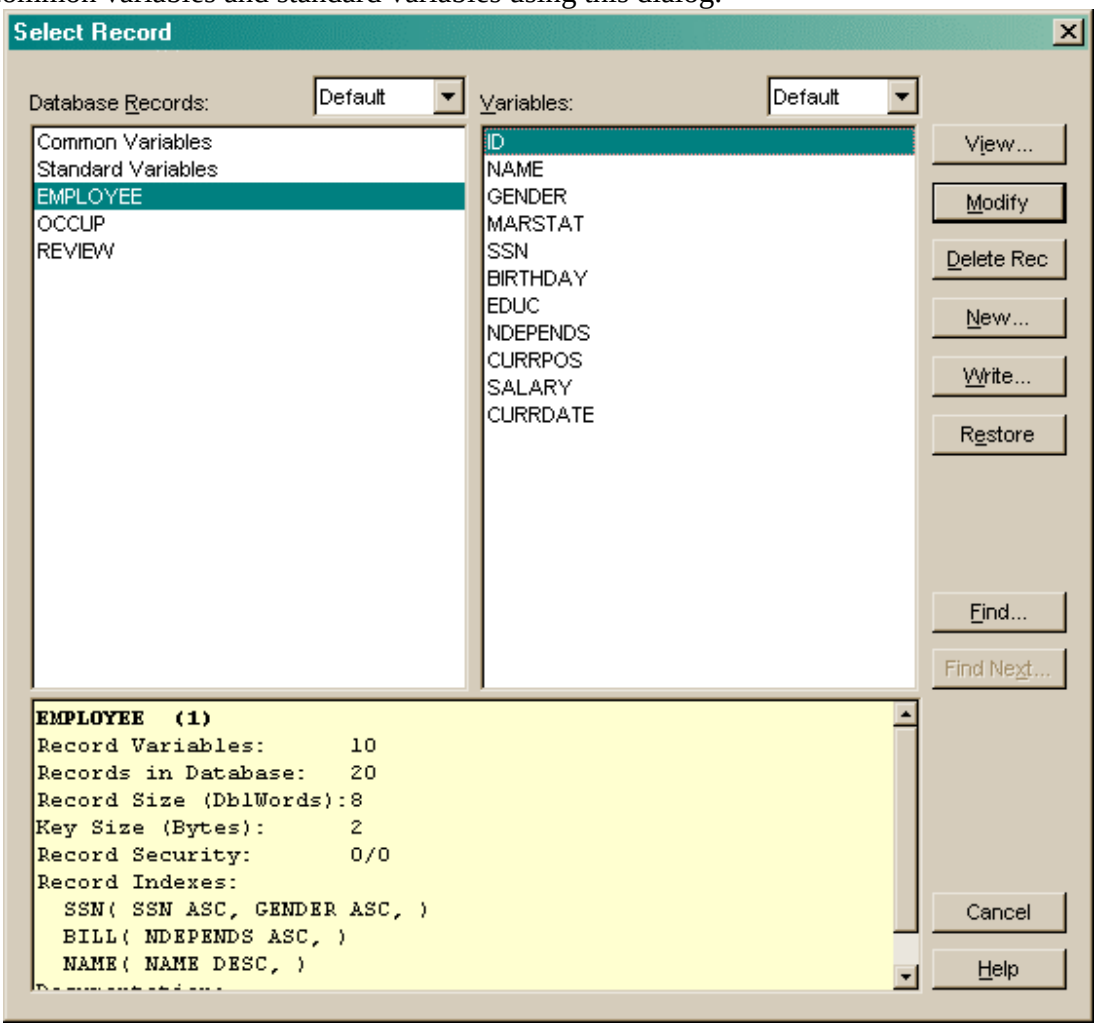
Press **Edit** to copy a selected existing data file definition to the text areas below for editing.

Press **Delete** to remove a data file definition.

Press **OK** exit and save the data file definitions. Note: the data files are not applied until the database is rebuilt by unload/reload or export/import

Record Schema

Creates and modifies record definitions in the database dictionary. You can also modify common variables and standard variables using this dialog.



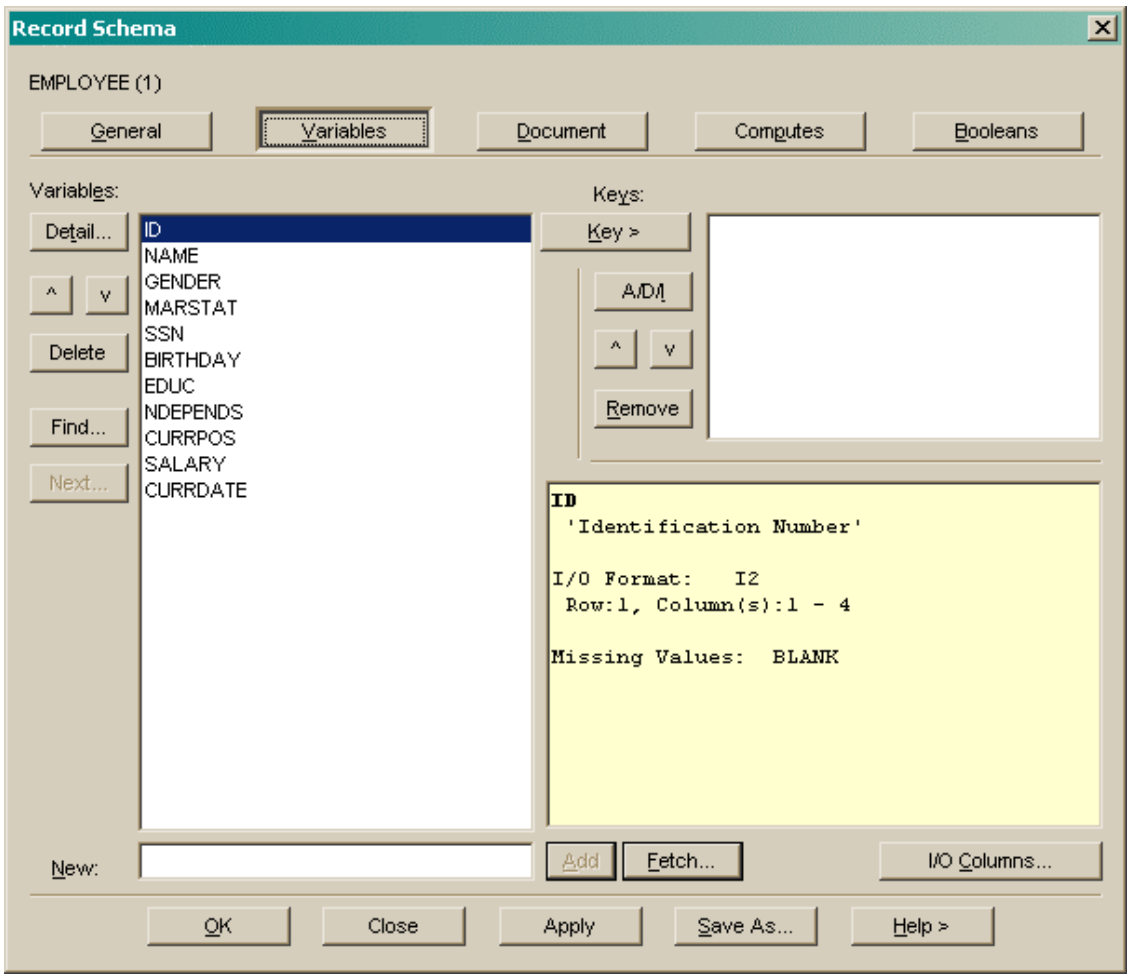
Select or Create a Record Type

Press **View...** to look at the record definition without risk of changing any settings.
To **Modify an existing record**, select a record from the list and press **Modify...**.
Press **Delete** to remove the schema definition (and any record data) from the database.
To **Create a New Record**, Press **New...**, select a record number and enter a **Record Name**. If there are existing records you can choose to start modifying from a copy of an existing record's definition. You can also just create a copy of the record schema and all the data records (but this may take some time).
Then press **OK**.
If you have previously used these dialogs to modify a schema and saved a definition to a file then press **Restore** to continue with that definition.

You can save record schema commands to a file using the **Write...** button.

Record Schema Definition

This is the main record schema definition screen.



Record Schema Settings

The record **Name** and **Number** are displayed.

General

Optionally enter a descriptive **Record Label** of up to 60 characters.

Check **LOCK** to specify that, if the schema modification requires it, that this record type may be locked after the changes are made. The database must then be restructured using unload/reload before the data in the record is accessible. If lock is not requested and the modifications require the record be locked then an error message is displayed and the modifications are not applied.

Enter the **R/W Security** levels for accessing data in this record. Specify one or two numbers from zero to thirty separated by a blank or slash. If one number is specified then this is both the Read and Write security level required to access this record data.

The **Max Rec Count** limits the number of records of this type in one case. If there are no key fields defined then this number must be 1.
Max Input Cols is an overall database setting that can be increased if required.
Lines/Rec sets the number of lines for multi lined input/output files.

Variables

A list of the current record variables is displayed. Left of the list are the buttons:

- Press **Detail...** to view and change the definition of the selected variable.
- Press ^ to move the selected variable **up** the list of variables.
- Press **v** to move the selected variable **down** the list of variables.
- Press **Delete** to remove the selected variable from the definition.
- Press **Find...** search the list for a variable. You can search by name, var label or value labels.
- Press **Next...** find the next matching variable from the search.

To add a new record variable to the record definition, type in the name at the **New:** prompt and press **Add**.
New variables are inserted after the currently selected variable in the list.
You can also add variables and copy their meta data from other variables using the **Fetch...** button. If you enter a new name and press Fetch then that new name is used, otherwise the variable name is copied from the chosen variable. Select a variable from another record definition and press OK.

Keys

A list of the current record **key** variables is displayed. Left of the list are the buttons:

- Press **Key>** to make the selected record variable a **Key Field** for this record.
- Press **A/D/I** to toggle the sort order of the selected key (**A**scending , **D**escending , or Auto**I**ncrement).
- Press ^ to move the selected variable **up** the list of keys.
- Press **v** to move the selected variable **down** the list of keys.
- Press **Remove** to remove the selected variable from the list of key variables.

I/O Columns

Input Output columns are used when data is dumped to or read from a text file. These are not required and do not effect the way the data is stored in the database. The I/O columns screen helps you modify the format of dumped data file.

Select items from the data list and press **Auto Number** to assign non-overlapping column numbers to the selected items.
De-Number removes column numbers from the selected items.
Move Up and **Move Down** change the order in which the variables are defined.
Remove and **Insert** can be used to cut and paste items from one place in the list to another.
Sort changes the order of definition to match the input column order.
Change the I/O position of a single item by selecting it, changing the details at the bottom and pressing the **Change** button.

Document

Lets you view and modify the database record documentation.

Computes

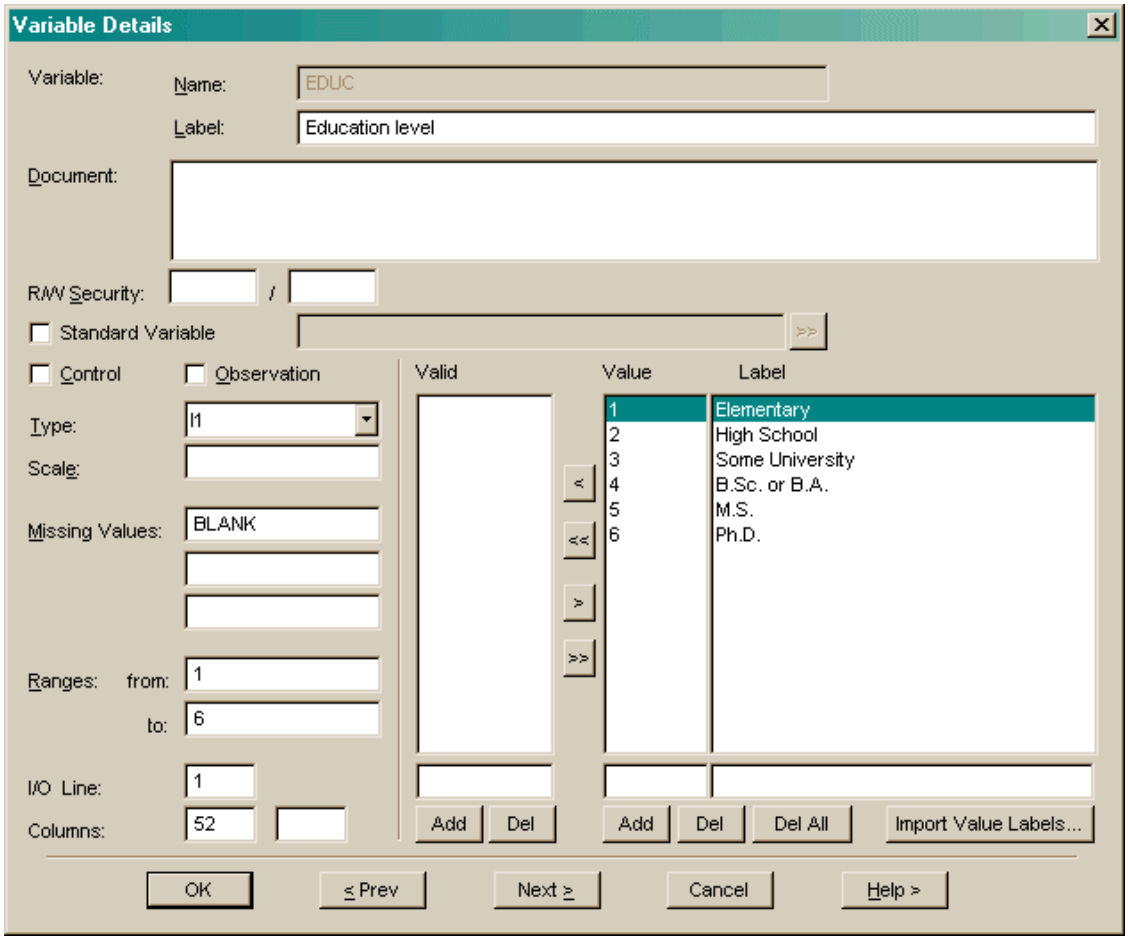
Lets you view and edit the **batch data input recodes**, **computes** and **ifs** for this record.

Booleans

Lets you view and edit the **batch data input accept** and **reject** conditions for this record.

See record commands. Press **OK** to apply the changes and exit.
Press **Apply** to apply the changes and not exit.
Press **Save As** to save the changed definition to a file. This saved file can be restored and the record definition continued.

Record Schema Variables



This is the main screen for variable detail definition.

The name of the current **Variable** is displayed.

Label

Optionally enter a descriptive **Label** for the variable. This label appears in reports and tabulations.
You can also include a long variable **description** in the Document box.

Enter the **R / W Security** level to access this variable data (e.g. `20,30`).

Standard Variables

Standard variables are defined in a *standard schema*. Define a record variable as having the same meta data as a standard variable by checking the Standard Variable box and

selecting one of the previously defined standard variables. When a variable is a standard variable, you can give it its own label or security level, overwriting the standard definition.

Type

New variables default to four byte integers. Change the type as required. Choose:

- **Integer** and enter the internal length in **bytes** (1, 2 or 4).
- **Real** and enter the default number of **decimal places**.
- **String** and enter the maximum string length.
- **Date** and enter the **date format**.
- **Time** and enter the **time format**.

Enter a **Scale Factor** if this variable is a **scaled integer**.

Numeric variables can have a **Statistical Type**.

- Choose **Auto** to let the software select the appropriate statistical type. Variables with **valid values** or **value labels** are generally control variables.
- Choose **Control** if this variable is a category (e.g. `Gender`, `Type`, `Department`).
- Choose **Observation** if this variable contains a continuous measurement (e.g. `Salary`, `Age`, `Duration`).

Missing Values

Enter one, two or three values to be missing values for this variable. The keyword `BLANK` can be used.

Minimum & Maximum Value

Use Minimum and Maximum to define a valid range for this variable. Specify both Minimum and Maximum or neither.

Output Format:

Define the input / output format for this variable. This format is used by the File Dump and File Input utilities.

- Type the **Line** number for multiple line / record input/output.
- Type the start (and end) **Columns** for this variable.

Valid Values

Lists the values of a variable that can be accepted into the database. Missing values are not included in the valid value list.

If a string variable has valid values it becomes a categorical string variable (or CATVAR) and is stored internally as a number corresponding to the valid value.

- Press **Del** to remove the selected valid value from the list.
- Type in a new **Value** and press **Add** to add a value to the list.

When entering valid values the Add button becomes the default action so that you can quickly enter a list of values by typing and pressing enter after each value.

Value & Label

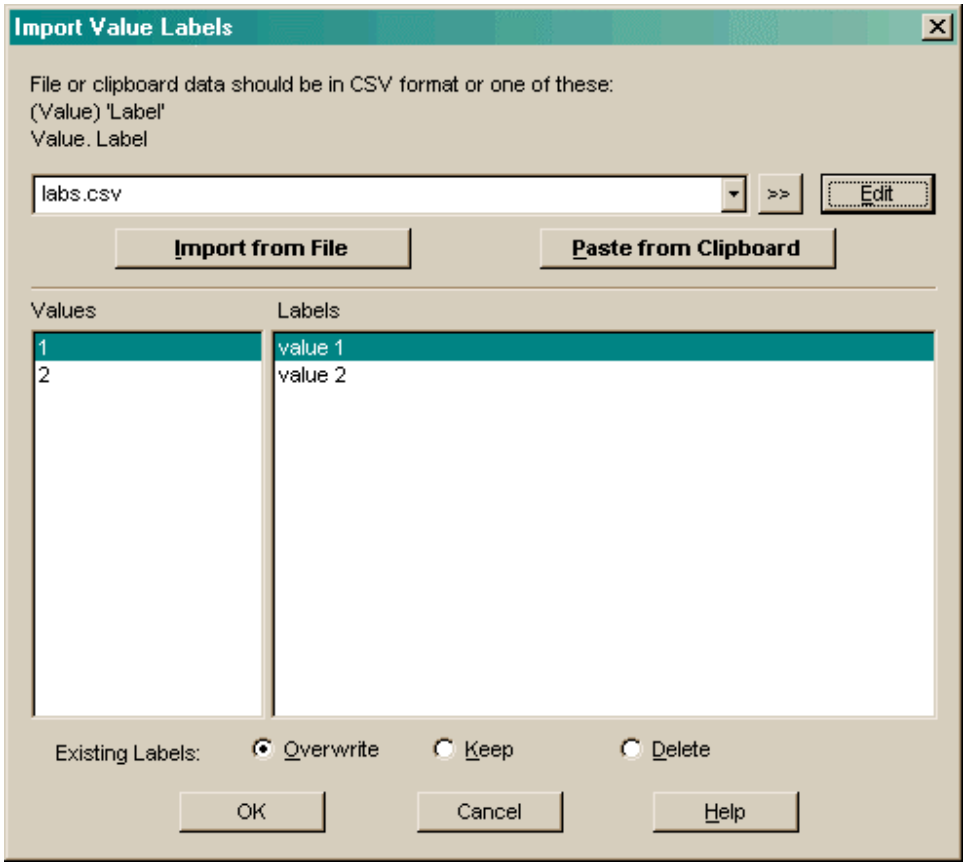
Lists the labels associated with values. This can include labels associated with missing values.

- Press **Del** to remove the selected valid value and label from the list.
- Press **Del All** to remove all valid values and labels from the list.
- Type in a new **Value** and press **Add** to add a value to the list.

Press **Import Value Labels** to get labels from a file or the clipboard.

Press **Previous** to save any changes to this variable and get the definition from the previous record variable.
Press **Next** to save any changes to this variable and get the definition from the next record variable. You can also use the Find... button to search for a variable.

Import Value Labels



Lets you import values and labels from an external file or from the clipboard.

The file or copied text should be either in comma separated value (CSV) format or lines containing values a labels. For example:

```
value. label
(value) 'label'
"value" "label"
...
```

You can either select a file with a predefined set of labels, create a new file and enter values and labels using a text editor then press **Import From File**; *or* copy labels from an external document and press **Paste from Clipboard**.

Use the **Overwrite**, **Keep** or **Delete** radio buttons to specify what happens to existing labels on this variable:

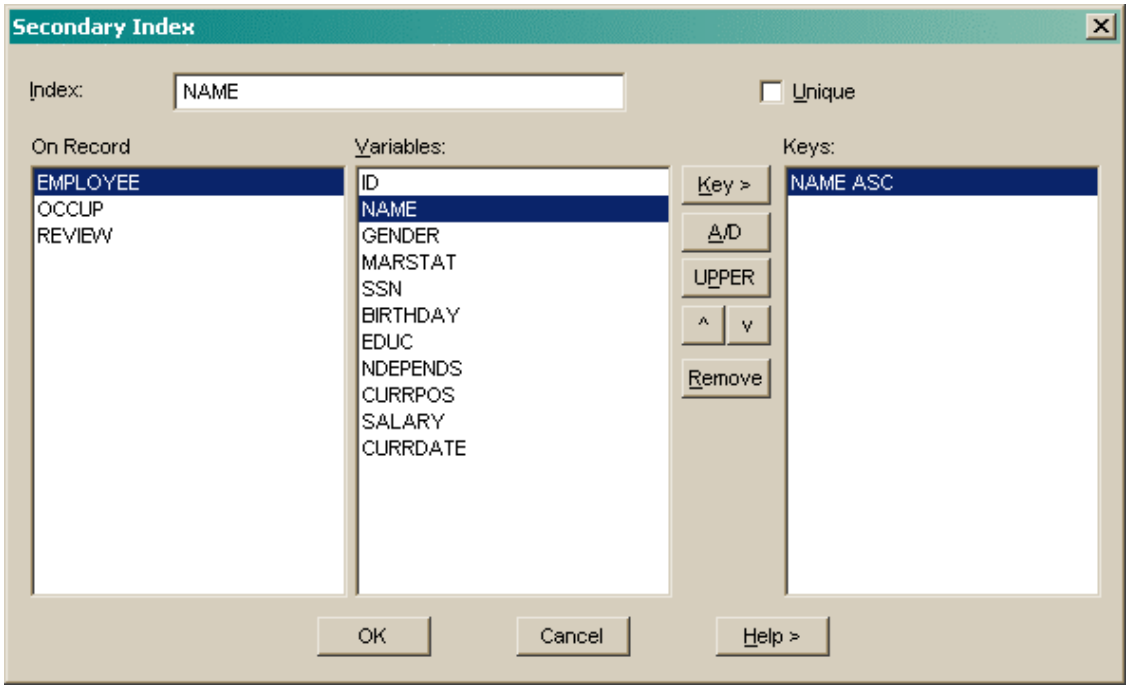
- **Overwrite** will replace any old label for a duplicate value with the imported one.
- **Keep** will keep the old label for a duplicate value.

- **Delete** will remove all the old value labels and replace with the new list.

Database Secondary Indexes

Lets you define and modify secondary indexes on database records. Note that if you redefine an index then the index is deleted and recreated.

Select a secondary index from the list and press Modify, or press New to create a new index.



The index must have a valid SIR/XS name and this name must be unique to this database.

Check the **Unique** box to create an index that points to a single record (e.g. Social Security Number on the Employee record). This ensures that no two records are added with the same value for the indexed variable(s).

Select a record and variable(s) to index on. Use the **Key >** button to add the variable to the key list.

Once a variable is in the key list:
use the **A/D** button to toggle the sort order;
Use the **UPPER** button to indicate that string keys are treated as case insensitive (converted to uppercase);
Use the **v** and ^ buttons to change the hierarchy of the keys;

Import records from ODBC

Lets you create record definitions and read data from **ODBC** data sources or the **SirSQL Server**. There are several dialogs to guide you through this procedure.

Import Data Source

Choose between **ODBC** and the **SQLServer**.

- **ODBC** Type the **name** of the ODBC source. These names are set up in the **Windows Control Panel** under **ODBC**, **User DSN**. If the data source has passwords associated with it then enter these in **Password** and **Security**.
- **SirSQL Server** Type the address and port of SirSQL Server.
  Enter the **Database** name.
  Enter the full Prefix (or path) to the database **on the machine running the SQLServer**
  Enter any database **Password** and **Security** passwords.

  Press **Next >** and SIR/XS attempts to connect to the data source. If it fails an error message is displayed.

Select Table and Columns

A list of Tables and Columns is displayed. Select one Table and any or all of the Columns. You can enter a **Where** condition to select a subset of table rows. Press **Next >** and SIR/XS attempts to query the data source.

Save Data In

Choose the database record type where you want to place the new data.

- Select **Create New Record Type** or **Use Existing Record Type**.
- Select a **Record Number** from the list of available numbers.
- Enter a **Record Name** if this is a new record type.
- **Case ID** If this is a case database, choose where the case id on the imported data is coming from. It can be either the first sort variable selected below or a constant entered in the box beside.
- **Variables** lists the columns to be imported. Select any sort variables from this list and press: **Sort >** to append to the list of **Sort Variables**. If this is an existing record type then choose the same number of sort variables as defined in this record's schema.
- Press **A/D** to reverse the sort order of a variable (**A**scending or **D**escending). This only applies to new record types.

- Press ^ **v** change the order of the sort variables.
- Press **Remove** to remove a sort variable from the list of sort variables.

Press **Next >** to see a list of source and target variables. You can change the target variable names and types. If this is an existing record type, select the target names from the list of available names. If it is a new record then specify any name or type of target variable.

Press **Next >** and you are prompted to save a data update program. This program can be run at any time to read the data source into the defined record type without needing to go through these steps again.

If this is a new record type it is created. There is a possibility that the creation of the record type may require a database restructure. If this is the case, unload and reload the database and then run the data update program that you saved above.

Write Schema

Produces output based on the database schema.

Choose **Commands** to produce a copy of the schema definition.
Choose **PQLForm** to produce a default data input PQLForm definition. This can be used to input data into the database.
Choose **Form** to produce a default data input form definition. This can be used with SirForms to input data into the database.
Choose **Report** to produce a report documenting the structure of the database.

Enter a **Filename** for the output. If no filename is entered then the output is displayed in the main window. Use the "[>>]" button to find a file using the operating system file browser.

Options

Not all options apply to each output type. Non applicable options are disabled.

- Check **Passwords** to include the listing of passwords in the output.
- Check **CIR** to include the definition of the common information record in the output.
- Check **Group by Variable** to include all the meta data for each variable in the same place. The default is to group by the type of meta data.
- Check **Expand TO lists** to explicitly list all variables in meta data definition commands. The default is to group variables with the same definition into TO lists. For example `MISSING VALUES NAME to CURRPOS (BLANK)/.`
- Check **Booleans** to include any accept and reject record clauses in a form.
- Check **Computes** to include any compute clauses in a form.
- Check **Short** to suppress the printing of long value labels.
- Check **Structured** to include information on the case id, record key fields and documentary text.
- Choose **Regular** to list most of the variable information; **Detailed** to include value labels for each variable; **Labels** to include variable names and labels.

Press **Records** to define a subset of record data for the output.
Press **Variables** to define the variables included in the output
Press **Database Help** to view details on the structure of the current database, records and variables.

See the `WRITE SCHEMA` and `SIR SCHEMA LIST` commands.

List Stats

Produces a status report about the database showing overall statistics and a line per defined record type.

See the `LIST STATS` command.

Control Tabfiles

Shortcut: **CTRL+T**

Lists connected tabfiles and lets you connect, disconnect, view details of, and set the default tabfile.

Press **Recent >** to display a pick list of recently connected tabfiles.

Press **Connect...** to browse the file system for tabfile files.

Press **New...** to create a new tabfile.

Press **Delete** to delete the tabfile.

Press **Details** to show tabfile information.

Select a connected tabfile and press **Default** to set that tabfile as the default.

Select a connected tabfile and press **Disconnect** to disconnect that tabfile.

Press **Close** to close the dialog.

Tabfile Information

Displays a list of tables defined on the selected tabfile. The selected tabfile can be changed using the pull down list at the top.

Tables can be added with the **New...** button or viewed with the **Details...** button.

.

Table Information

Displays Table information. Press the **Columns** button to view the variables in the table. Select a variable to see its type, label and ranges.

Press the **Indexes** button to view and modify the indexes on the table.
Select a secondary index from the list or press New to create a new index.

The index must have a valid SIR/XS name and this name must be unique to this tabfile.

Check the **Unique** box to create an index that points to a single row (e.g. Social Security Number on the Employee table). This ensures that no two rows are added with the same value for the indexed variable(s).

Select the variable(s) to index on. Use the **Key >** button to add the variable to the key list.

Once a variable is in the key list:
use the **A/D** button to toggle the sort order;
Use the **UPPER** button to indicate that string keys are treated as case insensitive (converted to uppercase);
Use the **v** and ^ buttons to change the hierarchy of the keys;

Variable labels and ranges

The left half of this dialog shows labels defined for this variable. The right half shows valid values and ranges.

Disconnect Tabfile

Closes the connection to the selected tabfile.

Default Tabfile

Sets the chosen tabfile as the default. Tabfile operations where the tabfile is not explicitly named apply to the default tabfile.

View Tabfile

Displays details of the chosen tabfile.

The **Tabfile** name is displayed along with its attributes.

Tables

A list of the tables in the tabfile is given.

- Press **New** to create a new table for this tabfile.
- Press **Delete** to remove the selected table from the tabfile.
- Press **View** to see details of the selected table.

Indices

A list of the indices on the selected table of the tabfile is given.

- Press **New** to create a new index for this table.
- Press **Delete** to remove the selected index from the tabfile.
- Press **View** to see details of the selected index.

Table Details

Shows the columns (or variables) and their data types for the selected table.

Index Details

Shows the columns (or variables) used and their sort sequence for the selected index.

Create Index

Creates a new index for the chosen table on the tabfile.

The tabfile and table names are displayed.
Enter the new **Index Name**.
Check **Unique Index** if only one instance of the selected keys is allowed in this index.
Enter a **Percent Free** for this index. This is between 0 and 99 and indicates how much free space is left in each index block. Use small numbers for stable, read only tables and larger numbers (e.g. 50) for dynamic tables.

Columns

A list of the columns in the table is displayed. Press **Key>** to add the selected column to the list of keys for this index.

Keys

A list of the current keys in this index is displayed.

- Press **A/D** to toggle the sort order of the selected key (**A**scending or **D**escending).
- Press ^ to move the selected key **up** the list of keys.
- Press **v** to move the selected key **down** the list of keys.
- Press **Del** to remove the selected key from the list of key variables.

See the `CREATE INDEX` command.

Connect Tabfile

Attaches a tabfile as the named user in the specified mode.

Enter the **Tabfile Name** and, optionally the **Filename**. If you don't specify a filename, it is set to the tabfile name with extension `.tbf` in the default directory. Use the "[>>]" button to find a file using the operating system file browser.

Security

Enter **Group Name**/**Password** and **User Name**/**Password** if these are required for this tabfile.

Mode

Choose **auto** to open the tabfile in read or write mode as needed.
Choose **read** to open the tabfile in read only mode.
Choose **write** to open the tabfile in read/write mode.

See the `CONNECT TABFILE` command.

Create Table

Creates a new table on this tabfile.

The **Tabfile** name is displayed.
Enter the new **Table** name.

A list of the currently defined **Columns** is given.

- Press **Del** to remove the current column.
- Enter a **New Column** name, select a variable type and press **Add** to add a new column to the table.

Verify Tabfile

Performs an integrity check on a tabfile. The tabfile need not be connected.

Enter the **Tabfile Name** and, optionally the **Filename**. If you don't specify a filename, the tabfile name with extension `.tbf` in the default directory is used. Use the "[>>]" button to find a file using the operating system file browser.

See the `VERIFY TABFILE` command.

Run SIRSQL

Run SirSQL starts the SIR/XS SQL module as a new process. SirSQL allows you to use ANSI Standard SQL to query existing data, modify that data and to define new tables, indexes and views.

File Attributes

File attributes are SIR/XS names associated with operating system filenames. All file references in programs are to the internal name that allows applications to operate on different physical files without altering program code.

For example, the attribute for the first file in the first database is always `SIR011`. This is mapped to the specific operating system file such as `C:\mydir\myproject\projdb.sr1`.

Internal names may be specified by a program, may be the filename if it meets the specification of a SIR/XS name or may be system generated such as `SYSUSR1`. Attributes are created automatically when you refer to a file just using a long operating system filename, but you can also create and amended them specifically as necessary. This then allows you to refer to that file using a standard SIR/XS name.

In general, when specifying a full operating system filename that does not meet the criteria for a standard SIR/XS name, it is best to enclose the filename in quotes.

Buffers

SIR/XS Buffers are temporary text storage areas that exist until deleted or the SIR/XS session is terminated.

Buffers may be created by the menu system or by user programs. A list of the current buffers is displayed for selection.

- Press **Edit** to edit the file using your current text editor.
- Press **New** to create a new buffer.
- Press **Delete** to remove the buffer.
- Press **Clear** to delete the contents of the buffer without removing the buffer.
- Press **Run** to execute a set of SIR/XS commands stored in the file.

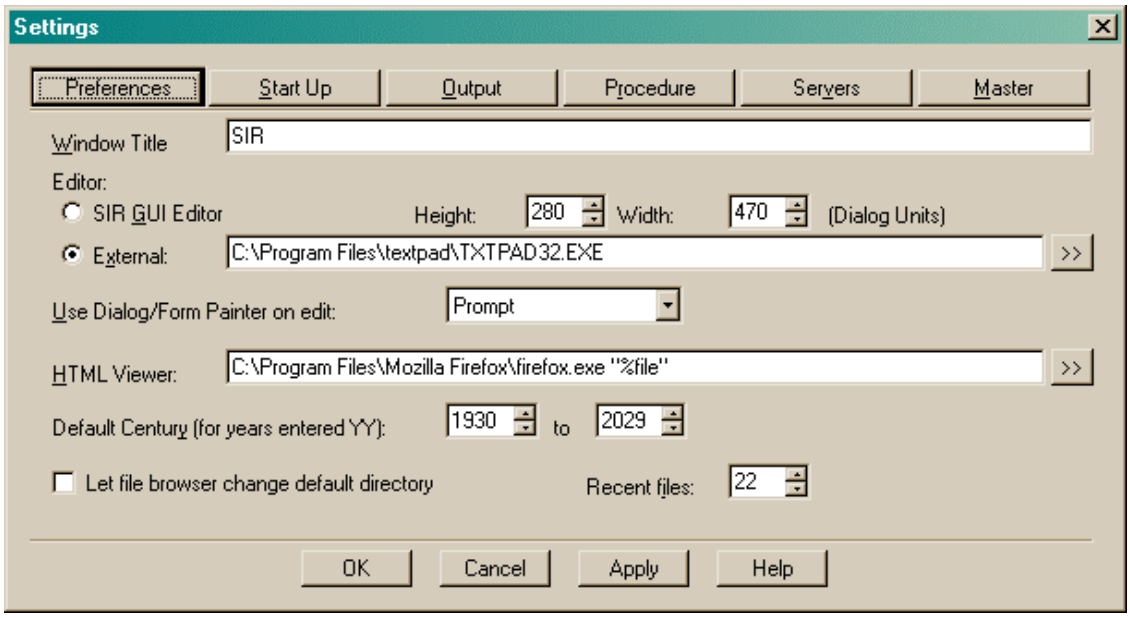
Global Variables

Global variables are text strings that are substituted where the global name appears in angle brackets (<>), as a command is read. There are several System Globals, and these are not listed here.

- Enter a new or existing global name and value then press **Modify** to update the list of globals.
- Press **Delete** to remove the selected global definition from the list.

See globals.

Preferences



The settings in this dialog control this and future SIR/XS sessions. The setting are saved in the `sir.ini` file in the user's *My Documents* directory in Windows, or the .sir file in the user's home directory in Unix.

Window Title

Sets the text displayed in the main window title bar.

Editor

SIR/XS has an internal editor but also allows you to use your favourite editor in a seamless and integrated way for all your editing. Once set, the editor is used to edit procedure file members, text files and SIR/XS memory buffers.

To use a text editor that is already on your system, set the settings-preferences editor choice to *External* and enter (or browse for) the name and location of the executable editor you want SIR/XS to use. e.g.`C:\WINNT\Notepad.exe`. When you edit a member or a buffer, it is written to a temporary file for editing and, when you finish, it is copied back and the file is deleted.

The internal editor is a simple multi-line text graphical object. This is used in any dialogs and can be used directly if required. Set the settings-preferences editor choice to *SIR*. This editor works in a basic way as per normal text editing on your computer.

You can set the size of the internal editor dialog by changing the **Height** and **Width** (in dialog units) attributes on this dialog. This setting also effects the DBMS Command dialog.

Caution

When SIR/XS uses an external editor, it writes the member or buffer to a temporary file and spawns the external editor as a process. When that process terminates the temporary file is checked to see if it has been modified and if so, the file is copied back to the original location. Problems occur when the spawned editor passes the file on to another process and ends itself. This can happen when the file is too big for the editor (Windows NOTEPAD starts WORDPAD if the file is too big) or if the editor is set to only allow one instance to run (in which case the file is opened as a document in an already running instance of the editor). If this happens, the buffer or member is not updated after editing. Make sure you use an external editor that can handle large files and that it is set to allow multiple instances to run.

When a file or member contains suitably formatted code, it can be edited with the dialog or forms painter. You can get the painter to start automatically, never, or after a prompt when this type of file is open for editing.

HTML Viewer

The SIR/XS on-line help is in HTML format. Use the **>>** button to locate the executable file that starts an HTML browser (e.g. Mosaic / Netscape / Internet Explorer...).

In Windows this defaults to your default web browser.

Default Century

This defines the one hundred year default range for dates entered with **only two digits for the year**.

Let file browser change default directory

Checking this option makes the file browser start browsing in the last directory you left the browser in previously. The default is to start the browser in the current default directory (where SIR/XS was started or the directory explicitly set using the SETDIR function).

Recent Files

Set the number of filenames that are remembered for the *recent files* drop down lists in various dialogs.

Start Up

These settings are not applied until you restart SIR/XS.

Novice Welcome

On starting SIR/XS with no database connection parameters, a "Welcome to SIR/XS" dialog can be displayed. Uncheck this box to suppress this welcome dialog.

Flat Toolbar Buttons

Toggles the appearance of the toolbar buttons.

Highlight Toolbar Buttons

Check this option will dim the toolbar buttons. When the mouse is over a button it will be highlit.

Font Size/Name

On Windows, enter a number (around 10) to set the size of the text used in the main menu and dialogs. On other machines enter the name of the font to be used. (the unix command `xlsfonts` lists available font names).

Splash screen / Duration

On Windows, uncheck this box to suppress the graphic displayed when SIR/XS is started. Check the box and enter a number in the **duration** box to set the number of seconds this graphic is displayed.

MouseOver Delay

Sets the time (in seconds) that the MOUSE OVER message will be sent after the mouse has been stationary over a control. A negative number indicates that this message will never be sent.

Choice Drop Height

Sets the height (in dialog units) of the drop down list in choice and combo boxes.

Log File

Check the Log File box and enter a file name. This file is a log of the output window. This can be useful for debugging user menu programs that display error messages in the main window and then exit.

Output

These settings apply to this session only. The Page size, Error/Warning limits and Print back settings can be saved as defaults for future sessions.

Default Directory

Sets the default directory.

Output File

Sets a **Filename** to write output to. A blank filename (or the name CONSOL) directs output to the session window. Use the "[>>]" button to find a file using the operating system file browser.

Page Size

Sets the size of output file in character rows and columns. The number of rows applies to file output only and sets the number of lines on a page.

Error Warning Limit

Sets the limit on the number of error and warning messages displayed during the processing of input commands. Once the limit has been reached, processing continues but the messages are suppressed.

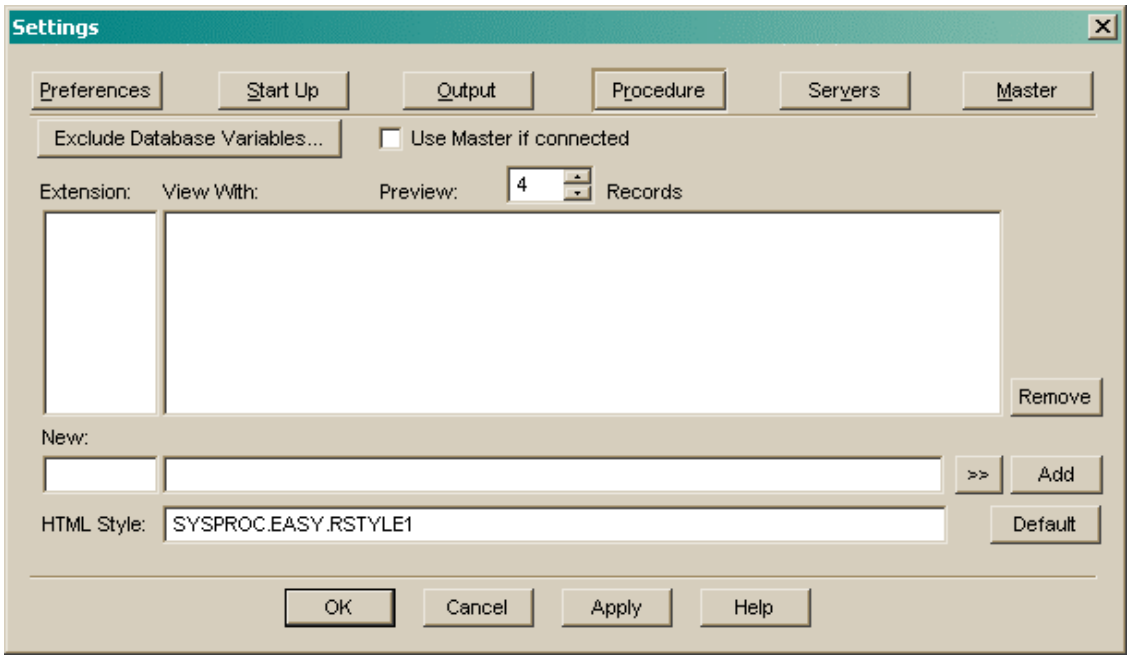
Print Back

Sets what types of messages are displayed or printed to the current output.
Print back options control the text written to the output file or output window.

Check **Remarks** to show system remarks and messages;
Check **Commands** to echo top level commands;
Check **Called** to echo commands generated from the CALL command;
Check **Do Repeat** to echo commands generated by DO REPEAT blocks;
Check **Skipped** to display commands not processed due to CIF (Compile IF) commands;
Check **Attributes** to display information as a file is initially referenced;
Check **Task Stats** to display process information after a task completes .

See the `PRINT BACK` command. See the `SET` command.

Procedure Settings



These settings effect the behaviour of the commands under the **Procedures** menu. The setting are saved in the `sir.ini` file in the user's *My Documents* directory in Windows, or the .sir file in the user's home directory in Unix.

Exclude Database Variables defines a set of database variables to be excluded from the variable lists in the Procedures.
Sometimes database variables and records (and dummy CASES) are used for admin/audit and other purposes not directly related to the study. You can exclude these variables from the pick lists in the output procedures. Because these are related to a DATABASE rather than a user, these are defined and stored in the database PROCFILE. The member `SYSTEM.EASYPROF` or the member named by the GLOBAL "EASYPROF" is used. If the member does not exist all variables and cases are used.
The format of the file is:

```
CASE LIST = caseid_list
INCLUDE
 recname.varname
 recname.varname
 recname.varname
 ...
EXCLUDE
 recname.varname
 recname.varname
 recname.varname
 ...
```

Recname.varname can be a pattern (like rec@.va@me) in the **EXCLUDE** list. **Extension / View With** Shows the external programs used to view the output of the procedure.

Press **Remove** to delete the association between the filename extension and external program.

New / View With enter a new or existing extension and use the **>>** button to locate the executable file associated with this extension.
Press **Add** to update the list with the new details.

Preview Records set the number of records to be processed in preview mode.

Use Master If you are accessing the database through Master and you uncheck this box then Master is not used.

HTML Style points to a member that is used with HTML reports. There are some members in the sysproc file with names like `EASY.RSTYLEx`. you can enter one of these or create your own member based on one of these and use that.

Server Settings

Set timeouts in tenths of a second for SirMaster and SirSQLserver.

Master Settings

Controls concurrent database access through master. A SirMaster must be running and have write access to a database to use concurrent database access.

Enter the name and port of the Master process and press `Connect`. This name is shown when master is started, e.g.: `SirMaster Started 09/27/05 11:31:18 on '`**`b2:3000`**`'` - the name is **B2:3000**).

Press the `Disconnect` button to access databases in single user mode. Single user mode is much faster than concurrent access but, if master is controlling the database for other users, you are only able to read and the data you retrieve is current to the point master last copied the data to disk.

When Master is connected its name is shown in the SIR/XS main window title bar.

When Master is connected you can use this dialog to control the master process.

If master has been started with a password specification then this password must be entered correctly.

Master can be set to automatically perform a regular `Difference File Copy`. To set the update interval in minutes, enter a number and press the `Reset Copy Interval` button. A value of 0 specifies that the difference file copy is only done when all users stop using the database. This command requires a correct password.

The `Copy Diff. File Now` button does a difference file copy.

A list of clients of the current master is displayed. The current session client is marked with a "*". The connection time and last transaction time of the selected client is displayed below the list. Press the `Refresh` button to update this list.

Press the `Disconnect User` button to disconnect a user from master. This requires a password. Normally allow users to disconnect themselves.

A list of active databases opened by the current master is displayed.

Press the `Shutdown Nice` button to initiate master shutdown. This requires a password. Each master user is sent a message and no new users are allowed to connect to this master. When all users have disconnected then master shuts down.

Press the `Shutdown Now` button to shut master down now. This requires a password. No warnings are given to users.

LIST REMARKS/LIST COMMANDS

These can be turned on or off to limit the amount of information displayed in the output as commands are processed. These are here for convenience. List Remarks, when on, means that various informative messages are displayed. List Commands, when on, echoes all input commands to the output.

Remarks and commands, as well as several other output message types, can be switched on and off in the Preferences dialog. They can also be controlled using the PRINT BACK DBMS commands

SIR/XS HELP

The SIR/XS help is HTML based and can be viewed using any HTML browser. The preferred HTML browser can be chosen in user preferences.

Welcome to SIR/XS

This dialog is displayed when SIR/XS is started without a database parameter and when the novice welcome indicator is on.

Novice Welcome

Press **Cancel** or select "**Start with no database**" to start SIR/XS without connecting to any database.

Select "**Create a new database**" to start defining a new database.

Select "**Open new database**" to connect to selected databases in the list below. You are prompted for any passwords. If More... is selected then the file browser allows you to search for databases files.

Check the "**Don't show this anymore**" to prevent this dialog being displayed every time you start SIR/XS. It can be displayed again by checking "Display Novice Welcome" in the startup section in the Preferences dialog.

Introduction

The dialog and forms painters are visual tools for generating VisualPQL Code. They are written in VisualPQL using the DEDIT set of commands.

DIALOG PAINTER

The Dialog Painter is a visual tool for creating and modifying specifically formatted PQL programs. You can use a text editor to modify a program created by the painter providing that you leave the painter specific commands.

All the dialogs in the SIR/XS system interface have been created and can be modified using the dialog painter.

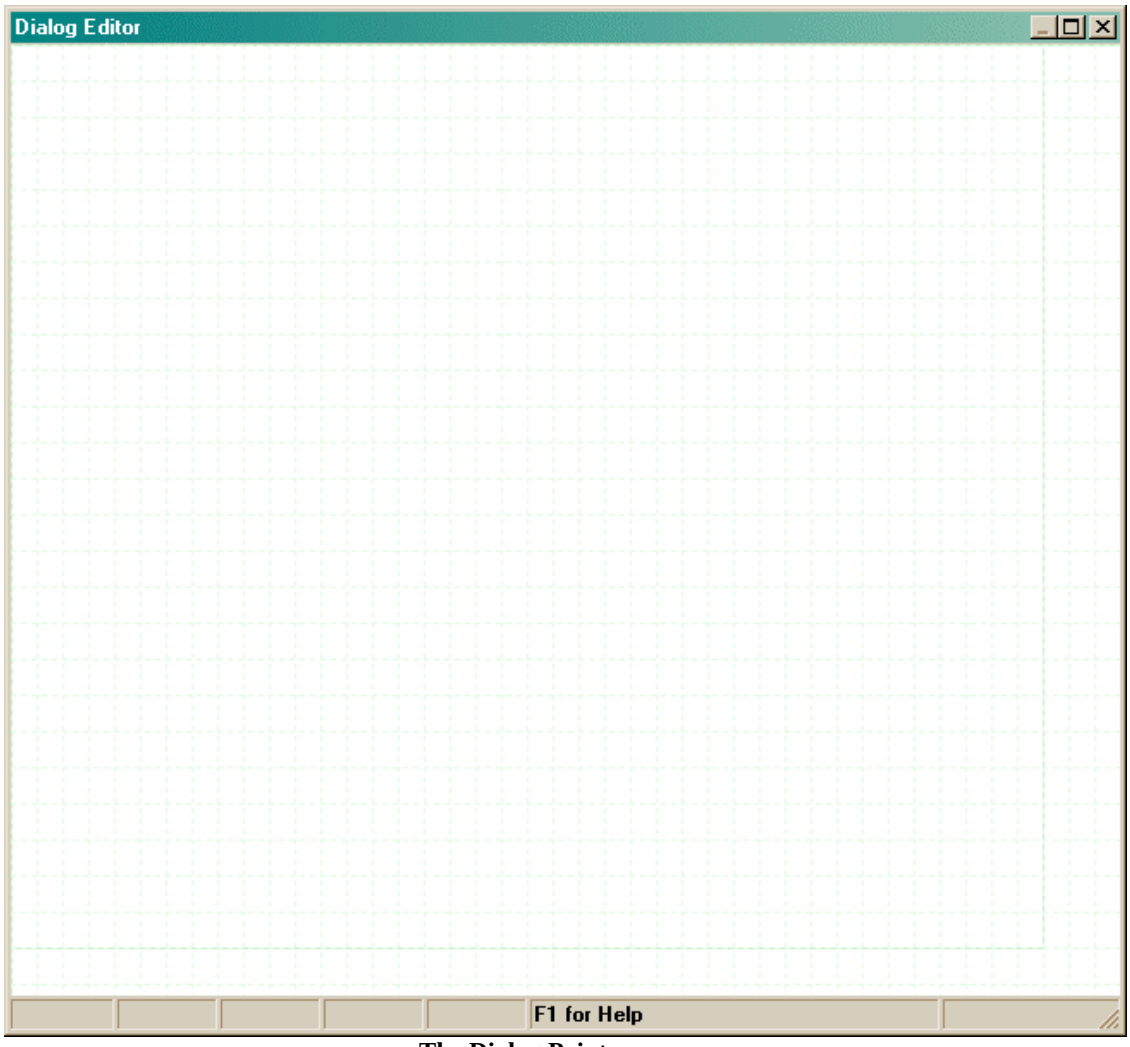
Dialogs generated using the Dialog Commander can be used with the dialog painter not vice versa.

To run the dialog painter from the menu system, select Program, Dialog Painter... and enter a member or filename that either contains a dialog painter program or does not exist. The dialog painter can also be started by editing a member from the members dialog where the member contains "`|| Generated by DP ...`" as the first line of text.

You use the dialog painter to control the layout of the dialog; the type, size and position of the individual controls on the dialog. From the dialog painter you can call other routines that control the rest of the dialog program.

Creating a new Dialog Program

When the dialog painter is started with a new file or member, a blank dialog painter canvas is displayed:



The Dialog Painter canvas

Press F2 or right mouse click to see a menu of commands.

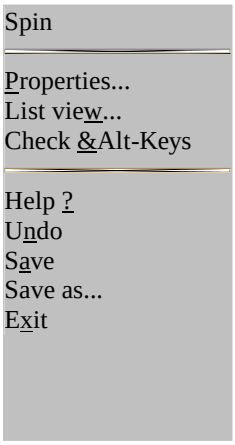
Label
Edit
Button
CheckBox
Radio
Choice
List (M)
Text
Horizontal Line
Vertical Line
Rectangle (S)
Image
ComboBox
Progress Bar
Tree control
Slider

The **New control** (label,Edit,Button etc) commands insert a control of that type at the point where the mouse click occurred or, if the mouse was not used, just below the last control. The control can then be dimensioned as required.

The **Properties** (Enter) command controls the properties of the whole VisualPQL program.

The **List View** command shows the sequence in which dialog controls are processed and allows you to change this.

Check &Keys checks for duplicate alt+letter shortcuts and displays a list of duplicates as well as list of assigned and available letters.



list of duplicates as well as list of assigned and available letters.

Help (F1) displays this help page.

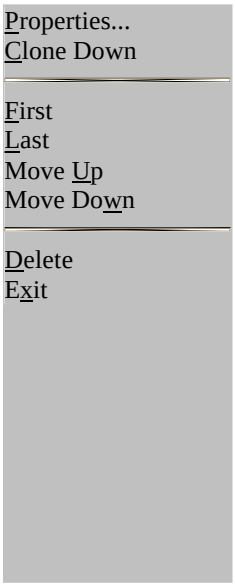
Undo (Ctrl+Z)undoes the last painter action or set of moves or resizes.

Save saves the dialog program to the current file or member.

Save as... prompts for a file and saves the dialog program to that file.

Exit terminates the painter, prompting to save any changes.

Once you have put a control on the canvas, right mouse click over it to see a menu of commands related to this control.



The **Properties** (Enter) command controls the properties of this particular control. This includes the VisualPQL executed when this control is used in the dialog.

Clone Down creates a copy of the control (same size and text) just below the original.

First,**Last**,**Move Up** and **Move Down** change the order in which the dialog items are processed. This is the order of focus when the tab key is used to move around controls in the dialog (or in the painter). Note that Radio buttons are considered to be in the same group if they are defined together with no intervening control.

Delete removes the control from the dialog. It does not remove any associated VisualPQL code that you have entered but this is not executed as long as no other control re-uses the deleted ID.

You can select multiple controls by holding down the left mouse button and dragging to create a rectangle around them, or by holding down CTRL and selecting each of them. A right mouse click on one of the selected items displays a menu of commands related multiple controls.

The **Align Left**, **Right**, **Top** & **Bottom** commands align the selected controls with the control that was right mouse clicked. That control does not move.

The **Align Row Centre** command aligns all selected controls with the control that was right mouse clicked using vertical centre alignment. That control does not move.

That control does not move.

The **Align Column Centre** command aligns all selected controls with the control that was right mouse clicked using horizontal centre alignment. That control does not move.

The **Space evenly** commands keeps the position of the top/right most control and the bottom/left most control and moves the others so that the space between each control is the same.

The **Abut** commands keeps the position of the top/right most control and moves the others so that there is no space between each control.

The **Same Width** & **Height** commands resize all selected controls to be the same as the control that was right mouse clicked. That control does not change.

Clone Down creates a copy of each of the selected controls (same size and text) just below the original.

Delete removes the controls from the dialog.

Adding Controls

Add a new control to the dialog by pressing the right mouse button and selecting the control type from the pull down menu. The new control is inserted where the mouse was when the right mouse button was pressed.

You can also insert new controls using the Insert key and selecting the type or by pressing one of the character keys L(label) E(edit) B(button) K(check) R(radio) C(choice) M(list) T(text) H(horizontal line) V(vertical) S(box) or I(image). If the mouse is not used then the new control is placed just below the previous control.

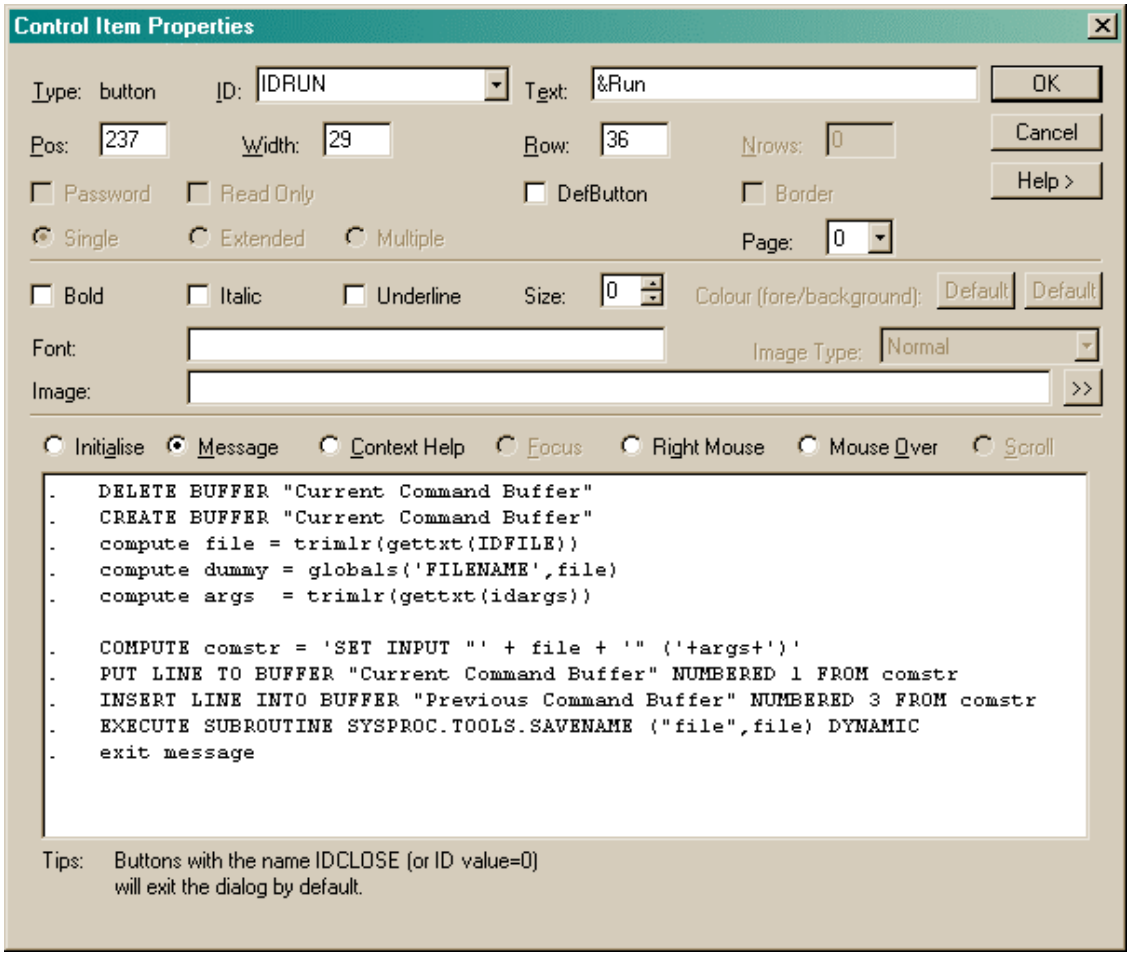
Moving and resizing controls

You can move and resize controls with the mouse or keyboard. Click the mouse anywhere in the control's border to select it. Drag from anywhere inside the border to move the control or drag one of the handles on the edge to resize.
Use the arrow keys to move the selected control up, down left or right. Ctrl+arrow allows fine movements.
Use Shift+arrow keys to resize the control. Right arrow makes the control wider; left arrow, narrower. Ctrl+Shift+arrow allows fine resizing.

If more than one control is selected then all are resized or moved.

Control Properties

Double clicking on a dialog item gives the item's properties.



Properties of a Button

Here you can define the identifying name of the control, any text associated with that control (This does not include text in lists, choices or multi-line text controls). You can also define other control specific attributes in this dialog.

ID Gives the control's name. Internally these identifiers are unique numbers. The painter automatically assigns numbers to the names.
Pos,**Width**,**Row**,**Nrows** Set the control's position and size. Some control types have predefined height.
Password Readonly Set attributes for text controls. Password will hide entered text and read only will prevent text modification.
DefButton Sets this button as the default button on the dialog. There can only be one default button. This will be the action taken when the user presses the enter key (unless a

text control has the current focus).
Border Displays a rectangle around the control.
Single Extended Multiple Sets the type of list selection. **Single** allows exactly one selection. **Extended** is the same as single except you can select more items using shift, ctrl or by dragging the mouse. Multiple allows for zero or more items to be selected.
Page Sets the dialog page where this control will be displayed.
Bold Italic Underline Set these attributes for the control's font.
Size Sets the relative font size. 0 (zero) is the default size, 1 or more is larger, -1 or less is smaller than normal.
Colour Sets the foreground and background colour for the control.
Font Sets the font name for the control's text.
Image Type Can be either normal, centre or resize to position the image within the control area.
Image Sets the bitmap file image for the image or button control. If this is a button then the image ill be used instead of the button text.

In the lower part of the dialog is a text box that contains the VisualPQL commands that are executed when various messages associated with this item are sent from the windows system to the VisualPQL program.

The **Initialise** section contains the VisualPQL that is executed when dialog is initialised. You could also put this code in the dialog properties initial section. Defining it here makes it easier to find the control specific code using the dialog painter.

The **Message** section contains the VisualPQL that is executed when the button is pressed, or a change is made in an edit, choice, check box, list or radio control, or the mouse is clicked in an image.

The **Context Help** section contains the VisualPQL that is executed when the HELP button is pressed or the question icon on the dialog frame is pressed followed by a click on this control. The code typically pops an information box with some kind of help about the control. E.g:

```
DISPLAY INFOBOX "Enter your FULL name"
```

The **Focus** section applies to edit and multi-line text controls. It contains the VisualPQL that is executed when the control is left (ie looses focus). This VisualPQL might be used to validate a string or perform some other action based on the entire string. The ordinary edit messages are sent each time a character is typed.

The **Right Mouse** code is executed when the right mouse button is pressed on a control.

The **Mouse Over** code is executed when the mouse pointer hovers over a control.

The **Scroll** message applies to lists and multiline text controls. The code is executed when the list or text is scrolled and it passes the position of the top visible item or line as

arg1. Use this message along with the `SETPOS` function to implement synchronised list scrolling.

Identifier Names

Controls in the dialog are identified internally by unique numbers. The Identifying names you use in the painter generates VisualPQL integer commands and automatically assigns numeric values to these. There are two predefined names, **`IDCLOSE`** and **`IDSTATIC`** that are always assigned the values 0 (zero) and -1 respectively.

The value -1 need not be unique, many controls in the dialog can have this identifying value but no processing can occur on these controls. They are usually labels or lines.

VisualPQL code is generated to terminate the dialog when a control with an identifying value of zero (IDCLOSE) is used. You can define code that is executed before the dialog terminates. Closing the dialog using the X (or Alt+F4 in Windows) sends a message with ID zero to the VisualPQL program.

Names beginning with "UD" (for user defined) are not automatically preset in the generated program and you must assign values to these yourself. This is useful when you want to have a meaningful assignment of numbers to identifiers (eg a set of radio buttons with values and labels "2006","2007","2008","2009").

You can use numbers (integers greater than -1) as control identifiers in the painter.

All other identifying names are assigned a unique arbitrary positive integer value.

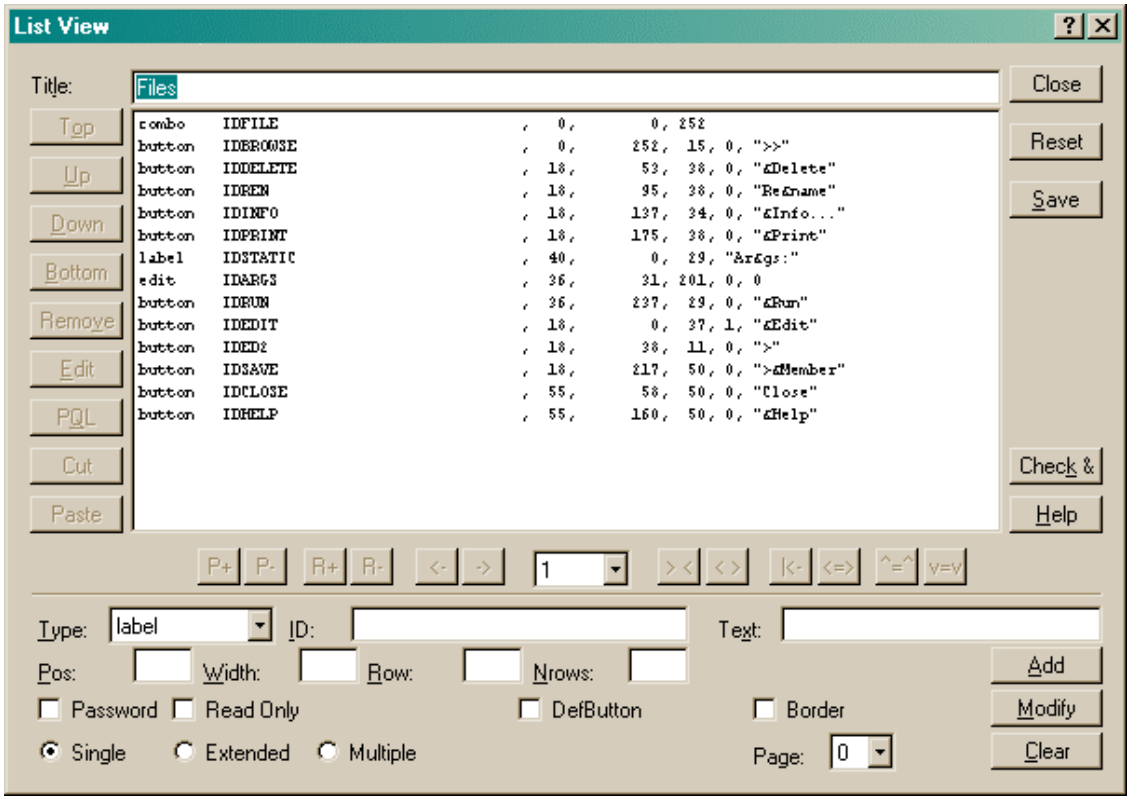
Predefined Variables

The variables **`m_id`** **`m_arg1`** **`m_arg2`** and **`m_page`** are declared and used in the generated VisualPQL program. Do not use these variable names as control identifiers.

`m_id` contains the value of the dialog item that has just been pressed or changed.
`m_arg1` and **`m_arg1`** contain more information on the control.
`m_page` indicates which pane of a multi-paned dialog is displayed.

List View

The List View can be used to do all the dialog layout functions that can be done in the painter. The controls are listed in a menu and these can be selected and modified using the list view buttons.



List View

The dialog is split into two parts with a horizontal line. The top part controls the layout of the whole dialog. The **Dialog title** is the text to display in the title bar of the generated dialog. The main box under the title is a list of the controls currently defined for the dialog. A new control can be added by making selections in the bottom section of the designer and pressing **Append**. There are a number of action buttons down the left and right of the control list as follows:-

- **Up** Moves the selected control up in the list of controls. The order in the list defines the order that focus is set when the user 'tabs' around the dialogs fields. This does not effect the layout of the dialog.
- **Down** Moves the selected control down in the list of controls.
- **Remove** Removes the selected control(s) from the dialog
- **Edit** Copies the settings for the selected control to the bottom section of the Dialog Designer for editing.

- **PQL** Displays the selected control's properties.
- **Cut** Removes selected controls to the clipboard.
- **Paste** Inserts previously cut controls into the current selected position.
- **Close** Exits the List View and returns to the painter after prompting to save any changes.
- **Reset** Removes all controls from the dialog.
- **Save** Saves changes made to the dialog layout.
- **Check &** checks for duplicate alt-letter assignments. It displays a list of the duplicates as well as lists of used and available letters.

There are a number of action buttons underneath the control list as follows:-

- **P+** Moves the selected control(s) to the next page in a multi-paged dialog.
- **P-** Moves the selected control(s) to the previous page in a multi-paged dialog.
- **R+** Moves the selected control(s) up one row in the dialog layout.
- **R-** Moves the selected control(s) down one row in the dialog layout.
- << Moves the selected control(s) left by the number of columns shown in the choice field.
- >> Moves the selected control(s) right by the number of columns shown in the choice field.
- >< Shortens the selected control(s) by the number of columns shown in the choice field.
- <> Lengthens the selected control(s) by the number of columns shown in the choice field.
- |<- Aligns all selected controls to the left most column of the selected controls.
- **[=]** Makes all selected controls the same width as the widest of the selected controls.
- ^=^ Aligns all selected controls to the top of the topmost control.
- **v=v** Aligns all selected controls to the bottom of the bottom most control.

The bottom section is for working on a single control within the dialog. Once changes have been made in this section, press Modify or Append to apply the changes to the dialog.

- **Type** selects the type of dialog control. The options are:
  - Label - a static or programmer changeable labels;
  - Edit - a single line text entry box;
  - Button - a button;
  - Check - an on/off button;
  - Radio - one of a set of on/off buttons of which only one may be on;
  - Choice - a pull down list;
  - List - a scrollable list;
  - Text - a multi-line text entry box;
  - Line - a drawn line or box;

  - Image - a picture;
  - Combo - a combination of edit and choice;
  - Progress - a progress bar;
  - Tree - a tree list;
  - Slider - a dragable slider;
  - Spin - a numeric edit with up/down arrows.
- **Id** Each item must have an identifier associated with it. This identifier is an integer and if it is to be referenced elsewhere in the VisualPQL code then the number must be unique to this dialog. When using the Dialog Commander, you can use two predefined names with values viz. `IDCLOSE` value 0 (zero) and `IDSTATIC` value -1. These names can be used many times in a dialog but cannot be reference by the VisualPQL code (ie you cannot modify or read values during execution).

  Any name starting with uppercase `ID` is automatically assigned a unique numeric value. If the id does not start with `ID` then you must define and set it to a unique number in the PROLOGUE section.

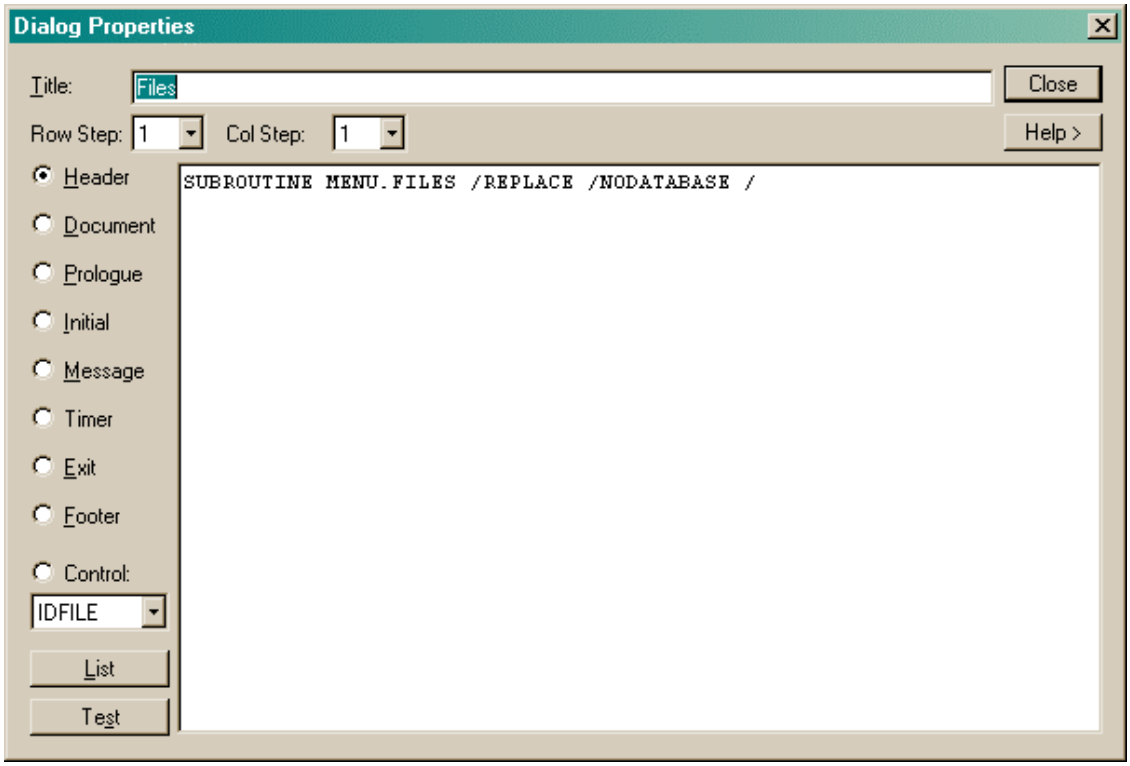
- **Text** is the label on Buttons, Radio Buttons, Check Boxes, and plain Labels. This is disabled for other control types.
- **Row** is the vertical position of the control in the dialog. 0 (zero) is the top row.
- **Nrows** in the height in rows for Lists, multiline Text, Lines and Images. This is disabled for other control types.
- **Pos** is the horizontal position of the control in the dialog. 0 (zero) is the leftmost column. There are approximately 4 units to a character.
- **Width** is the width of the control in horizontal units.
- **Password** can be checked for Edit controls. Text entered in password edit areas can not be read. This is disabled for other control types.
- **ReadOnly** can be checked for Edit and multiline Text controls. Text displayed in these areas cannot be modified. This is disabled for other control types.
- **DefButton** can be checked for one Button control. This button is the Dialog's default button. This is disabled for other control types.
- **Border** draws a box around Image controls. This is disabled for other control types.
- **Single / Extended / Multiple** selects the behaviour of List controls. Single means one and only one selection can be made. Extended means one or more selections and multiple means zero or more. Multiple selections are made by clicking on the menu item and unselected by clicking on a selected item.

There are three buttons in the lower section:

- **Append** Appends the control defined in this section to the end of the list of controls.
- **Modify** Replaces the selected item in the list of controls with the control defined in this section.
- **Clear** Clears the details in this section.

Dialog Properties

The Dialog Properties controls all user sections of the VisualPQL program.



The Dialog Properties

On selecting dialog properties a dialog appears as above.
The text in the **Title** field at the top is displayed in the title bar of the generated dialog.

Below are some settings specific to the painter. The Row and Col step set a grid size for use in the painter. You can also set the dialog page to display if there is more than one page defined.

The left side of the Dialog Commander is a list of the sections in this program. The default program sections are **Header**, **Document**, **Prologue**, **Initial**, **Message**, **Timer**, **Exit**, **Footer** and specific **Control:**. the right side of the dialog is the VisualPQL code associated with the current section. You can modify this code.

Header
The text associated with **Header** typically includes the command to identify the beginning of a VisualPQL routine i.e. `PROGRAM` or `RETRIEVAL` or `SUBROUTINE`. If it does not then the generated code must be included in a program and cannot be run stand-alone.

Document
This text will become a comment at the start of the generated program.

Prologue
The **Prologue** contains code to be executed before any other executable commands.

Initial
The **Initial** section contains the code to initial the controls in the dialog.

Message
The **Message** section contains the code executed each time a dialog event message is issued, irrespective of which control is involved.

Timer
The **Timer** section contains the code executed when a timer message is sent. The timer is set with the `ENABLE TIMER n` command. This command makes the timer message block execute every $^{n}/_{10}$ seconds. The `DISABLE TIMER` command stops timer messages.

Exit
The **Exit** section contains the code executed after the dialog is removed from the screen.

Footer
The **Footer** should include the corresponding `END` command from the **Header**. e.g. `END PROGRAM`. The Footer can also be used for user subprocedures.

Control:
The **Control:** section contains the VisualPQL message block for the control selected in the choice. This is the same code as the message block in the control properties dialog.

Dialog Painter Format

The files created by the dialog painter follow this format:

```
|| Generated by DP - don't edit anything outside |{...|}

|{ Header
```

Header section VisualPQL

```
|}

integer*4 m_id, m_arg1, m_arg2

integer*4 IDSTATIC; preset IDSTATIC (-1)
integer*4 IDCLOSE ; preset IDCLOSE  ( 0)
integer*4 ....... ; preset .......  ( 1)
integer*4 ....... ; preset .......  ( 2)
...

|{ Prologue
```

Prologue section VisualPQL

```
|}

dialog "...title"

|{ Controls
```

Control layout definition

```
|}

initial
|{ Init
```

Initial section VisualPQL

```
|}
end initial

message ALL m_id, m_arg1, m_arg2

|{ Message
```

general Message section VisualPQL

```
|}

ifthen (m_id eq CONTROL_ID )
|< CONTROL_ID
```

Control message section VisualPQL

```
|>
next message
endif

if (m_id eq 0) exit message

end message

|{ Context Help
message HELP m_id

ifthen (m_id eq CONTROL )
|< CONTROL_ID
```

Control context help section VisualPQL

```
|>
next message
endif


end message
|}
message FOCUS m_id
|{ Focus

ifthen (m_id eq CONTROL )
|< CONTROL_ID
```

Text Focus-out message section VisualPQL

```
|>
next message
endif


|}
end message
message TIMER
|{ Timer
```

Timer message section VisualPQL

```
|}
end message

end dialog
|{ Exit
```

Exit section VisualPQL

```
|}
|{ Footer
```

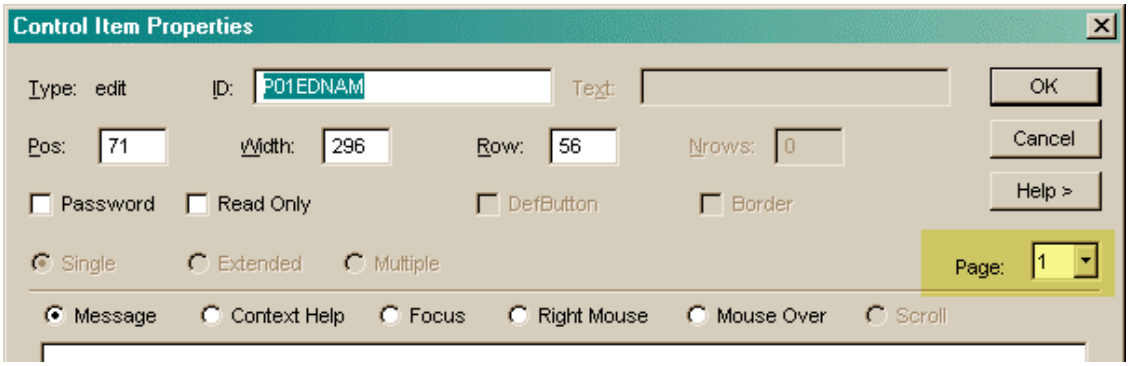
Footer section VisualPQL

```
|}
```

Multipage Dialogs

Dialogs can have multiple pages. Each item on the dialog has a page number associated with it. The default page number is **zero** and controls with page zero appear on every page.

You can change a control's page number in the control item properties screen.

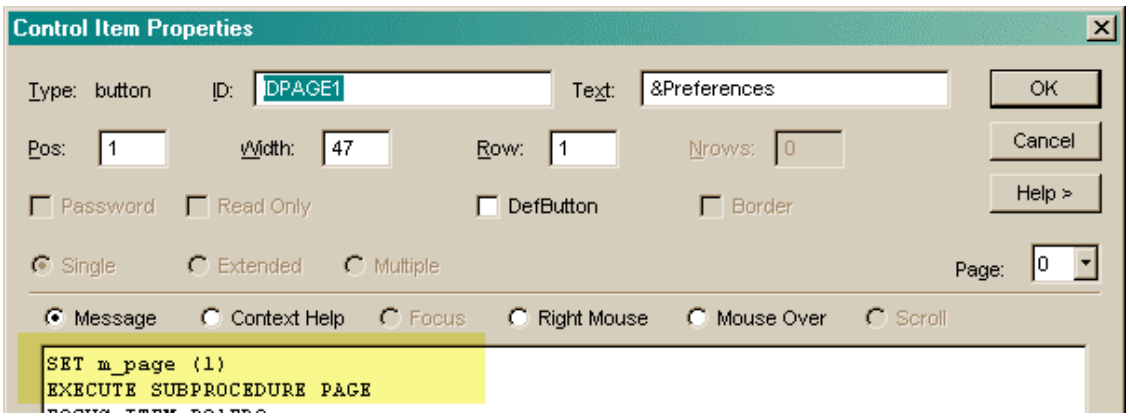


Page set to one

Change the page you work on in the painter in dialog properties or by pressing the PgUp, PgDn, Home or End keys.

To make a dialog with multiple pages using the painter you need some switching method. If this is under user control this might be a choice control or a set of buttons. You can also use paging to show or hide controls under other conditions.

The switch must change the value of the variable `m_page` and then `EXECUTE SUBPROCEDURE PAGE`.



Page switched by a radio button

A small paged dialog example follows with user VisualPQL in red:

```
|| Generated by DP - don't edit anything outside |{...|}

|{ Header
PROGRAM
|}

integer*4 m_id, m_arg1, m_arg2

integer*4 IDSTATIC; preset IDSTATIC (-1)
integer*4 IDCLOSE ; preset IDCLOSE  ( 0)
integer*4 IDBUTT1 ; preset IDBUTT1  (1 )
integer*4 IDEDIT2 ; preset IDEDIT2  (2 )
integer*4 IDPAGE1 ; preset IDPAGE1  (3 )
integer*4 IDPAGE2 ; preset IDPAGE2  (4 )
integer*4 m_page; preset m_page (1)

|{ Prologue
|}

dialog ""

|{ Controls
postype 1
button    IDBUTT1 ,  14,        0, 120, 0, "Press ME"
edit    IDEDIT2 ,  14,        0, 117, 0, 0
radio   IDPAGE1 ,   0,        2,  40, "Page 1"
radio   IDPAGE2 ,   0,       52,  40, "Page 2"
|}

initial
|{ Init
|}
. EXECUTE SUBPROCEDURE PAGE
end initial

message ALL m_id, m_arg1, m_arg2


ifthen (m_id eq IDBUTT1 )
|< IDBUTT1
display infobox "Thank you"
|>
next message
endif

ifthen (m_id eq IDPAGE1 )
|< IDPAGE1
SET M_PAGE (1)
EXECUTE SUBPROCEDURE PAGE
|>
next message
endif
```

```
ifthen (m_id eq IDPAGE2 )
|< IDPAGE2
SET M_PAGE (2)
EXECUTE SUBPROCEDURE PAGE
|>
next message
endif

if (m_id eq 0) exit message

end message


end dialog
|{ Exit
|}
|{ Pages
SUBPROCEDURE PAGE
.  HIDE ITEM IDBUTT1
.  HIDE ITEM IDEDIT2
.  IFTHEN (m_page EQ 1  )
.    SHOW ITEM IDBUTT1
.  ENDIF
.  IFTHEN (m_page EQ 2  )
.    SHOW ITEM IDEDIT2
.  ENDIF
END SUBPROCEDURE
|}

|{ Footer
END PROGRAM
|}
```

PQLFORMS PAINTER

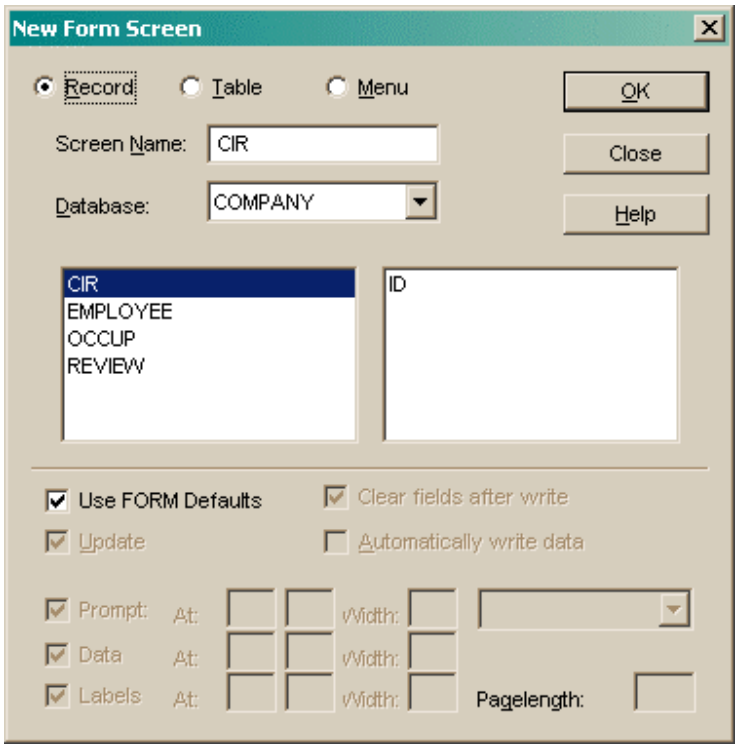
The PQLForms Painter is a tool for creating and modifying PQLForms programs, in particular the layout of screens.

Run the painter from the menu system by selecting Program, PQLForms Painter... and entering a member or filename that either contains a PQLForms program or does not exist.

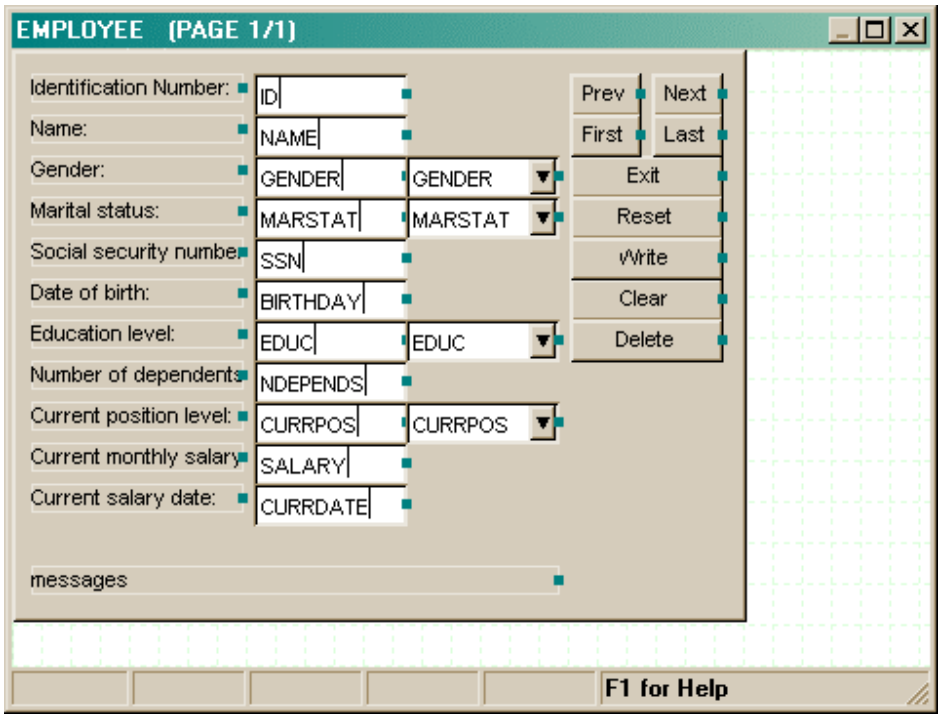
The PQLForms painter allows you to control the layout of a form screen; the type, size and position of the individual controls on the form.

Creating a PQLForm

There are several ways to create a new PQLForm.
You can generate default code using the `WRITE SCHEMA` command with the `PQLFORMS` option.
You can hand write PQLForms code. The forms painter does not require a formatted input file.
You can start the PQLForms painter with a new file/member and on the Select Screen Dialog, choose New.

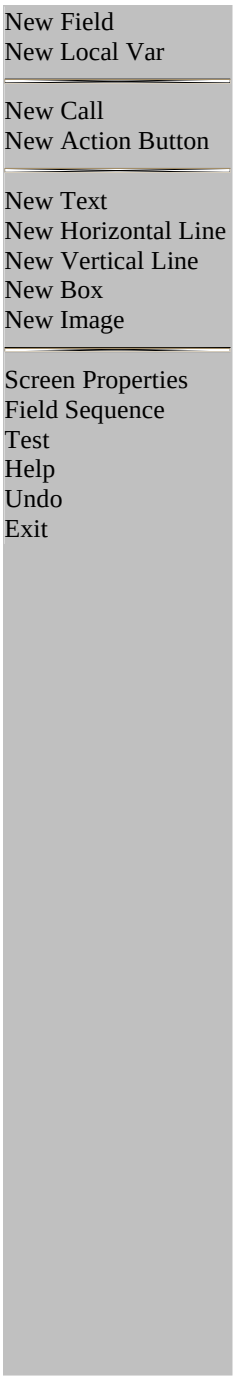


New Database Record Screen



Record Screen Painter

Press F2 or right mouse click to see a menu of commands.



The **New Field** command inserts a field from the current record or table.

The **New Local Var** command inserts a field based on a derived variable.

The **New Call** command inserts a CALL button to access another PQLForms screen.

The **New Action Button** command inserts an Action Button. A list of Action button types is given. Action buttons can be used to navigate through the data in to form or to execute some user defined VisualPQL.

The **New Horizontal Line, Vertical Line and Box** commands draw lines on the screen.

The **New Image** command defines a bitmap image area for the screen.

The **Screen Properties** (Enter) command controls the properties of the whole PQLForms screen. It also lets you set which page of the VisualPQL screen is visible on the painter canvas.

The **Field Sequence** command lets you change the order in which the fields are processed (using the tab key). This does not change the visual layout. It also lets you define pages of the screen and move fields from one page to another.

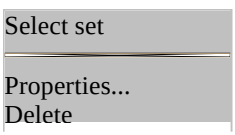
Test (F5) executes the VisualPQLform layout commands (not the processing commands) so you can see how the final layout looks.

H**e**lp (F1) displays this help page.

Undo (Ctrl+Z)undoes the last painter action or set of moves or resizes.

Exit terminates the painter, prompting to save any changes.

Once you have put a field or button on the canvas, right mouse click over it to see a menu of commands related to this control.



Select Set selects any Prompt, Data and Label for the field, allowing it to be moved as a group.

The **Properties** (Enter) command controls the properties of this particular field or button. This includes the VisualPQL executed after the field is entered.

Delete removes the field or button from the dialog.

When you have more than one field on the canvas, you can select more that one control (by dragging a rectangle around them with the mouse, or CTRL and select), right mouse click on one of the selected items to see a menu of commands related multiple fields.

The **Align Left**, **Right**, **Top** & **Bottom** commands move all selected controls into alignment with the control that was right mouse clicked. That control does not move.

The **Align Row** command move all selected controls into vertical centre alignment with the control that was right mouse clicked. That control does not move.

The **Same Width** & **Height** commands resize all selected controls to be the same as the control that was right mouse clicked. That control does not change.

The **Properties...** option will display a multiple field properties dialog which will allow changing of attributes across all selected fields.

Delete removes the fields from the dialog.

Moving and resizing

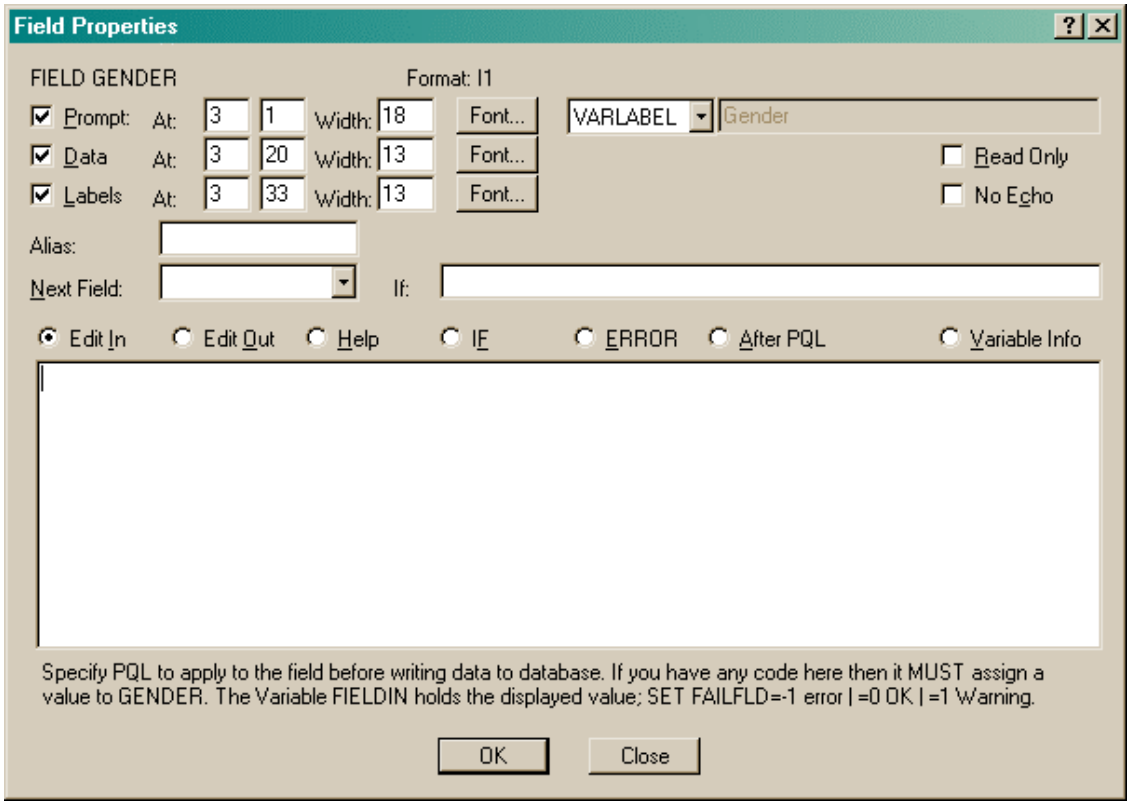
Controls can be moved and resized with the mouse or keyboard. Click the mouse anywhere in the control's border to select it. Drag from anywhere inside the border to move the control or drag one of the handles on the edge to resize.
Use the arrow keys to move the selected control up, down left or right.
Use Shift+arrow keys to resize the control. Right arrow makes the control wider; left arrow, narrower.

PQLforms coordinates are based on text rows and columns. Fine movements and resizing cannot be made.

If more than one control is selected then all are resized or moved.

Field Properties

Double clicking on a part of a field (prompt data or label) gives the field's properties.



Properties of a Field

Here you can set field level options such as [NO]DATA [NO]LABEL [NO]PROMPT [NO]ECHO and READONLY as well as VisualPQL code to executed after the field is entered.

The Position and size of the prompt, data, and labels can be manually altered here or in the painter canvas. You can set various font attributes for prompt, data and label using the corresponding **Font...** button.

Check the **Read Only** box to make the field not updatable by the forms user.

Check the **No Echo** box to make the field secret. When data are entered into these fields they are displayed as asterisks.

The **Alias** field can be used to enter the variable's name if this field's name is not a record variable. Use this if you have several variables in the database with the same name but

different data types. You would give each of these different field names with aliases to the real variable name.

Select a **Next Field** which the keyboard focus will move to after this field. Optionally enter an **If** condition for the next field processing.

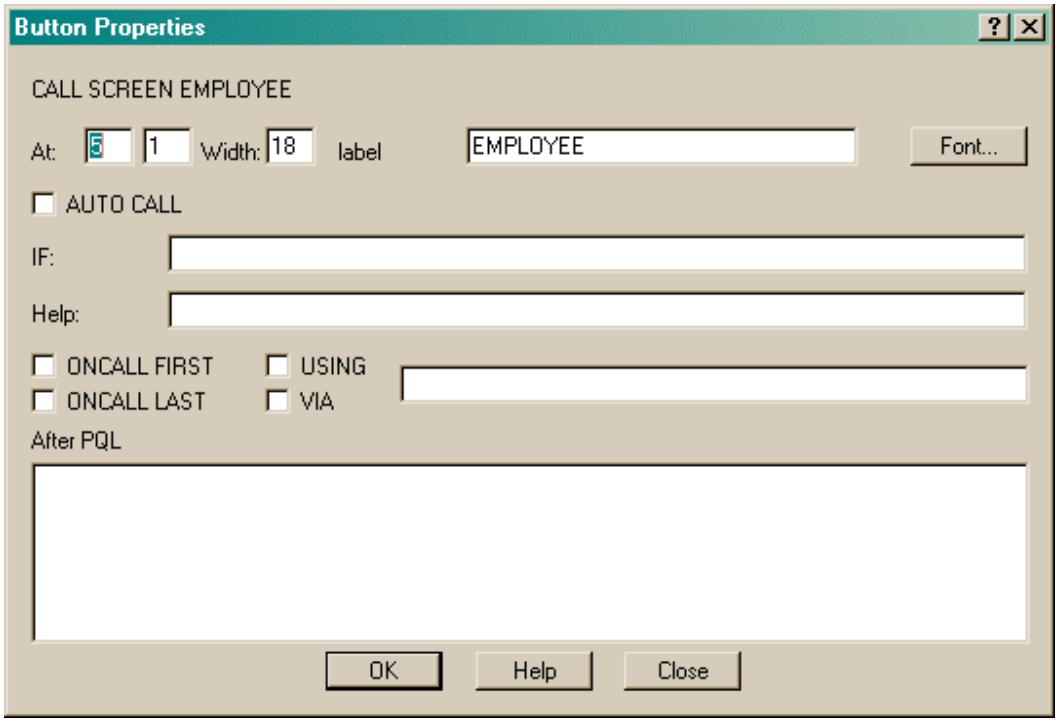
The radio buttons correspond to various field clauses. See the PQLForms FIELD page for full details.:

- **Editin** - Specify PQL code that takes the value from the predefined string variable FIELDIN and sets the value of the record, row or local variable. Separate multiple commands with a semi-colon ';'. Example: `COMPUTE ssn = REPLACE (fieldin,'-','',LEN(fieldin),1,0)`
- **Editout** - Specify PQL code that alters the way the field is displayed. The specified commands are typically a function or set of functions that use the field as the input and create the predefined string output field FIELDOUT. The form then uses this as the field displayed in the screen. Separate multiple commands with a semi-colon ';'. Example: `compute fieldout = edit (ssn,"^^^-^^-^^^^")`
- **Help** - Specify a string expression which is displayed when the user requests context sensitive help on the field.
- **IF** - Specify a condition which must be true to allow field modification. If the condition is false then the field will be disabled.
- **Error** - Enter error number and message pairs to change the default error messages for the field. Example:
- ```
    47     'Not a valid value'
  57     'Failed Edit tests'
  ```

- **After PQL** - specifies PQL code executed after the field has been processed. This can be multi-lined code and there is no need to end lines with ;.
- *Variable Info* shows some read only information about this variable from the schema. This is not part of the form code but is to help the form designer.
  ```

Button Properties

Double clicking on a CALL or Action button gives the button's properties.



Properties of a CALL Button

If the button is a CALL SCREEN button then the following properties can be set:

The Position and size of the button can be manually altered here or in the painter canvas. You can set various font attributes for the button text using the **Font...** button.

Check the **AUTO CALL** box to make the button hidden and automatically call the screen after processing the previous field (an AUTO CALL button cannot be the first control in the screen). The button will still be visible in the painter but not when the form is run.

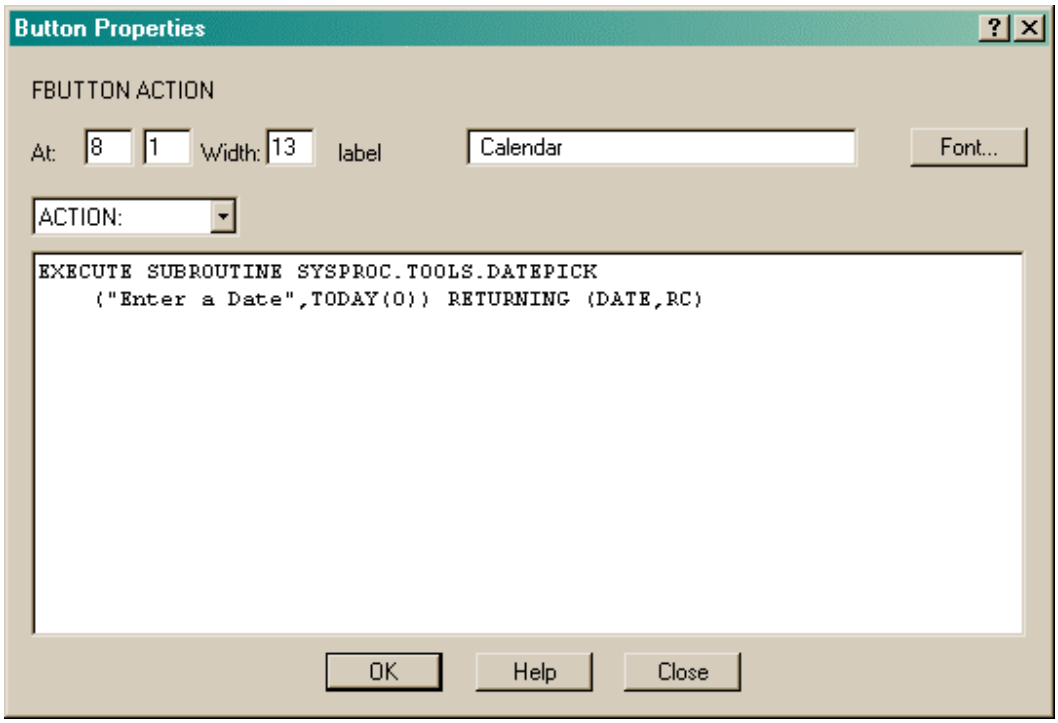
Enter an **IF** condition which must be true to allow the button to be pressed.

Specify a **HELP** string expression which is displayed when the user requests context sensitive help on the button.

Check **ONCALL LAST** or **ONCALL FIRST** to automatically go to the first or last record when the screen is called.

Check **USING** or **VIA** to specify the keys to use when calling the record. USING needs a case id and keys whereas VIA just needs the record keys.

After PQL specifies PQL code executed after the button has been presses. This can be multi-lined code and there is no need to end lines with ;.

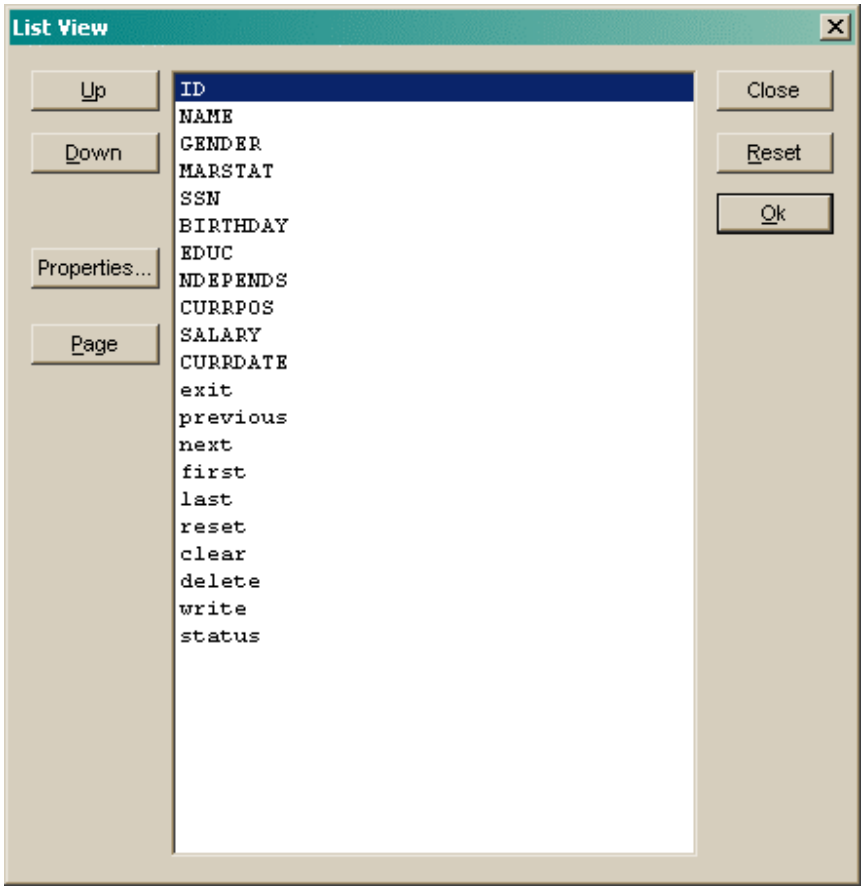


Properties of an ACTION Button

If the button is an ACTION button you can select one of the predefined actions or select ACTION: and enter your own PQL code. You need to separate line with a semicolon - ";".

Field Sequence

Field Sequence can be used to do change the processing order of fields, to add, move or remove paging and to move a field from one page to another.



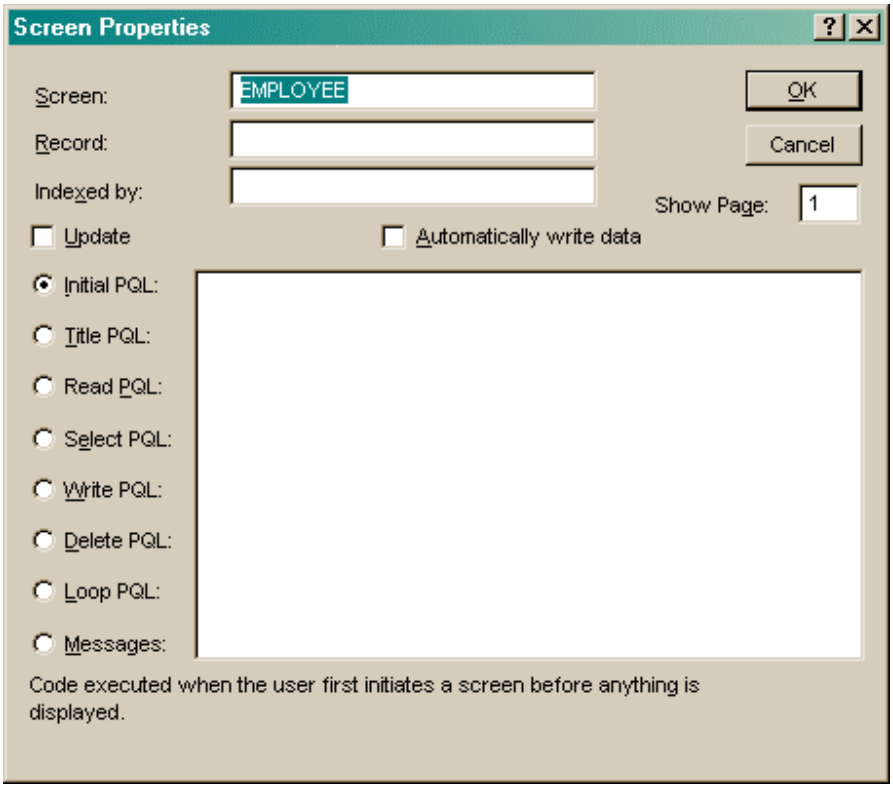
Field Sequence

The list shows variables in processing order. Select a variable and use the buttons on the left to move the field up or down in the processing order. Press the Page button to insert a page break between fields.

The field sequence does not effect the position of the fields on the screen.

Screen Properties

The Screen Properties controls the SCREEN command level options.



The SCREEN Properties

When you select screen properties, a dialog appears as above.
Use this dialog to set the active page to view with the painter. (You can also change the active page from the painter by pressing the PgUp, PgDn, Home or End keys.)

The **Screen** name is specific to this PQLForms program and is used to reference the screen in `CALL`s from other places in the program.

The **record** name defines which database record or tabfile table the screen accesses.

The **Index** name defines which database or tabfile index is used to process the data.

You can enter VisualPQL code to execute at various stages of data entry.
Initial Code is executed when the screen is first called.
Title Code is executed when each page is first displayed. Use this to set the dialog title.
Select Code is executed after a data record is read from the database. Set FAILFLD to non-zero to skip this record and read the next.
Read is executed after a data record is read from the database.

Write before the form data is written to the record.
Loop is executed every time a field is processed.

Form Properties

Individual screens are modified using the painter. The whole PQLForms program can contain many screens. Use the **Edit Header** button on the main screen selection dialog to control the options that apply to the whole PQLForms program.

Commands

Virtually all SIR/XS features can be run either from menus or from commands. A single command can be entered through the `DBMS Command` menu item but normally commands are run as scripts or *procedures*. There are a specific set of commands that can be used when creating procedures. These commands can be anywhere in the input set and can be used to:

- Include comments and remarks.
- Modify the input text. As input is read, it can be modified by:
  - Global variable substitution.
    Global variables are named variables that have a value, primarily used for text substitution. If input text includes the name of the global variable enclosed in angle brackets < >, the current value of the global variable is substituted into the text. Globals can be defined and assigned values with the `GLOBAL` and `GCOMPUTE` commands. There is also a set of predefined system globals that can be used for such things as today's date.
  - Parameter substitution.
    Parameters can be specified when initiating a command stream and these can be used by the commands to set particular values.
  - Text inclusion.
    The `INCLUDE` commands include text from a buffer, file or member at the point of the command. Text inclusion commands can appear within any input source including included text. The `FINISH` and `PEND` commands terminates processing of the current input source. `PEND` returns to the next higher level input source to resume execution. `FINISH` returns to the menus or ends a batch run.
  - Text generation and repetition.
    Repetitive code can be generated with the `DO REPEAT` command. This takes a block of text and expands it. When the entire block has been expanded, the command processor begins processing the commands in the expanded block.
- Control command processing.
  Commands can be processed conditionally either through the block structured `CIF` (for **C**ompile **IF**) command or by the logical tests that can skip to specified places in an input procedure. These control which commands are processed.

  `CIF` sets conditions that either initiate or suspend processing of subsequent commands. When processing is suspended, commands are skipped until another `CIF` command halts skipping. The `CIF TRUE & FALSE` and `CIF END` subcommands allow conditional processing within a `CIF`.

- Control output. Commands that control output and other settings include:
  - `EJECT`

- ERROR LIMIT
- PRINT BACK
- SPACE
- STRING LENGTH
- WARNING LIMIT

Comments and Remarks

Any text after a vertical bar (from that point to the physical end of the input line) is treated as comments. Command continuation lines are not affected by comments after a vertical bar. i.e. as long as the comment is at the end of the line, the command can be continued on a subsequent line.

The comment command can also be used.

```
COMMENT text
```

Inserts comment text. This is printed in the output listing. This is a command and therefore cannot appear between continuation lines for another command. The comment can have continuation lines.

```
REMARK 'text'
```

Writes a remark to the output file.

The printing of remarks and comments is controlled by `PRINT BACK` settings.

Global Variables

Global variables are named variables that have a value. These variables are global in the sense that they are stored in the environment externally to any specific routine. Global variables are primarily used for text substitution but also provide an easy and convenient method of passing small amounts of information between menus and programs.

Global variables have standard SIR/XS names.

Globals can be defined and assigned values in four ways:

- with the `GLOBAL` command
- with the `GCOMPUTE` command
- with the `Globals` option on the `Settings` Menu.
- In VisualPQL programs with the `GLOBALN` and `GLOBALS` functions

If input text includes the name of the global variable enclosed in angle brackets < >, the current value of the global variable is substituted into the text. For example, suppose a global variable called `RECNAME` has the value `PARTS` assigned to it. If the following line is input:

```
PROCESS REC <RECNAME>
```

The value of the global variable `RECNAME` is substituted into this line before the VisualPQL compiler deals with it. The line that the compiler works with is:

```
PROCESS REC PARTS
```

Whenever a global variable is found in the input text enclosed in angle brackets, the current value of the variable is substituted in the text. This substitution happens at input time. If the program is compiled at one time and then run at some later time, the value of the global variable at run time is irrelevant.

Because the system employs a "read ahead" strategy to discover whether the next line is a continuation of the previous line, the value of the global variable is NOT available on the first line after the command that creates (or updates) the variable. For example:

```
PROGRAM
GLOBAL GREET = Hello World
```

```
WRITE <GREET>
END PROGRAM
```

This example does not write "Hello World" because the global variable was not set when the line `WRITE <GREET>` was read. Use the following:

```
GLOBAL GREET = Hello World
C    blank line for global setting
WRITE <GREET>
```

This example writes the greeting. Place a blank comment line after the global assignment to use the value immediately in the next subsequent command.

Global variables at run time

The value in a global variable can be tested in a program at run time by referencing it in a function. There are two VisualPQL functions `NGLOBAL` and `SGLOBAL` that read the run time value of a global variable; the `GLOBALN` and `GLOBALS` functions update the current value of a global variable. The S and N prefix/suffix refer to string variables or numeric variables. These functions happen at run time.

These functions can be used to pass parameters between two compiled programs; between a compiled program and a text procedure; between a text procedure and a compiled program; etc. The value can be used outside a program at compile time for straightforward text substitution and can be referenced in `CIF` and `DO REPEAT` commands.

GLOBAL

```
GLOBAL global_name = value [, global_name = value] ...
```

The GLOBAL command creates global variables and assigns values. Multiple global variables can be defined on a single GLOBAL command, in which case use a comma as the delimiter between global variable specifications.

The value of a global is held internally as text. By default, leading and trailing blanks are deleted. Leading and trailing blanks as well as commas can be specified by delimiting the value with the dollar sign ($), single or double quotes or open/close parentheses () and these delimiters are stripped out of the text. For example:

```
GLOBAL DATEGLOB = $   Jan 24, 2006$
```

If you need to include a delimiting quote or dollar as part of the parameter, specify the character twice e.g.

MONEY is A$41,500:

```
GLOBAL MONEY = $A$$41,500$
```

alternatively use:

```
GLOBAL MONEY = 'A$41,500'
```

GCOMPUTE

```
GCOMPUTE global_varname = expression
```

Initialises a new global variable if it does not already exist or recalculates the value of the variable if it already exists.

The expression may contain:

- string constants (quoted strings)
- string operators (`+`)
- numeric constants
- numeric operators (`+ , - , / , * , **`)
- parentheses
- global variable names

Note: `GCOMPUTE` differs from `GLOBAL` in that it evaluates the expression and may substitute values for globals in the expression. The `GLOBAL` command treats the value as literal text and does no evaluation.

System Globals

As well as global variables defined by the user there is also a set of system globals. The system global variables are:

`CBLEVEL`
: The count of the number of times that the data dictionary of the current database has been updated.

`CPTIME`
: The elapsed time in milliseconds since the beginning of the session.

`DATE`
: The current date in the format 'Mmm DD, YYYY'.

`DBNAME`
: The name of the current database.

`EXMEMBER`
: The name of the member executed on startup or database connection.

`GERROR`
: The total number of errors in commands during the current session.

`GWARNING`
: The total number of warnings issued during the current session.

`LINECNT`
: The lines remaining on the current page on the output file.

`LINEWID`
: The width setting of the output file.

`MASTER`
: The name of the MASTER process controlling this session. If this is not a concurrent session, the value of this global is eight blanks.

`PAGESIZE`
: The current setting of the number of lines per page for the current output device or file.

`SIRCODE`
: The SIR/XS license code.

`SIRCUST`
: The SIR/XS customer number.

`SIRID`
: The SIR/XS release number.

`SIRVER`
: The SIR/XS release version number.

`SIREXP`
: The SIR/XS license expiry date.

`SYSPROC`
: The attribute that points to the system procedure file.

`TERROR`
: The count of task errors; the number of errors detected in the current run.

`TIME`
: The current time in the format 'HH:MM:SS'.

`TWARNING`

The count of task warnings; the number of warnings issued in the current run.

Parameter Substitution

Parameter substitution is similar to global variable text substitution but works with positional specifications rather than named variables.

Specify the parameter to be substituted in the text as a number inside angle brackets, such as <1> or <2>. When the set of text is RUN, supply the parameters to substitute as a list of values. The numbers specified on the text refer to the position of the parameter on the input list. Thus <1> refers to the first parameter, <2> the second and so on. The same number can be used multiple times in the input text if the same value is to be substituted. Separate each entry in the list with commas and enclose the whole list in parentheses. For example, suppose the following program:

```
RETRIEVAL
.  PROCESS REC <1>
.  WRITE <2>
.  END REC
END RETRIEVAL
```

This might be run with a parameter list such as:

```
(PARTS , PARTNUM PARTNAME PARTCOST)
```

The two parameters are separated by the comma. After text substitution has been performed, the program is:

```
RETRIEVAL
.  PROCESS REC PARTS
.  WRITE PARTNUM PARTNAME PARTCOST
.  END REC
END RETRIEVAL
```

Run time parameters can be specified:

- On the `RUN` command
- At the `Args` string when using the `Run` option from the member dialog.
- On `INCLUDE` commands.

Just as with global variables, text parameter substitution happens at compile time. The `NARG` and `SARG` functions return the numeric or string parameters at run time.

INCLUDE BUFFER | FILE | MEMBER

```
INCLUDE BUFFER "buffer_name"          [ (parameter_list) ]
INCLUDE FILE { filename | attribute } [ (parameter_list) ]
INCLUDE MEMBER member                 [ (parameter_list) ]
```

Synonyms:

```
SET BUFFER "buffer_name"          [ (parameter_list) ]
SET FILE { filename | attribute } [ (parameter_list) ]
CALL member                       [ (parameter_list) ]
```

The `INCLUDE` commands include text from a buffer, file or member at the point of the command. Text inclusion commands can appear within any input source including included text.

Positional parameters specified on the `INCLUDE` applies only to the commands directly included by that command. If there are further text inclusion commands within an included set of text, then parameter numbers are reset. When a level is returned to from a deeper level, the calling parameters are once again in effect. The `INCLUDE BUFFER` command includes the text of the specified edit buffer at the point of the command. `SET BUFFER` is a synonym for `INCLUDE BUFFER`. Buffer names obey normal SIR/XS rules.

The `INCLUDE FILE` command includes the text of the specified file at the point of the command. `SET INPUT` is a synonym for `INCLUDE FILE`.

Specify a full filename or attribute.

The `INCLUDE MEMBER` command includes the text of the specified file at the point of the command. `CALL` is a synonym for `INCLUDE MEMBER`. Specify the member to include. Only text members (with the :T suffix) can be referenced. Use `INCLUDE MEMBER *` in batch runs or at the front of a command set, to pass parameters to the commands that follow.

Specify a parameter list where required. Enclose the whole list in parentheses, separating individual parameters with commas. Note that blanks do not separate parameters. The parameters can be delimited if required. The delimiters are quotes (single/double), open/close parens () and dollar signs. Note that a start/end dollar or open/close parentheses are stripped out while the quotes are passed down as part of the parameter. Parameters are strings and maintain their upper/lower case regardless as to delimiters.

The only character that is significant to the command is comma so, if you need to pass a comma as part of the parameter, you must delimit the parameter, but avoid accidentally starting your parameter with a delimiter. If you need to include a delimiting quote or dollar as part of the parameter, specify the character twice.

Examples:

```
INCLUDE FILE 'C:\PROGS\WEEKLY.PQL'
INCLUDE FILE WAREHOUS.PQL (PARTS, PARTNUM PARTNAM PARTCOST)
INCLUDE BUFFER "Test Program"
INCLUDE BUFFER PARTS ( PARTS, PARTNUM PARTNAM PARTCOST)
INCLUDE MEMBER REPORTS.WEEKLY
CALL REPORTS.WEEKLY
CALL REPORTS.WAREHOUS ( PARTS, PARTNUM PARTNAM PARTCOST)
```

Assume a program, TEST1 is called. Any positional parameters in that program are referred to as <1>, <2>, etc. If this now calls a second program, TEST2, it also refers to any positional parameters as <1>, <2>. In the example, in TEST1, the positional parameter <1> is replaced with a "a"; in TEST2, the positional parameter <1> is replaced with a "c". For example:

```
INCLUDE MEMBER TEST1 (a,b)

TEST1:T
WRITE <1>
INCLUDE MEMBER TEST2 (c,d)

TEST2:T
WRITE <1>
```

FINISH

`FINISH`

Terminates processing of all input source commands and returns control to the current environment i.e. the operating system for batch runs or the menus.

If `FINISH` occurs in a set of either VisualPQL or Record Schema commands, the following actions are taken:

- Within a ViusalPQL program, `FINISH` generates an `END PROGRAM` or `END RETRIEVAL` command, then executes the compiled VisualPQL program. At the end of VisualPQL execution, control is returned to the executing process, skipping any other commands in the input source(s).
- Within a Record Schema command set (either `RECORD SCHEMA` or `MODIFY SCHEMA`), `FINISH` generates an `END SCHEMA`, then executes the schema definition. After the schema has been executed, control is returned to the executing process, skipping any other commands in the input source(s)

.

PEND

PEND

Terminates processing of commands in the input source where it appears and resumes execution in the calling input source. Execution resumes with the command immediately following the CALL or INCLUDE that initiated the processing of the procedure containing the PEND. If PEND is executed in the highest level input source, it is equivalent to FINISH.

If PEND is executed within either a VisualPQL or Record Schema definition, the following actions are taken:

- Within a VisualPQL program, PEND generates an END PROGRAM or END RETRIEVAL command, then executes the compiled VisualPQL program. At the end of VisualPQL execution, control is returned to the calling level.
- Within a Record Schema command set (either RECORD SCHEMA or MODIFY SCHEMA), PEND generates an END SCHEMA, then executes the schema definition. After the schema has been executed, control is returned to the calling process.

DO REPEAT

```
DO REPEAT repeat_symbol = parameter_list [ / ... ]
.
.  commands to be repeated with symbol value replacement
.
END REPEAT
```

The `DO REPEAT` command defines a block of text, usually commands, that is repeatedly copied as input. For each copy of the commands, substitution parameters are inserted into the block. For example, if a `DO REPEAT` contains ten lines of code and is repeated ten times, one hundred lines of code are generated. This command is not an alternative to one of the VisualPQL execution time commands such as the `FOR` command as it generates source code rather than looping at execution time.

A `DO REPEAT` block cannot be specified in another `DO REPEAT`.

The *repeat symbol* is a SIR/XS name. The parameter list is composed of one or more parameters. Delimit parameters from each other with blanks or commas. The text within the `DO REPEAT` block is repeated once for each parameter in the list. The value of the parameter is substituted where the repeat symbol is found within the text. By default, the text of parameters is taken to be a SIR/XS name.

For example:

```
PROGRAM
DO REPEAT REPSYMB = First Second Third
WRITE 'REPSYMB'
END REPEAT
END PROGRAM
```

The above text is expanded to be:

```
PROGRAM
WRITE 'FIRST'
WRITE 'SECOND'
WRITE 'THIRD'
```

```
END PROGRAM
```

To specify parameter values that are not SIR/XS names (whether standard names mapped to uppercase or non-standard names enclosed in curly brackets), delimit the parameter with quotes (single or double), the dollar sign ($) or opening/closing parentheses. Delimited parameters can be of arbitrary length (greater than 32 characters) and contain any characters. If you need to include a delimiting quote or dollar as part of the parameter, specify the character twice. Delimiters are stripped out of the text. For example:

```
PROGRAM
DO REPEAT REPSYMB = $FIRST Param$ 'SECOND Param' $THIRD$$Param$
WRITE 'REPSYMB'
END REPEAT
END PROGRAM
```

The above program is expanded to be:

```
PROGRAM
WRITE 'FIRST Param'
WRITE 'SECOND Param'
WRITE 'THIRD$Param'
END  PROGRAM
```

Note. If you happen to have a SIR/XS name that starts with a dollar sign, then you must delimit this name

TO Keyword

Parameter values can be specified with the `TO` keyword. When the parameter values end in numbers and the parameter to the left of the `TO` has a number of lower value than the parameter on the right, the list includes all the numbers implied in the range. For example:

```
PROGRAM
DO REPEAT X = PARAM1 TO PARAM5
WRITE 'X'
END REPEAT
END PROGRAM
```

The above program is expanded to be:

```
PROGRAM
WRITE 'PARAM1'
WRITE 'PARAM2'
WRITE 'PARAM3'
WRITE 'PARAM4'
WRITE 'PARAM5'
END PROGRAM
```

Parameters cannot have embedded numbers. The non-numeric part must be the same. For example the following are both invalid parameters:

```
DO REPEAT x = X12P1 to X12P2
DO REPEAT  x = X1 to Y3
```

Numeric constants can be used as parameters. For example:

```
PROGRAM
DO REPEAT X = 1 TO 3
COMPUTE NUM = X
WRITE NUM
END REPEAT
END PROGRAM
```

The above program is expanded to be:

```
PROGRAM
COMPUTE NUM = 1
WRITE NUM
COMPUTE NUM = 2
WRITE NUM
COMPUTE NUM = 3
WRITE NUM
END PROGRAM
```

Concatenation

`DO REPEAT` can be used to construct names by appending the text of the parameter to other text by specifying the repeat symbol preceded by an exclamation mark (!). For example, if the results of multiple tests are in variables called `MATH1 TO MATH5, READ1 TO READ5` and `AVG1 TO AVG5`:

```
.  DO REPEAT X = 1 TO 5
.  WRITE 'Test Number X Scores: ' MATH!X READ!X AVG!X
.  END REPEAT
```

The repeat block expands to:

```
WRITE 'Test Number 1 Scores: ' MATH1 READ1 AVG1
WRITE 'Test Number 2 Scores: ' MATH2 READ2 AVG2
WRITE 'Test Number 3 Scores: ' MATH3 READ3 AVG3
WRITE 'Test Number 4 Scores: ' MATH4 READ4 AVG4
WRITE 'Test Number 5 Scores: ' MATH5 READ5 AVG5
```

Multiple Repeat Symbols

Multiple repeat symbols, each with its own parameter list can be specified. These specifications are delimited from each other with a slash (/). The lists should be of equal length and a warning is issued if they are not. When multiple repeat symbols are specified, the `DO REPEAT` block is repeated once for every parameter in the longest parameter list, substituting the next parameter from each list in turn. When unequal length lists exist, the parameters of the shorter lists are cycled as often as is necessary to exhaust the longest list. The same result of the previous example can be achieved with the following code.

```
DO REPEAT X = 1 TO 5/  M = MATH1 TO MATH5/
          R = READ1 TO READ5/
          A = AVG1  TO AVG5
.  WRITE 'Test Number X Scores: ' M R A
END REPEAT
```

Other Examples

`DO REPEAT` is useful anytime a set of similar code must be generated repeatedly. Any command can appear within a `DO REPEAT` block except for another `DO REPEAT` command.

Example 1. The following generates three different retrievals, each creating an SPSS file for the data in a different record type. Note that the repeat symbol `RTNUM` is used on the `PROCESS REC` statement and in the filename for the `SPSS` procedure command.

```
DO REPEAT RTNUM = 1 2 4
RETRIEVAL
.  PROCESS CASES
.    PROCESS REC RTNUM
.    GET VARS ALL
.    PERFORM PROCS
.    END REC
.  END CASE
SPSS SAVE FILE FILENAME = 'SPSS!RTNUM.SYS'
END RETRIEVAL
END REPEAT
```

Example 2. Consider a typical schema modification task in which many record types have the same variable with value labels. The following generates the value labels for a variable called `MEDICINE` in ten different record types:

```
DO REPEAT RTNUM = 1,4,7,8,9,13,14,15,20,22
MODIFY SCHEMA RTNUM
VALUE LABELS MEDICINE (1) 'Aspirin'       (2) 'Cough Syrup'
                      (3) 'Antihistamine' (4) 'Ointment'
END SCHEMA
END REPEAT
```

Example 3. Consider an application where many data files are created that have to be entered into the database with the `ADD REC` batch data input utility. The following set of commands reads in data from 27 files and writes out rejected records for each record type to a separate file:

```
DO REPEAT FILENUM = 1 TO 27
ADD REC INPUT = DATA!FILENUM.DAT
        ERRFILE = ERR!FILENUM.LIS
END REPEAT
```

Note that the repeat symbol is a SIR/XS name which allows curly brackets {} as part of the syntax. Command labels, that is labels used by the various `SKIP` command processing commands, also use curly brackets. In the unlikely event that a command label is also a repeat symbol, this can conflict as the `DO REPEAT` processor substitutes the name by the supplied parameter thus eliminating the brackets. In this instance, the substitution parameter must specify brackets in order to operate correctly. e.g.

```
DO REPEAT LAB = ${LABEL1}$
IFCOND (...)  SKIP LABEL1
.....
{LAB}
.....
END REPEAT
```

CIF

```
CIF operator  [arg1, [arg2]]
```

Specify a `CIF` command to begin a block of commands to process only if the stated condition is met. Terminate the block of commands with a `CIF END` command .

The general structure of the `CIF` command and sub-commands is:

```
CIF operator [ argument1 [, argument2 ]]
.   Commands conditionally
processed
CIF TRUE
.   Commands processed if not in skipping mode
CIF FALSE
.   Commands processed if in skipping mode
CIF TF
. Commands processed under all conditions
CIF END
```

`CIF` blocks can be nested within one another. The arguments can only be constants (strings in quotes or numbers). Parameter and global variable substitution into the `CIF` commands is allowed.

The `CIF` command specifies an operator and zero, one or two arguments. The arguments are often values of global variables or of parameters on the `RUN, CALL, INCLUDE FILE` and `INCLUDE BUFFER` commands that are substituted in when the command is processed. The arguments can be strings in quotes or numbers. For the relational tests, arguments must both be of the same type.
(Note that the form of this logical condition is different from VisualPQL logical expressions.)
The operators are:

- `CIF EQ arg1 , arg2` arg1 is equal to arg2
- `CIF NE arg1 , arg2` arg1 is not equal to arg2
- `CIF LT arg1 , arg2` arg1 is less than arg2
- `CIF LE arg1 , arg2` arg1 is less than or equal to arg2
- `CIF GT arg1 , arg2` arg1 is greater than arg2

- `CIF GE arg1 , arg2` arg1 is greater than or equal to arg2
- `CIF B arg1` arg1 is blank, i.e. no argument is specified
- `CIF NB arg1` arg1 is not blank, i.e. an argument is specified
- `CIF DEF global_variable` global variable is defined
- `CIF NDEF global_variable` global variable is not defined

The **B** and **NB** conditions test whether the remainder of the command line is blank or not.

The **DEF** and **NDEF** conditions test whether the global variable name specified is currently defined.

Conditional blocks can be nested to any level, i.e. a conditional block can be contained within another conditional block. The inner conditional block is executed only if it is encountered while commands are not being skipped by any outer conditional block.

Each `CIF` conditional command must have a corresponding `CIF END` command.

Consider the following:

```
CIF EQ <1>,3
.
. commands
.
CIF END
```

If these commands are run with a parameter list of `(5)`, the commands within the conditional block are skipped and not processed. If the parameter list is `(3)`, the commands within the conditional block are processed.

CIF FALSE | TRUE | TF

```
CIF  FALSE
CIF  TRUE
CIF  TF
```

The `CIF` sub-conditional commands alter the effect of the previous `CIF` conditional. The sub-condition affects commands up to the next sub-condition or until the end of the block specified by the matching `CIF END` command. If a `CIF` conditional command is encountered while commands are being skipped, that entire block is skipped, up to and including its matching `CIF END` command.

`CIF FALSE`
: Stops statement skipping if the last condition was false. If the last condition was true, skipping either starts or continues.

`CIF TRUE`
: Stops statement skipping if the previous condition was true. If the previous condition was false, skipping either starts or continues.

`CIF TF`
: Stops skipping. Statements following `CIF TF` are always processed.

CIF END

```
CIF END
```

Terminates a conditional block. If this command terminates a nested conditional block, the condition of the previous block is restored, i.e. proper nesting occurs.

```
CIF EQ 3,3
.
.  (processed)
.
CIF FALSE
.
.  (skipped)
.
CIF EQ 1,1
.
.  (skipped)
.
CIF END
.
.  (skipped)
.
CIF END
```

```
CIF EQ 3,4
.
.  (skipped)
.
CIF FALSE
.
. (processed)
.
CIF EQ 1,2
.
. (skipped)
.
CIF END
.
.  (processed)
.
CIF END
```

Input Flow Control

As an alternative to the `CIF` block structures, there are a number of commands that give the ability to skip commands to a named point depending on conditions.

Seven commands test logical conditions relating to global variables and the existence of procedure families and members. Logical test commands specify the condition being tested and skip when the condition is true.

The general syntax of the commands is:

```
IFxxxxxx condition SKIP label
```

The commands are:

`IFCOND (condition) SKIP label`
: Test a specified condition.

`IFMEMBER membname SKIP label`
: Tests for the existence of the named member and skips when the member exists. See member references for details on specifying member names.

`IFNOTMEMBER membname SKIP label`
: Tests for the existence of the named member and skips when the member does not exist.

`IFFAMILY famname SKIP label`
: Tests for the existence of the named family and skips when the family exists.

`IFNOTFAMILY famname SKIP label`
: Tests for the existence of the named family and skips when the family does not exists.

`IFGLOBAL global_varname SKIP label`
: Tests whether a global variable is initialized and executes the flow command when the global is initialized.

`IFNOTGLOBAL global_varname SKIP label`
: Tests whether a global variable is initialized and executes the flow command when the global is not initialized.

The `SKIP` that follows a condition directs the flow of control to a named label. The name is used in the standard way on the `SKIP`.

Label a line in the input source by enclosing the name in braces {} starting in column one. Although this may look like a SIR non-standard name, it is not. Specify a SIR name inside these curly brackets. The label syntax of a curly bracket in column 1 predates non-standard names and has been retained for compatibility with existing procedures.

There are three variant `SKIP` commands:

`SKIPBACK`
: Looks for a label from the beginning of the input source. Execution continues with the line following the specified statement label.

`SKIPFORWARD`
: Looks for a label from the current command. Execution continues with the line following the specified statement label. `SKIP` is a synonym.

`SKIPRETURN`
: Returns to the calling procedure. One input source can `INCLUDE` another procedure. `SKIPRETURN` returns control back up a level.

  If there is no label on the command, control is transferred to the first statement following the calling command in the calling procedure.

  If the command specifies a statement label, control is transferred to the beginning of the calling procedure, statement skipping begins and execution continues with the first command following the specified label. This is equivalent to a `SKIPRETURN` and `SKIPBACK`

  . There is no direct `SKIPRETURN` and `SKIPFORWARD`.

Labels need not be unique. The first match found is used. There is one predefined label, the {*}. When looking for a named label, the {*} label matches anything.

IFCOND

`IFCOND (logical_condition) SKIP label`
Tests the specified logical condition and skips when the condition is true. The command must fit on a single physical line. Specify the condition in parentheses. It consist of:

- global variables treated as string variables
- string constants (enclosed in single quotation marks)
- numeric constants
- global variable text substitution specifications (e.g. <gvarname>)
- parentheses
- operators:
    - string (+)
    - numeric (EQ =, LT <, GT >, LE <= , GE >=, NE ><)
    - logical (AND, OR, XOR)
    - unary (+, - , NOT)

The `IFCOND` command tests the value in a global variable. So global variables can either be used for text substitution by enclosing them in angle brackets, or referenced as string variables. If a global variable enclosed in angle brackets is found, the value of the global variable is substituted in the text. If a string, not in quotes, is found, it is treated as a global variable. All globals are string variables.

To use globals to set numeric values, the global name must be enclosed in angle brackets. To use a global that contains a string value, either specify the name of the variable, without angle brackets or quotes, or specify quotes around the angle brackets.

- `IFCOND (gvar = 1)` Mixed mode error since gvar is taken as a string
- `IFCOND (gvar = 'A')` OK since gvar is a string
- `IFCOND (<gvar> = 1)` Works if a numeric value assigned to gvar. Gives 'Unknown global' error, if an unquoted string is assigned to gvar.
- `IFCOND ('<gvar>' = 'A')` OK since a string is being compared to a string.

For example:

```
IF (<gvar> EQ 1) SKIPBACK TOP
IF ('<gvar>' EQ 'A') SKIPBACK TOP
IF (GVAR EQ 'A' ) SKIPBACK TOP
```

EJECT

```
EJECT [ ODD | EVEN ]
```

Causes the output listing in batch to skip to a new page. The `EJECT` command is not listed.

`ODD`
: Causes the listing to continue on the next odd numbered page.

`EVEN`
: Causes the listing to continue on the next even numbered page.

ERROR LIMIT

```
ERROR LIMIT { number | NONE }
```

Specifies the maximum number of error messages that are written for a given task such as a Record Schema definition or VisualPQL program. Once the limit is reached, no more messages are printed. Syntax checking continues and the summary of error messages lists all errors encountered, including those detected after the limit was reached. The keyword `NONE` suppresses all error messages.

A task is defined by `TASK NAME` commands. If no tasks are defined, the whole session is a single task.

PRINT BACK

```
PRINT BACK [ ON | OFF ] [ SAVE | RESTORE]
 [ [NO]CALL     ]
 [ [NO]COMMANDS ]
 [ [NO]FORMAT   ]
 [ [NO]REMARK   ]
 [ [NO]REPEAT   ]
 [ [NO]SKIPPED  ]
 [ [NO]TASK     ]
 [ [NO]USER     ]
```

Controls the listing of procedure commands and remarks in the output file or on the screen. `PRINT BACK` can be used anywhere in a set of commands.

`ON or OFF`
: Controls whether or not the commands and related messages following the `PRINT BACK` command are listed in the output file. `OFF` is equivalent to specifying all the `NO` options on the command.

  In batch mode, all procedure commands, remarks, error messages and warnings are written to the output file. These can be suppressed with the `OFF` keyword, in which case only the program output appears in the output file.

  In interactive mode, remarks and messages are displayed on the scrolled output buffer and commands are not. The interactive output can be directed to an output file with the `SET OUTPUT` command in which case the output is treated as in batch mode.

`SAVE | RESTORE`
: `SAVE` stores the current `PRINT BACK` options. `RESTORE` restores a previously saved set of `PRINT BACK` options. For example, to suppress the listing of a procedure that is `CALL`ed, use the following in the procedure:

  ```
  PRINT BACK SAVE , OFF
  .   text of procedure
  PRINT BACK RESTORE
  ```

`CALL`
: Controls whether the commands inserted into the procedure through a `CALL` command are listed.

`COMMANDS`
: Controls whether or not the commands are listed.

`REMARK`
: Controls the listing of remarks such as "Start Translation" and "Finish Execution".

`REPEAT`
: Controls whether the commands generated by a `DO REPEAT` are listed.

`SKIPPED`
: Specifies that commands skipped during translation of `CIF` blocks are listed. The default is `SKIPPED`, skipped commands are listed.

`TASK`
: Specifies that messages are printed about compilation, CPU usage and other information is output in the job log for batch runs. Use `SET TASK` for interactive runs.

`USER`
: Specifies that attributes are listed when they are created.

SPACE

```
SPACE number
```

Generates the specified number of blank lines in the output listing.

STRING LENGTH

```
STRING LENGTH num
```

Sets the default length of implicitly declared string variables in VisualPQL or in the Record Schema. This is also the default length used for explicitly declared string variables where a length is not specified. The default string length is 32 characters.

WARNING LIMIT

```
WARNING LIMIT { number | NONE }
```

Specifies the maximum number of warning messages written to the output file for a given task. Once the limit is reached, no more warning messages are printed. Syntax checking continues and the summary of warning messages lists all warnings encountered, including those detected after the limit was reached.

The keyword `NONE` suppresses all warning messages.

A task is defined by `TASK NAME` commands. If no tasks are defined, the whole session is a single task.

Commands

This section describes commands that are not generic and can only be used outside VisualPQL programs or schema compilation in a similar way to running database utilities or batch data entry utilities. Commands are:

- `ATTRIBUTE`
- `COPY DIFFERENCE FILE`
- `PAGE SIZE`
- `PRINT FILE`
- `RUN MEMBER`
- `RUN NAME`
- `SET`
- `SHUTDOWN MASTER`
- `TASK NAME`

There are also a set of commands to manage members in a procedure file.

Note. Commands that are not valid VisualPQL are flagged as errors. `RUN NAME` and `TASK NAME` are recognised by the VisualPQL compiler and treated as indicating the end of that program.

ATTRIBUTE

```
ATTRIBUTE attribute_name FILENAME = "file_name"
```

Defines a SIR/XS name that is equivalent to the specified external filename. The file name must be enclosed in quotes. The file name can be any valid operating system file name, including any path, subdirectory or device specifications.

Once defined, the short attribute name may be used anywhere that a filename can be specified.

SIR/XS creates and uses an attribute for every filename that it uses. List the currently defined attributes from the *Settings* menu item. Define attributes with the `ATTRIBUTE` command, with the `OPEN` command in VisualPQL and from the *Settings* system menu.

COPY DIFFERENCE FILE

```
COPY DIFFERENCE FILE
```

If the session is connected to Master, requests that all databases connected at that time through that Master are written to disk so that updates are available for other processes.

PAGESIZE

```
PAGESIZE { lines_page | NOEJECT } , characters_line
```

Controls the size of the page if output is directed to an output listing file. The lines per page specification is the maximum number of printable lines on a page. The default is 60 lines per page including three lines at the top used for run and task headings. The `NOEJECT` keyword suppresses new page generation, i.e. the printing is continuous. The default characters per line is 121.

This has no effect on VisualPQL procedure output files that have their own `PAGESIZE` options.

PRINT FILE

```
PRINT FILE filename
```

Initiates the operating systems print routine for the named text file.

RUN MEMBER

```
RUN MEMBER membername:[E|T] (args)
```

Runs the named member with any specified positional parameters. Can be used to compile text (:T) members or execute saved (:E) executables.

If a member name without a suffix is `RUN`, and both a :T or :E version exist, the system determines which to run based on dates and times of last update.

RUN NAME

```
RUN NAME text
```

Sets a main title for the output listing. Ends VisualPQL or schema processing. Has no effect on error or warning counts.

SET
CLEAR

There are a number of system settings and parameters that can be set by command. The `SET` command sets the parameter to a particular value specified on the command.

The `CLEAR` command can be used with some parameters to reset back to the default.

The following parameters can be used:

`CENY yyyy`
: Sets the year used for calculation of the default century when two digit years are input. When a two digit year is input, the system compares the input year with this parameter. If the parameter is less than the input, the century is set to the specified century. If the parameter is greater than or equal to the input, the century is set to the specified century plus 1.

  The default is 1930.

  To illustrate the calculation, assume the default setting of 1930. Then any input year in the range 00 - 29 is assigned a century of 20nn; any input year in the range 30 - 99 is assigned a century of 19nn.

  All variables defined as dates are held internally as a number of days since the start of the Julian calendar on Oct 15 1582.

`DEFINDEX indexname`
: Sets the default index on a table. Can be cleared.

`DEFTABFILE tabfilename`
: Sets the default tabfile. Can be cleared.

`DEFTABLE tablename`
: Sets the default table.

`EDITOR editorname`
: Sets the name of the external editor. Can be cleared.

`FAMILY familyname`
: Sets the default family name and password. The default family is also updated by commands that reference families and members.

`LOADING .nn`
: Specify a value between .01 and .99.

  Sets the loading factor used to determine how data blocks in the database are split as they become full. This setting affects Import and Reload of a database, VisualPQL retrieval updates and batch data input.

This can also be set by parameters on the VisualPQL `RETRIEVAL` command and on Batch Data Input commands.

`MASTER name`
: Sets the use of Master. If a name is not specified, sets the Master to be the previously used Master.
`CLEAR MASTER` turns the use of Master off.
Equivalent execution statement parameter: MST=name

`MEMBER [family[/password].member[/password]`
: Sets the default member name and password. Equivalent execution statement parameter: M=membername

`OUTPUT filename`
: Sets a file as the standard output file. Can be cleared and, by default, output goes to the output window. Output is written to the standard output file by VisualPQL programs that WRITE to an unspecified file, and by utilities.
`CLEAR OUTPUT` cancels the filename of the default output file and output is then sent to the standard file (in batch mode) or output window (in interactive mode).

`PROCFILE filename`
: `SET PROCFILE procfilename` connects the named procedure file and disconnects the currently connected procedure file. Use the system attribute `SYSPROC` to connect the system procedure file. Use a blank filename to set the procfile to the procedure file of the current database.

`SERVER NOOUTPUT`
: Controls output when running on the PQLServer. Can be used anywhere in the command stream to selectively control what output is made available for retrieval by the client. `SET` means that any output directed to standard output is thrown away. If this command is within a PQL program it would affect any compilation listing. Use the `SERNOOUT` function for execution time control.

`SORTN nnnnnn`
: Sets the number of records expected to be sorted as a default. This is used where the default (number of records on the database) does not apply. Can also be set on VisualPQL Procedure commands.

`WINPAGE`
: Output written to the output window is normally not paged. Set `WINPAGE` to put page breaks, headings, etc. in the output window for subsequent printing.

SHUTDOWN MASTER

```
SHUTDOWN MASTER [NOLOGONS] [password]
```

If the session is connected to Master, requests that Master to shutdown.

The keyword `NOLOGONS` specifies that the Master rejects any further client logons and warns existing clients that the shutdown is in progress. Master only completes the shutdown when there are no clients connected.

If Master was started with a password, specify the password on this command.

TASK NAME

```
TASK NAME text
```

Indicates the end of one task and the start of a new task. This ends any VisualPQL compilation or schema definition.

Can define a subheading printed below the main heading at the top of each page of the print back listing. `TASK NAME` can be specified as often as needed.

Procedure File Commands

There are a number of commands that can be used to manage a procedure file. Commands are:

- `CREATE PROCFILE`
- `CONDENSE`
- `CREATE FAMILY`
- `DELETE FAMILY`
- `DELETE MEMBER`
- `PLIST`
- `PREAD`
- `PROCEDURE`
- `PWRITE`
- `RENAME`

Creating

A procedure file is automatically created when a database is created, You can create procedure files that are available for generic use and not part of a particular database with the `CREATE PROCFILE` command.

Procedure files may be protected by access restrictions defined at the operating system level. Users of an alternate procedure file must have operating system read access and may be allowed write access.

Deleting

The procedure file is normally deleted along with the rest of the database files when the database is purged (deleted). When the `Delete Database` option is used from the `Database- Recover` menu, the procedure file can be deleted or can be left intact. If a database is reloaded and the procedure file already exists, the existing procedure file is used and any reloading of procedures is skipped. If a database is imported and the procedure file already exists, the existing procedure file is used and any imported procedures are added to it. If they have the same name as existing procedures, you are prompted as to whether to replace the existing with the import.

Member References

Many commands allow the specification of a member name or a list of member names and member names can be specified in a number of ways. A member can be uniquely referenced by its procedure file, family, member name and member type and both the family and member may be followed by a password. The various elements only have to be specified where used. For example, if the member does not have a password, it is not necessary to specify a blank password.

When specifying names with multiple elements, delimit the procedure file, family and member by full stops. Delimit passwords by a slash. For example:

```
CALL SYSPROC.PQL/FAMPASS.PROG1/MEMPASS:T
```

`SYSPROC` is the procedure file, `PQL` is the family with `FAMPASS` as the family password and `PROG1` is the member name with `MEMPASS` as the member password. The **:T** indicates that this is a text member and is typically not required when referencing text members. Note that the procedure file specification is a name and this must have been associated with the required long filename, for example by a `SET PROCFILE` command.

If `PROG1` has no passwords and is in the default procedure file, specify the following:

```
CALL PQL.PROG1
```

The default family is used if a family name is not specified. The initial default family is the `SYSTEM` family. After a member is accessed, the family of the last accessed member becomes the default family. The default family may be explicitly set with the `SET FAMILY` command.

The SYSTEM Family

The `SYSTEM` family is predefined. It may contain login scripts as well as other members.

Family References

Some commands operate on families rather than members. The above specification of names applies to specifying a family name, that is the family name can be preceeded by a

procedure file reference and followed by a password as required.

CONDENSE

```
CONDENSE
```

Condenses the procedure file removing unused space.

Use this command on a regular basis, especially after many members have been updated.

Copies all members in the procedure file to a temporary scratch file and then rebuilds a condensed procedure file.

CREATE FAMILY

```
CREATE FAMILY fname [ /password ]
```

Creates a family on the procedure file. A family is a group of members.

Specify a password if required.

DELETE FAMILY

```
DELETE FAMILY [ fname, fname, ... ]  [ /NOINFORM ]  [ /OK ]
```

Prompts for permission, then deletes specified families.

All members in the family are deleted. This is a DBA-only command.

`NOINFORM`
: Suppresses informative messages for each family deleted.

`OK`
: Suppresses the OK prompt for permission to delete the family.

DELETE MEMBER

```
DELETE MEMBER membername,... [DBA ]  [NOINFORM ] [OK ]
```

Deletes the specified member(s).

`DBA`
: Deletes a member without requiring the member password. DBA security is required to do this.

`NOINFORM`
: Suppresses message that member has been deleted.

`OK`
: Deletes without requesting confirmation.

PLIST

```
PLIST fileid [ namelist ]
             [ FAMILY ]
             [ NOINFORM ]
             [ RULER ]
             [ UNNUMBERED ]
             [ UPPER ]
```

Writes a member to an external file in a suitable format for printing (with headers, paging, etc.).

`fileid`
: Specify the output file. This is required.

`namelist`
: A list of members or families to list. If the list is omitted, all members are listed.

`FAMILY`
: `FAMILY` specifies that the namelist is a list of families. By default, the namelist is a list of members.

`NOINFORM`
: Specifies that the number of lines listed is not reported. The default displays each member name and lines written.

`RULER`
: Specifies that a ruler is displayed before the first line is listed. The default does not display a ruler.

`UNNUMBERED`
: Specifies that the lines are not numbered. The default displays line numbers.

`UPPER`
: `UPPER` specifies that the listing is in uppercase. The default displays in upper and lower case.

PREAD

```
PREAD fileid  [CONFIRM]
              [NOINFORM]
              [PUBLIC]
              [REPLACE]
              [REPORT]
```

Reads the contents of the specified file and saves the contents as members in the procedure file.

A message is displayed for each member. If the new member has the same name as a current member, a prompt is issued for permission to replace the current member.

The disk file must be in the correct format for a `PREAD` that is produced by `PWRITE`. The file can contain many members, and each member must have the following format:

```
PROCEDURE  family.member
.      text of procedure
END PROCEDURE
```

`CONFIRM`
: Prompts to confirm the addition of any member.

`NOINFORM`
: Suppresses the display of informative messages for each member read. Sets the `REPORT` keyword.

`PUBLIC`
: Saves members as `PUBLIC`.

`REPLACE`
: Suppresses the confirmation prompt when replacing existing members of the same name. When `REPLACE` is not specified, a prompt is issued for permission to replace the member.

`REPORT`
: Gives the total number of added/replaced members after the entire file has been read. This is set on if `NOINFORM` is specified.

PROCEDURE command

```
PROCEDURE  [family[/password].]member[/password]
   [REPLACE] [PUBLIC]
.
.   text of member to be stored
.
END PROCEDURE
```

The text in the PROCEDURE block is saved in the named member. if this command appears in the input it saves a member and can be used to save a member during a batch run.

These commands can appear anywhere. If a PROCEDURE command appears within a VisualPQL program or Record Schema definition set, compilation is suspended until the END PROCEDURE command is encountered, upon which compilation is resumed.

REPLACE
: Allows an existing member of the same name to be overwritten (replaced). The default is not to replace an existing member.

PUBLIC
: If the family or member is protected by passwords, PUBLIC allows all users to execute the member but not to read or modify it.

PWRITE

```
PWRITE fileid  [ mlist ]  [ FAMILY ]  [ NOINFORM ]  [ NOTIMES ]
```

Writes the specified member(s) to the named file. If no members are specified, writes the entire Procedure File.

`FAMILY`
: Specifies that the list is a list of family names and that all members of the specified families are written to the fileid.

`NOINFORM`
: Normally a list of members written to the file is produced. `NOINFORM` suppresses that list.

`NOTIMES`
: Normally time and date stamps are written to the file for each procedure. `NOTIMES` suppresses this which creates a file that can be read by earlier versions of SIR.

RENAME

```
RENAME [family.]member1 [family.]member2 [[NO]PUBLIC]
```

Renames member1 to member2. Member1 no longer exists in the procedure file.

Rename a member to itself to alter the password or the `PUBLIC` option. Specify the old password on member1 and the new password on member2. `PUBLIC` allows people to run a member without knowing the member password.

RENAME FAMILY

```
RENAME FAMILY family1 family2
```

Renames family1 to family2. Family1 no longer exists in the procedure file, all members of family1 are now in family2. Rename a family to itself to alter the password. Specify the old password on family1 and the new password on family2.

Concurrency

Multiple users can work on the same database at the same time, providing they are not updating it. A database is protected against accidental updating by two separate users at the same time and a second update process is not given access.

If you need multiple processes to update a database at the same time, use MASTER. This is an independent process that controls updates ensuring that the same record cannot be updated by more than one person at the same time. This means that multiple programs can be running at the same time, and all of the updates they perform are coordinated.

The clients request concurrent update operations to a database through a central MASTER.

To make a client use MASTER, first start MASTER (see Running Master) and then start the client SirFORMS, SirSQL or SIR/XS session with the MST = execution parameter. These then pass all requests for data to MASTER. During a SIR/XS session, you can turn the use of Master on and off as necessary. You can do this through the Master Settings menu or with the SET MASTER/CLEAR MASTER commands. It is much faster to retrieve data from the database directly rather than through Master, so it is good practice to turn off Master when using retrievals that do no updating.

A client has an open connection to Master but only has databases attached while a VisualPQL program or SQL retrieval is running that uses them. In Forms, where a database is accessed interactively a record at time, the database is open and in use while the form is active. A database is only in use while the client is accessing it. A MASTER process does not access any database simply by being started.

While a client is using MASTER to access a database, the database is locked against update access by any other process. The database can only be updated through MASTER. If a database is opened in update mode without going through MASTER, the database is locked from access by any other process for the duration of that update.

There are three parts of a database that a process has to access, the dictionary, the procedure file and the data. MASTER only provides concurrent access to the data. Thus all client processes need access to the dictionary and procedure files and so must be able to access the database directly through the network as well as using MASTER.

While MASTER is connected to a database, there are certain things that cannot be done to it

- The schema cannot be changed;
- Utilities such as UNLOAD/RELOAD cannot be run;
- Batch data input utilities cannot be used.

All of these facilities must be run independently of any other processes and require exclusive access to a database.

Multiple Master/Clients/Databases

The concurrent processing environment can include multiple MASTER servers, multiple databases, and multiple client processes. Each MASTER server has the capability of being accessed by many client processes and accessing many databases. A given database may be opened by only one MASTER:

- A database can be connected to only one MASTER at a time.
- A MASTER can control many databases.
- A client process may only access one MASTER.
- A MASTER can be addressed by many client processes.

Simultaneous read access

While a database is being accessed through MASTER, only the MASTER clients can update the database. However, the non-concurrent products can read the database. In order to provide a consistent image for any independent read-only processes, MASTER does not update the database directly. It leaves all of the index blocks and data blocks that comprise the original database strictly alone. It creates a new master index block in memory and then creates new index or data blocks whenever it has to re-write a block to disk.

The database updates are written at the end of the database file and are referred to as the *Difference File*.

Users accessing the database through MASTER, see the current database, including the most up to date modifications.

Users accessing the system for read only through single-user products see the database as it was when the difference file was last copied. This view is frozen at the point that the retrieval process starts reading the database. Thus, some users can do retrievals on a stable database, while others can update the database without worrying about leaving a particular record only partially updated or created.

Difference File Copying

Difference File Copying means that the database is brought up to date. The phrase "Difference File Copy" is something of a misnomer since very little copying happens. All of the updated blocks are physically in the database but the Master Index for the database ignores them. A difference file copy is simply the re-writing of the new master index that is a trivial task in terms of the amount of work involved.

A difference file copy only happens when `MASTER` can get exclusive control of the database. Read access by a non-concurrent product always sees the database as it was when the program started. A Difference File Copy occurs:

- When `MASTER` is shutdown
- When the last concurrent user closes a database
- When told to do the copy
  There are two ways to tell `MASTER` to do a difference file copy:
  - Use the `COPY DIFFERENCE FILE` command from a SIR/XS session that is attached to Master.
  - On the machine that is running Master, interrupt Master and go into command mode and use the `COPY` command.

  All databases currently in use by Master are updated. The database updates are now available to stand-alone processes.

The difference file causes the database to grow by the maximum number of updates performed between difference file copies. Any block that is updated is copied to the difference file. If that block is updated again, it is simply rewritten. When the difference file copy is done, the original blocks are no longer required. These are not returned to the operating system but are available for additional updates. If more updates are done, these available blocks are used first. The database thus grows to the point required to hold the maximum number of temporary difference file blocks and no further. The database can be compressed by performing an `UNLOAD` and `RELOAD`.

Running Master

The communication process uses TCP/IP protocols and these must be available on your network.

The MST= parameter is the name of the machine where master is running. When Master starts, it tells you the machine and Port (normally :3000). To use a different port, specify the port to use with an MST= parameter on the Master start up.

You can find the machine name from Windows using Control Panel/Network/tcpip/config. The machine name consists of a host and a domain. It makes the start up for clients faster to quote both the host and domain name (DNS).
e.g. Start Master - Get message Master started SirNT:3000
Start a Forms session as a client SirForms.exe MST=SirNT
(If you know the domain name, startup is faster e.g.
SirForms.exe MST=SirNT.sir.com.au)

A connection is established to the specified master process. The client process then specifies a database to access in the normal fashion. MASTER attempts to access that database.

If the user running the client process did not start MASTER (for example, it may be started by the system administrator), the default directories that MASTER is using may differ from the client's. In this case, specify the database prefix in order for MASTER to locate the particular directory that holds the required database. In SIR/XS, this is the prefix on the database connection or on the execution statement, in SirSQL on the CONNECT and in SirFORMS on the DATABASE clause on the FORM statement.

Note: Master allows multiple clients to access the same database and must decide whether a request for access by a client refers to a database that Master is already using. It uses the prefix and the database name to establish this. Master resolves the prefix into a standard form removing any relative references but Master has no way of knowing that different forms of prefixes (e.g. use of different drive letters by different clients) point to the same database. Ensure all client prefix references are identical.

When MASTER is started, it is not attached to any database. MASTER is told which database(s) to use by a client. A single MASTER can access multiple databases. SIR/XS, SirFORMS and SirSQL can access multiple databases through a single MASTER. While a client is using MASTER, all database access is through MASTER.

Command mode

Clients signon and signoff from `MASTER` as necessary so `MASTER` is not expecting local input. If you wish to give `MASTER` a command, press `CTRL-Break` in the `MASTER` process window. This stops `MASTER` responding to any clients and makes it ready to receive commands. To allow `MASTER` to continue, press `Enter`. If you leave `MASTER` in command mode, clients cannot communicate with `MASTER`.

Shutdown

To shutdown `MASTER`, go into command mode and enter the command `EXIT`. The shutdown:

- disconnects all users;
- updates all attached databases with difference file copies;
- closes the attached databases.
- stops `MASTER`

Having a `MASTER` server that is not shut down has very little overhead. When the last concurrent user of a database signs off, the database is updated with the difference file and is then closed.

Information

Use the `LIST` command to get certain information about the current session. You can list information about:

- `CLIENTS` - currently connected client processes, ip addresses, time of logon, time of last message and overall number of logical requests made.
- `DATABASES` - currently connected databases.
- `STATISTICS` - Time started, physical messages processed*, logical requests processed*, number of logins, maximum number of users and when that maximum was reached.

The client listing assigns an id to each connection. You can delete client connections with the `DELETE id` command if necessary if a client process has simply 'gone away' without disconnecting properly.

Please note that while you are using command mode clients cannot use Master.

*While the counts of messages are difficult to relate to specific activity, they can be used to give an overall, general indication of usage or to see that a particular client connection is inactive. In outline, a program issuing a database request must go through a number of logical steps to establish the request, create keys for searches, test success, retrieve data, etc. Where possible these logical requests are combined into single physical messages. However large data records may have to be split across physical messages so the exact relationship between records accessed, physical messages and logical messages is highly dependent on individual circumstances.

Record Locking

Data integrity is ensured through record locking. When updating a record, the lock on the record being updated prevents other users from simultaneously updating the record. Defaults can be used or explicit locks can be specified as required. Locks apply to either whole cases or individual occurrences of a record; locks do not apply to the database as a whole, or to all records of one type.

Writing concurrent programs

When designing systems for a multi-user environment, there are particular facilities concerned with that environment such as locking records. A program that uses these commands for multiple users, can be run in the single user environment without change.

There do not have to be any changes in the source code of a program designed to run in the single user mode to use it in a concurrent environment. If a program was developed for single user mode, it can be run in a multiple user environment and default locks are used.

Lock Types

There are five lock types:

Lock Type	REQUESTS	ALLOWS
CONCURRENT READ	read only access	other readers and writers
CONCURRENT WRITE	read and write access	other readers and writers
PROTECTED READ	read only access	other readers - no writers
PROTECTED WRITE	read and write access	other readers - no writers
EXCLUSIVE	read and write access	no other readers or writers

When a lock is requested, any current locks on the requested record are scanned for compatibility. For example, if a Concurrent Write lock is requested on a record, the request is granted if the other locks currently granted are Concurrent Read or Concurrent Write. The request is denied if there is a Protected Read, Protected Write, or an Exclusive lock already taken out on the record.

Establishing a Set of Lock Types

Selecting appropriate lock types for an application depends on the type of work to be done.

For users whose jobs entail only reading of data use Protected Read. This allows multiple users to read a record, but prevents other users from simultaneously modifying or updating the record.

If numerous people have to read records simultaneously, but data input is handled by particular individuals, use separate definitions for readers and writers, and then use Concurrent Read locks for the readers and Protected Write locks for the writers.

For maximum control of data input and updating, use Exclusive locks. This allows only one writer and no readers.

Locking can be applied at the case-level or at the record-level. When a case-level lock is required, select a lock type to lock both the CIR and the records for a case. Specify separate lock types for the CIR and records for general access to the case, but a more restricted access to the records.

Locks are set and released by the individual module. Each module has defaults and these can be altered in most cases under program control. For example in `FORMS`, the `LOCK` command can be specified for the whole form, when calling a particular screen, and when a lookup is done on a record. An interactive forms user can change lock type on the records they are accessing if the form allows this. Locks can be set and altered in `SirFORMS` and `VisualPQL`. Locks in `SirSQL` cannot be altered.

ODBC

The SIR/XS ODBC driver implements Core level ODBC API, plus some functionality of Level 1 that is commonly used by ODBC applications to import data. The driver has been tested with MS Access, MS Excel, MS Query and Minitab, SAS and SPSS Statistical Software to import data from SIR/XS data sources into these products.

The SIR/XS ODBC Driver provides client services to ODBC enabled applications. It uses the SirSQL Server to retrieve data. The driver is designed for 32-bit systems only, and runs on Windows (95/98 and NT). It cannot be used under Win32s.

The SIR/XS ODBC Driver does not require a licence. You may copy all the files from the `SIRODBC` directory onto a floppy disk or onto a directory on a network drive for free distribution among your users. You may install and use the driver on as many computers as you need. (N.B. These can only access data from licenced remote SirSQL Servers.)

The SirSQL server runs under the various operating systems supported by SIR/XS (Windows 95/98, NT, Unix, OpenVMS) and SIR/XS handles all issues of cross machine compatibility. SirSQL Server is part of SIR/XS package and you need a licence for each computer where you install it.

Installation

The ODBC driver is automatically installed as part of the SIR/XS installation. There is a separate installation procedure for the ODBC driver in a standalone mode.

Installing the driver in a standalone mode

Run `setup.exe` from the `SIRODBC` directory.

This copies sirdbc.dll into the Windows System directory. If you install the complete SIR/XS locally after you have installed the standalone ODBC driver, remove the sirdbc.dll from your Windows System Directory since this is installed as a part of SIR/XS into the SIR/XS home directory on a full installation.

Configuring ODBC sources

After the driver is installed, use the ODBC item on the Control Panel to configure the data sources.
Select the User DSN tab, click Add...;
select SIR the list of ODBC drivers and click OK. You can use SIR/XS databases and tabfiles as ODBC data sources.

SIR/XS ODBC driver sends requests to a SirSQL server and the data sources are located by the server. If you use a remote server, the location of the data source must be specified so that the server can locate the data source on its directory structure.

If you setup a SIR/XS database as a data source, type the database name in **UPPERCASE** in SIR Name field and the directory name (make sure you specify a terminating directory separator) in Location field. The database name must be in **UPPERCASE** because many OBDC clients use this name literally (in double quotes) in their SELECTS. "COMPANY" is NOT the same as "company".

If you setup a SIR/XS tabfile as a data source, type the tabfile name (not the external file name) in the SIR Name field and the full name (directory and filename) of the physical file in the Location field.

Don't forget to select the corresponding storage type in the Type choice control.

For local data sources you can use the Browse button and the program configures the data source for you.

The Server field specifies the server's host name and, if the server uses a non-standard port, the port number. SIR/XS SQL Server uses port 3050 as its default, standard port.

If you run the server locally (on the same machine as the ODBC client), leave the Server field empty.

Examples:

1) Local database

```
DS Name:   Fast cars
Comments:  To qualify must do the quarter under 15 sec
Server:
File type: SIR Database
Location:  d:\personal\intimate\cars\
SIR name:  FASTCARS
```

2) Local tabfile

```
DS Name:   Speedtraps in NSW
Comments:  Speed kills
Server:
File type: SIR Tabfile
Location:  d:\personal\intimate\cars\police\radars_nsw.tbf
SIR name:  RADARS
```

3) Remote database on Unix system

```
DS Name:   Diving in Australia
Comments:  Dive sites, operators, clubs
Server:    wildrun.com.au
File type: SIR Database
Location:  /home/fred/diving/
SIR name:  DIVEAUS
```

4) Remote tabfile on Unix system (non-standard port 3800)

```
DS Name:   Dive gear
Comments:  Stuff from the latest catalogues
Server:    realmad.com.au:3800
File type: SIR Tabfile
Location:  /users/bob/underwater/gear.tbf
SIR name:  DIVEGEAR
```

Some ODBC applications can only use File DSNs (as opposed to User DSNs). You can tell if this is the case because, even though you have configured a User DSN for your ODBC source, the application does not show it in the list of available ODBC sources. If this is the case you need to create a non-shared File DSN pointing to your User or System DSN.

Create a File DSN for SIR/XS ODBC driver (ignore any verify error messages). Go to directory where File DSNs are stored and edit the .dsn file, so it points to your real DSN (let's say 'My data' is the name of your User DSN) and doesn't include the DRIVER keyword. e.g.

```
[ODBC]
DSN=My data
```

Your applications can then access SIR/XS data sources using both User and File DSN lists.

Logging on in an ODBC Application

Normally your ODBC application allows the SIR/XS driver to ask you about the login information and to specify passwords, etc. In this dialog, you are asked for user name and password plus passwords specific to the data source.

The user name and password are not pre-registered on the server and can be anything you wish. Each user can have multiple connections using ODBC and the user name and

password allow you to do queries across multiple data sources. You can query any data source with the same user name/password. Using a specific name and password prevents other users from retrieving the data from the data sources opened by the server for you.

Other passwords are normal SIR/XS database or tabfile passwords.

In the unlikely case that your ODBC application does not display the SIR/XS dialog but asks you about user id's and passwords directly, you may be given only two edit fields. In this case, type all id's separated by commas in the ID field, and all passwords separated by commas in the password field.

SirSQL Server

The SIRSQL server must be running somewhere on the network to support any ODBC clients or any VisualPQL client processing. Start the server by running `SirSQLs.exe`. This starts the server at the default port of 3050 and, in a text window, reports the current IP address. There are a small number of command line options:

`SirSQLS.EXE`

options

Options

`-help` Lists the command line options.
`-port:nnnn` Specify a non-standard port that must be an even number.
`-tout:nn` Specify a non-standard server timeout. The default is 10 secs.
`-kill:pwd` Specify a password that allows remote user to stop (kill) the server.
`-logw:filename` Specify a filename to write an activity log to.
`-loga:filename` Specify a filename to append an activity log to.
`-logd:conn` Specify that connections are logged.

`-mst:master_ip[:port]` Specify that concurrent database access is used through the named master server.

`-tron` Specifies that commands are written to the consol and logged.

To interact with the server, press CTRL-C/CTRL-Break then enter a command:
`exit` Stops the server.
`limit n` Limits the maximum attached clients. Default 128.
`list` Lists the attached clients
`kill n` Stops the specified client connection.
`tron` Switches on the detailed command trace.
`troff` Switches off the detailed command trace.

The server can be stopped by a remote process if the `-kill:pwd` is specified. The remote process runs `sirrmt.exe` specifying the IP address of the server and the appropriate password. This stops the server. e.g.If you started the server as:

```
c:\Program Files\SIRXS>sirsqls -kill:exit
Starting SIR SQL server...
Host: Larry (192.168.111.85) Port: 3050
Starting SIR SQL engine (XS.01.01)...
Initialized
Press Ctrl+C to interrupt
```

Then you can remotely stop the server with:

```
sirrmt larry:3050 exit
```

If the server is not started with a kill password then you cannot remotely stop it.

PQLServer

The SIR/XS PQLserver is an executable that allows another standard SIR/XS session to connect as a client and to transmit commands to the server, execute those commands remotely and retrieve output. This is done with a set of PQL functions. The PQLServer must be started to enable clients to communicate to it across the network. The client processes do not require any access to files or databases that are local to the server and the two processes (client/server) may be using different hardware/operating systems e.g. client on windows, server on Unix.

From the client point of view processing is as follows:

- Client logs on to server and gets back a client id. The client id is used in all subsequent communication with the server. If the server is started with the UPASS parameter then the client must specify the server administration password to be allowed to log on.
- Client sends any number of lines of text that would include SIR commands. Control usually passes back without any actual transmission taking place - transmission only happens when maximum message size is reached.
- Client starts execution of previously sent commands. Any commands not yet sent to the server are transmitted, any settings or output from a previous execution from the same client are re-initialised, the commands are run and a completion code is returned at which point any output is waiting on the server. Commands can include all SIR commands and can use procedures, etc. Note that commands must include connecting any databases/tabfiles/procedure files needed each time commands are submitted and executed. There are no saved settings between executions. The process may read/write files, update databases and generally do anything that a batch run of SIR can do. While executing, no communication is happening with other possible clients. It is thus good practice to keep execution streams as short as possible.
- The client may choose to wait for the execution to finish or to carry on processing locally and subsequently test to find if the execution has completed successfully.
- Client gets count of number of lines of output and can then get each text line or skip over unwanted lines. Lines are physically passed by the server in groups. If skipping lines and the lines have not yet been transmitted, they are skipped on the server. Lines once returned or skipped are no longer available. The client can get a count of the number of lines available at any point.
- Client can repeat the process.
- Client logs off when finished.

e.g.

```
program
```

```
compute client = serlog (0,'TONYDELL:4000','')
write client
compute x = sersend (client,'PROGRAM')
compute x = sersend (client,'WRITE "HELLO WORLD"')
compute x = sersend (client,'END PROGRAM')
compute rc = serexec (client,1)
write 'rc = ' rc
compute olines = serlines(client)
write 'lines ' olines
for i=1,olines
. compute line = serget (client,0)
. write line
rof
compute client = serlog (client,'TONYDELL:4000','')
end program
```

Client Functions when using PQLServer

`SERADMIN` Various server administration capabilities (returning numeric values)

`SERADMIS` Various server administration capabilities (returning string values)

`SEREXEC` Instructs server to execute previously sent commands

`SERGET` Gets a line of output from server

`SERLINES` Asks server how many lines of output are left

`SERLOG` Logs on to the server

`SERSEND` Sends a string to the server

`SERSENDB` Sends a buffer to the server

`SERTEST` Asks server if execution has completed

PQLServer Functions

(These have no effect if used in a program that is not running on the server)
`SERNOOUT` Suppresses server output

`SERWRITE` Writes a line of output from server

PQLServer commands

```
CLEAR SERVER NOOUTPUT
SET SERVER NOOUTPUT
```

These can be used anywhere in the command stream to selectively control what output is made available for retrieval by the client. SET means that any output directed to standard output is thrown away. If this command is within a PQL program, it affects any compilation listing. Use the `SERNOOUT(n)` function for execution time control.

Running the PQLServer

The PQLserver must be running somewhere on the network to support any VisualPQL client processing. Start the PQLserver by running `PQLServer.exe`. This starts the server at the default port of 4000 and, in a text window, reports the current IP address. There are a small number of command line options:

```
PQLServer.EXE
```

options

Options

`SERV = nnnn` Specifies a non-standard port that must be an even number - 4000 is standard.
`PASSWORD = NAME` Specifies a password that allows remote user to administer the server.
`LOG` Specifies that transmissions are logged to `sirserver.log`.
`UPASS` Specifies that only users supplying the password given by `PASSWORD=...` can log on.

Controlling server locally

To interact with the server, press CTRL-C/CTRL-Break then enter a command:
`help` Lists these commands.
`exit` Stops the server immediately.
`nologon` Stops the server when all clients have logged off. Does not allow any new logins.
`logon` Reverses a previous nologon and allows normal processing.
`list` Lists various statistics and the attached clients.
`kill n` Stops the specified client connection.
Note that the server is not responding to client messages while interacting locally in command mode. Press Enter to allow server continue with normal processing.

Message Log

Messages are all in plain text and have following format:
bytes 1- 4 - overall transmission length in dwrds up to 512
bytes 5- 8 - client id
bytes 9- 12 - message length in dwrds
bytes 13-16 - message type
0 - Logon
1 - SIR/XS Commands / Output lines
2 - Execute

3 - Send output lines
4 - Logoff
5 - Admin
dwrd 3 - n - message text (if applicable)
followed by second message in same format (if applicable) until overall transmission length met.

Error Messages

The following is a list of the numbered error messages that may be issued by `SIR/XS` .

```
Error Messages

Code Meaning
---- ------------------------------------------------------------
0001 Command complete but not at end of line
     Extra information found while looking
     for the beginning of the next command.
0002 Option is invalid
0003 Number is out of valid range.
0004 Keyword is invalid.
0005 Filename is invalid.
0006 Maximum record count for record type exceeded.
0007 Statement is invalid at this point.
0008 Numeric syntax is invalid.
0009 Option is not currently in operation.
0010 Variable is not a program variable.
0011 KEYFIELDS specification is invalid.
0012 Value List not valid.
0013 Unknown Command.
0014 Tabfile name is required.
0015 Password is invalid.
0016 Record number or name is invalid.
0017 Record number exceeds max rec types in the schema.
0018 Group/user name or password is invalid.
0019 Preset variable with wrong type value.
0020 Table specification is invalid.
0021 Sort records statement is incomplete.
0022 Tabfile delimiter "." is missing.
0023 Case ID size exceeds maximum.
0024 Table already exists.
     A table of with the same name already exists and
     the REPLACE option is not specified.
0025 Variable list statement is incomplete.
0026 Duplicate name in variable list.
     The same variable name has appeared more than once
     in a list of variable names.
0027 Invalid READ or WRITE operator or format specifier.
     The letter is not a valid operator or
     format for the READ or WRITE statement.
0028 "TO list" is invalid.
     The names before and after the keyword TO must
     either be previously declared variable names or
     names of the form AAANNN, where AAA are alphabetic
     characters and NNN are digits, and NNN of the right
     side is greater than NNN of the left side.
     Only variables that reside in the same record
     (either CIR or a given RECORD TYPE or a given
     TABLE) may be used in a "TO list".
```

```
0029 Input format syntax error.
0030 Right parenthesis ")" missing from command line.
0031 Format is invalid.
0032 Name is missing or invalid.
     Valid names are up to 32 characters long
     and are recognised as follows:
     (a) A letter or one of four special character $,#,@,_
     followed by letters, digits, or the four special characters.
     All lowercase characters are translated to uppercase.
     (b) Any character sequence enclosed in curly
     brackets {}. The character sequence may be up to 30 chs.
0033 Variable is undefined.
0034 Cannot write to tabfile open for READ.
0035 Date/time map is invalid.
0036 Tabfile previously open for a different GROUP/USER NAME.
0037 Scale factor is invalid.
0038 Format requested conflicts with variable type.
0039 Variable range is invalid.
0040 Missing value is invalid.
0041 VARIABLE LABEL is invalid.
0042 VALUE LABEL is invalid.
0043 You do not have permission to add tables to a tabfile.
0044 COMPUTE statement requires an equal sign "=".
0045 VALUE LABELS list is incomplete.
     A variable or a list of variables is not
     followed by a list of value labels.
0046 JOURNAL option is invalid.
     Journalling is an attribute of the database.
     Database file number 5 is the default journal file.
     JOURNAL clauses for batch data input, retrieval,
     and SIR MERGE are not valid.
0047 "TO list" out of order.
0048 Invalid "TO list".
0049 Multiple assignment statement is invalid.
0050 All variables in "TO list" must be of the same type.
0051 Existing VALUE LABEL conflicts with new VALID VALUES.
0052 Cannot extend CODEBOOK file.
0053 Variable type changed.  VALUE LABELS deleted.
0054 Existing VALUE LABEL conflicts with new RANGES.
0055 "TO list" subscript must be numeric.
0056 Option is invalid for variable.
0057 Cannot change attribute of CASE ID after data entered.
0058 Cannot open database journal file.
0059 Command valid only during data dictionary modification.
0060 PAGE SIZE is invalid.
     For the standard output file, the page width must
     be between 50 and 254, and the page length at least
     10 lines. The keyword NOEJECT may be used to turn
     off paging.
0061 Invalid record security code.
     DBA security has been assigned to the record.
0062 Variable security list is invalid.
0063 Unknown command.
0064 "TO list" has undefined variables on right side.
0065 Data file is empty or uninitialised.
0066 Variable is already declared as temporary.
0067 Cannot open database.
```

```
0068 Common/temporary name conflict.
0069 Only DBA may set record access levels.
0070 String length is invalid - 32 assumed.
0071 End quote mark (') missing from string.
0072 Character is invalid at this point.
0073 Variable name is invalid.
0074 Function is unknown.
0075 Required index name is missing.
     This command requires an index name.
0076 Wrong type of variable in function call.
0077 Required TABLE name is missing.
     This command requires a table name.
0078 Wrong number of arguments in function reference.
0079 Mismatched parentheses "( )".
0080 Equal sign "=" is in an invalid position.
0081 Can only assign values of same type.
0082 CASE ID or KEYFIELDS is missing from the data line.
0083 Expecting 'TO' in expression
0084 Expression is invalid.
0085 Invalid redefinition of a variable.
0086 New COMMON VARS defined, Database unload/reload required.
0087 Common variable type specification is invalid.
0088 Tabfile is not connected.
     An operation on a tabfile failed because the named
     or assumed tabfile is not connected.
0089 Index is not in tabfile.
0090 CASE ID or KEYFIELDS are not defined in the record.
0091 Table is already in use.
     The SAVE TABLE procedure cannot create or
     replace a table which is already used during
     retrieval or by another SAVE TABLE procedure.
0092 Larger CIR is required for this record type.
0094 Total name not found in PRINT list.
0095 Error in list specification.
     A list of variable names, values, keywords:
     - is not terminated properly; or
     - contains invalid characters, names, keywords; or
     - is empty.
0096 List of variables is missing.
0098 Option ignored for saved executables.
     The option is used only when running
     programs or retrievals and is ignored when
     saving executables.
0099 Break name on SUBTOTAL clause not found.
0100 Column name on SUBTOTAL clause not found.
0102 Error in sort package
     A call to a routine in the sort returned an error.
     Possibly a physical problem with disk or
     a file size limit exceeded.
0103 Cannot tab backwards in PRINT clause.
0104 Error in index value list.
     The number of values exceeds the number
     of variables in the index, or
     the value or variable are of incompatible
     type - one of type string, the other of type
     numeric.
0105 Index is not unique.
```

```
     Warning when the index used on a
     (OLD/NEW) ROW IS statement is not unique.
0106 Invalid use of standard file.
     A reserved file used incorrectly. For example,
     reading from the standard output file or a random
     write to the standard output file.
0107 Expected character not found in READ statement.
0108 FILENAME specification is invalid.
0109 Statement label clause is invalid.
0110 READ statement is invalid.
0111 Data type is incompatible with format.
0112 Invalid value on COUNT clause.
     A number must be a numeric constant greater than
     zero.
0113 Number of cases has been redefined. Reload Database.
0114 Records per case has been redefined. Reload Database.
0115 Record type columns have been redefined.
0116 Cannot define locally for a Standard Variable.
0117 You cannot decrease maximum record types.
     A record type exists that is greater than the maximum
     specified. Either delete the record type(s) or
     specify a larger maximum value.
0118 Maximum record types increased. Reload Database.
0119 No variables to move.
     The GET VARS or PUT VARS command has
     an empty variable list or uses ALL outside a
     PROCESS CASE/CASE IS, PROCESS RECS/REC IS, or
     PROCESS ROWS/ROW IS block.
0120 List of index values is incomplete.
     The number of index values in a (OLD/NEW)
     ROW IS statement is not equal to the number of
     variables in the index.
0121 PUT VARS can only be used in UPDATE mode.
     The database or the tabfile has not been opened
     for update.
0122 The KEYFIELD is invalid.
0123 Input format must follow a variable list statement.
0124 Slash "/" missing where expected.
0125 Numeric/string constant syntax error.
0126 Error in processing VALUE LABELS.
     An error occurred while inserting value labels.
     Possible reasons are:
     (1) Valid ranges exist and value is not in range.
     (2) Value is in missing range.
     (3) String is too long and is truncated.
0127 Format exceeds MAX INPUT COLS.
0128 Variable is not defined in record.
0129 Maximum number of records in database is too large.
     NO OF CASES multiplied by RECS PER CASE is the
     upper limit of records in the database. This number
     cannot exceed 2,147,483,648.
     Make sure that NO OF CASES and RECS PER CASE are
     reasonable values and their product does not exceed
     the maximum.
0130 Page width is too narrow for the generated report.
     The REPORT generated requires a larger page width
     than specified in the PAGESIZE-clause (or assumed
```

```
     by default). The amount needed is displayed in the
     error message, next to the error number.
0131 You cannot total/subtotal strings/dates/times/cat vars.
0132 DATE STRING syntax is invalid.
0133 Command valid in batch mode only.
     The command is ignored because it is
     invalid during an interactive run.
0134 Table name cannot be the same as the tabfile name.
0135 Unexpected end of data while reading a data record.
0136 Invalid record type field on data line.
0137 Data line sequence number out of order.
0138 Write security not granted for variable.
0139 Error reading numeric data.
0140 Value is outside valid range.
0141 Security not granted to write record.
0142 Record to be updated does not exist.
0143 New record in REPLACE ONLY mode.
0144 Old record in ADD ONLY mode.
0145 CASE ID or KEYFIELD missing is record to be updated.
0146 Category not found for variable.
0147 Default date/time MAP assigned. (warning)
0148 CASE ID or KEYFIELD is not defined in variable list.
0149 Invalid option.
0150 Quote mark (") missing where expected.
0151 CASE ID/KEYFIELD invalid on READ statement.
0152 Invalid or duplicate INCLUDE/EXCLUDE list.
0153 Excluded variable missing.
0154 Included variable missing.
0155 TOTAL list specified with SUPPRESS TOTALS.
0156 KEYWORD not valid with this command.
0157 Only DBA can modify access levels.
0158 Cannot mix Visual PQL and Execution window commands
0159 Data list item is invalid.
0160 Line number is invalid.
0161 Column number is invalid.
0162 Type is invalid.
0163 Columns do not divide evenly among variables.
0164 Field is too small for assumed decimal point.
0165 Input columns for variable overlap with another var.
0166 Record type does not match on every line of record.
0167 CASE ID/KEYFIELD does not match on every line of record.
0168 Value read is not in valid value list.
0169 Only DBA can use this command.
0170 No permission granted to read variable.
0171 Update cancelled due to previous errors.
0172 Case not found.
0173 Only DBA can change JOURNAL ON/OFF setting.
0174 Error in I/O status variable specification.
     The I/O status variable must be a program variable
     and it must be numeric.
0176 The value list in the PROCESS statement is invalid.
0177 Index range is invalid.
     INDEXED BY 'indexname' is not followed by a
     recognisable keyword.
0178 Redefinition of record schema - rectype locked.
     Run UNLOAD/RELOAD.
0179 Integer list item is invalid.
```

0182 Equal sign "=" missing where expected.
0183 Wrong type of file for this command/option.
0184 Missing quote mark (") at beginning of string variable.
0185 Maximum number of input characters exceeded for rec type.
0186 String variable is not allowed in this context.
0187 Sample specification is missing.
0188 Syntax error in sample specification.
0189 Invalid sample number. Use a number between 0 and 1.
0190 Character is invalid.
0191 Syntax error in statistics keyword.
0192 Read statement list exceeds available input columns.
0193 Output statement list exceeds available output columns.
0194 Invalid variable used as weight specification.
0195 File previously declared a different type.
0196 Duplicate option or keyword.
0197 Procedure specification is incomplete.
0198 Undefined, missing or unknown tabfile.
0199 Left parenthesis "(" missing where expected.
0200 Max record count for case exceeded - record skipped.
     This results in an error. Data are lost.
0201 Variable for filename spec must be a string variable.
0202 A key field has a missing value - record skipped.
0203 Record exists but NEW rec indicated on UPLOAD file.
0204 Record does not exists, but new or delete rec attempted.
0205 Bad definition value on upload file.
0206 Unexpected end of file.
0207 Loading factor must be a number between 0 and 1.
0208 Format specification missing.
0209 No filename.
     The FILENAME clause is missing and/or a default
     filename could not be constructed.
0210 Invalid record type value on upload record.
0211 Left parenthesis "(" missing where expected.
0212 Invalid format - does not specify any data.
0213 Record type found in download that was not selected.
0214 Update code on upload file is invalid.
0215 Bad variable number on upload file.
0216 Variable name or type does not match.
0217 Syntax error in limit specification.
0218 Keyword WITH is missing from command.
0219 Only one independent variable.
0220 Only one dependent variable allowed if control var used.
0221 Too many dependent variables.
0222 Keyword BY missing from command.
0223 DIVISION parameter value is invalid.
0224 Unknown device name entered.
0225 Device specification is invalid.
0226 Size exceeds device limit.
0227 Incorrect number of symbols.
0228 Data on UPLOAD file is invalid.
0229 Only one control variable allowed per chart/plot.
0230 Label field not allowed in current context.
0231 Statement allowed in RETRIEVAL/PROGRAM UPDATE only.
0232 Grouped BREAK VARS with SUPPRESS GROUPING.
0233 PRINT list is required for generated report.
0234 Unknown label referenced in command.
0235 Column heading with SUPPRESS COLHEAD.

```
0236 Invalid to add/delete rows in table being processed.
     It is invalid to add new rows to a table or to
     delete rows from a table while this table is being
     processed in an outer loop. Nested processing of
     tables can only be in read-only mode.
0237 LABEL is invalid.
0238 Attempted to modify var when CASE/RECORD/ROW invalid.
     Restore the record command.
0239 Attempted to access var when CASE/RECORD/ROW invalid.
     Restore the record command.
0240 Missing value is invalid for CASE ID/KEYFIELDS.
0241 Data file is full, download stopped.
     Do an UNLOAD/RELOAD when data file is full.
0244 CASE ID/KEYFIELDS not within valid range or list.
0245 Common variable previously specified as a non-integer.
0246 User interrupt - download stopped.
0247 Invalid value  on ERROR/WARNING limit command.
0248 Sort list with SUPPRESS SORT.
0249 Record added to a case that has been deleted.
0250 Table previously in non-update mode requires update.
0251 Can't merge caseless database into case structured db.
0252 Subfile variable must be a numeric program variable.
0253 Sort order specification is invalid.
0254 Invalid in batch mode - retrieval stopped.
0255 Invalid key field reference in UPLOAD file.
0256 Previous value labels ignored due to type change of var.
0257 Categorical/missing value not defined.
0258 Variables moved do not match in type.
     A CIR, RECORD, or ROW variable moved using GET/PUT
     VARS has a different type than the corresponding
     program variable.
0259 Invalid KEYFIELDS value on PROCESS REC/REC IS statement.
0260 Security not granted to read record.
0261 "DO REPEAT" cannot appear inside another "DO REPEAT".
0262 Invalid array index - Undefined value returned.
0263 Invalid array index - not executed.
0264 Duplicate variable.
0265 Maximum temporary variables exceeded.
0266 You can't declare existing record variable as an integer.
0267 VARLAB/VALLAB requires a program variable as an argument.
0268 Variable in list is not a program variable.
0269 Valid value not previously defined.
0270 Program variables cannot be modified by procedures.
0271 Too many files referenced in this run.
0272 Invalid input/output mode specification.
0273 Only one decimal point allowed in picture
0274 Invalid sequence of special characters in picture.
0275 Index variable missing.
0276 Error writing to the PROC file.
0277 Source database name missing.
0278 Cannot reopen tabfile with a table open.
0279 Invalid TABFILE CONNECT security, trying PUBLIC.
0280 Invalid set of keyword options.
0281 Variable must be numeric.
0282 Cannot close tabfile with a table open.
0283 RECODE list is invalid.
0284 No variables to be recoded.
```

```
0285 You cannot mix data types in a RECODE statement.
0286 Invalid BREAK LEVEL NUMBER/NAME.
0287 Previous BREAK LEVEL has no node to come from.
0288 Variable is not summary variable
0289 Subtotal list with SUPPRESS SUBTOTAL.
0290 SUBTOTAL LIST without any BREAK LIST.
0291 Cannot close table from within a PROCESS ROW block.
0292 Invalid pseudo variable.
0293 Substitution value is invalid.
0294 Invalid range for substitution value.
0295 Invalid BUFFER line number specification.
0296 Invalid "DO REPEAT" termination.
0297 Invalid BUFFER line range specification.
0298 PURGE SIR FILE invalid with errors in the current run.
0299 Security not granted to delete record or case.
0300 Mismatched parentheses "()" on HEADING/FOOTING statement.
0301 Error opening file during retrieval execution.
0302 End of data reached while reading a file.
0303 Undefined record type.
0304 READ/WRITE attempted after END-OF-FILE or I/O error.
     Attempted operation on a file after either an
     end-of-file condition or an error found.
0305 Record type already specified.
0306 Error in COUNT specification.
0307 More than one CASE SELECT option.
0308 File opened in the wrong mode for this operation.
0309 New data type is incompatible with previous definition.
0310 Old variables can not be modified.
0311 New variables can not be defined.
0312 Record name matches RECORD SCHEMA command keyword.
0313 Database name is invalid.
0314 Record type name must be appear on RECORD SCHEMA command.
0315 CASE ID variable invalid.
0316 Record type lock required, but permission is not given.
0317 Record type is invalid.
0318 SIR SAVE FILE specification is invalid.
0319 Too many sort files - only 9999 are allowed.
0320 READ/WRITE error or END-OF-FILE condition.
     A read/write error or end-of-file condition
     occurred and no error processing set. The
     status variable (IOSTAT= varname) or error branch
     (ERROR= label) may be used to process I/O errors in
     a retrieval.
0321 CASE ID variable is not a program variable.
0322 Record type missing.
0323 KEYFIELDS variable matches CASE ID.
0324 Data exceeded record length.
     The data actually entered in the record exceeds
     what the record length allows for. The data will
     not be inserted.
0325 Command invalid without defined schema.
0326 Syntax error in array list.
0327 Invalid dimension of array.
0328 Wrong number of subscripts in array reference.
     The number of subscripts referenced in this
     expression disagrees with the number of subscripts
     on the array statement.
```

```
0329 Invalid variable or array reference.
     In a context where only a variable or array
     reference is expected, unexpected characters are
     detected.
0330 Array references must be followed by a "(".
0331 Subscript expression contains a missing result.
     A subscript expression for an array reference
     contains a missing result. If the array element is
     being loaded, its value will be "undefined". If the
     array element is being stored into, the store
     operation will be skipped.
0332 Subscript value violates array bounds.
     A subscript value for an array reference is out of
     the range specified by the bounds on the array
     statement. If the array element is being loaded,
     its value will be "undefined". If the array element
     is being stored into, the store operation will
     be skipped.
0333 Format requested conflicts with expression type.
0334 Closing bracket is missing from expression.
0335 Database security violation.
0336 Source file header is missing.
     This error message could indicate a failure to
     locate the source database on the specified file.
     Check the database name on the DATABASE
     clause and/or run ITEMIZE FILE on the source file.
0337 Inconsistent/duplicate keyword/option.
     A keyword or option has been specified twice or two
     options are inconsistent with each other.
0338 Error reading source file.
0339 Password incorrect for source file.
0340 Source database security violation.
0341 Array cannot be used in this context.
     An array reference is not permitted here. Only a
     simple variable may be referenced.
0342 Name required after AS keyword.
0343 Security specified is invalid.
0344 Subscripted expression used on non-array variable.
     A variable is followed by a "(" but the variable is
     not an array.
0345 An array destination variable requires a source variable.
0346 Record does not exist on source file.
0347 Unknown buffer name.
0348 Record type is invalid.
0349 UPDATE specification is invalid.
0350 Error in RENAME specification.
0351 Undefined template referenced.
0352 Record type in host file is not compatible.
0353 Specified edit buffer is empty.
0354 Source database name is incorrect.
0355 Statement not valid unless in DEBUG mode.
0356 Duplicate variable name after rename.
0357 Saved executable not is not in the proper format.
     When reading the saved executable or compiled subroutine,
     one of the control words or counts did not match.
0358 Merged Record type causes CIR record count to be exceeded
0359 Host MAX KEY SIZE is too small for merging this rec type.
```

```
0360 Data file is empty.
0361 HOST common/temporary variable matches source.
0362 KEYFIELDS cannot be temporary or common variables.
0363 Unallocated common variable referenced - rec type locked.
0364 Rec type number exceeds max rec types - rec type locked.
0365 Data record larger than data block - record type locked.
0366 Data record larger than recommended value.
0367 Record type key size larger than max - rec type locked.
0368 Keysize increased - database locked.
     Increasing the maximum keysize caused the database
     to be locked. An UNLOAD - RELOAD sequence is
     required to unlock the database.
0369 DBA security required to lock the database.
0370 Common var changed to REAL to match previous definition.
0371 Data file never initialised.
0372 Invalid syntax in BOOLEAN clause.
0373 VALUE LABELS cannot precede CAT VAR definition.
0374 Invalid DATE/TIME string.
0375 Record type locked - database reload required.
0376 Invalid keysize.
     The value for the maximum keysize is invalid. It
     must be greater than zero and less or equal the
     maximum keysize possible (320 characters).
     The new value for the maximum keysize cannot be
     less than the current value.
0377 Common variable not yet allocated to CIR.
0378 The saved executable is not for the current database.
     When reading the saved executable or compiled subroutine,
     the internally referenced database is not the current
     database.
0380 MAX REC COUNT has been increased too much.
0381 Only DBA can modify an existing schema.
0382 New maximum case value less than current number of cases.
0383 Record type name generated.
0384 Constant table is too large.
0385 Available storage space for variables exceeded.
     Too many variables were specified causing certain internal
     fields to overflow. Limit the number of variables in one routine
     to less than 4094 and try again. Sub-routines have separate counts
     and are each limited to 4094.
0386 Fewer variables than format specifies- remainder ignored.
     This warning denotes an inconsistency between the
     number of variables specified in the VARIABLE LIST
     command and the number of format specifications in
     the INPUT FORMAT statement: the format allows more
     variables to be read than there are in the variable
     list.
0387 No new cases allowed with NONEW option.
0388 More variables than format specifies - format will cycle.
0389 Maximum number of program variables exceeded.
0390 Invalid to modify current key variables
     Variables used to create the key for the current
     record (CIR and Keyfields) or row (variables used in
     the current index) cannot be modified in the
     program.
0391 No record types defined.
     In order to execute the command, at least one
```

```
     record type must be defined for the database.
0392 This command may only appear once in a PQL program.
0393 Expected character not found.
0394 Lower and upper range values are equal.
0396 No BREAK LEVEL statement found in report definition.
0397 Report specification incomplete (/PRINT may be missing).
     A report statement without a /PRINT clause is not
     followed by statements forming a report body.
     This usually indicates that a /PRINT clause is
     missing from the REPORT statement: only the presence
     of a /PRINT clause initiates the processing by the
     Simplified Report Procedure.
0398 Run terminated due to previous error(s).
0399 Security not granted to execute retrieval.
0400 Command is not valid when terminal in full screen mode.
0401 Invalid Schema Command for CIR or STANDARD SCHEMA.
0402 New filename is invalid.
0403 New password is invalid.
0404 Sort key size exceeds maximum.
     Procedures sort the Procedure Table.
     The combined length of the variables which form the
     sort key exceeds the allowed maximum 320.
     To solve the problem, reduce the
     number of variables or their length.
0405 Source rec size larger than host data block - can't merge
0406 Variable list limit for procedure exceeded.
0407 No variables were selected to be processed.
0408 Variable must be a string variable.
0409 Record/common variable modification in UPDATE mode only.
0410 No permission granted to modify variable.
0411 I/O error while writing to procedure file.
     The last command which involved writing to the
     Procedure File could not be completed. The problem
     with the disk file containing the Procedure File
     must be resolved before any further I/O on the
     Procedure File may be done.
0412 Line data cannot be read from a called member.
0413 Previous EXIT/NEXT statement(s) are not in a loop.
0414 Security not granted to write case.
0415 First line of CALL text is not a command.
0416 Error in external sort package.
0417 I/O error on file.
0418 Format exceeds record length.
0419 End of line, null name used.
0420 Zero length, quoted filename found.
0421 Filename is too long.
0422 Invalid quoted filename.
0423 Filename does not start with an alphabetic character.
0424 Duplicate statement label.
0425 VAR LABEL reference for common variable has been deleted.
0426 VALUE LABEL reference for common var has been deleted.
0427 Label truncated during schema re-creation.
     When the VAR or VALUE LABELS command is
     reproduced, the extended label text (i.e. with
     enclosing quotes, and doubled, single quotes) does
     not fit into a single line. The label has to be
     modified before it can be reproduced properly.
```

0428 Database in executable has creation date/time mismatch.
The creation date of the database internally referenced
in the saved executable or compiled subroutine does not
match the creation date and time of the current database.
0429 Error creating variable reference. Variable number >= 4096.
0430 Error creating variable reference. Table number >=8.
This is an internal error which should
never occur. Please run the VERIFY FILE utility. If
the error still occurs, send as much documentation
as possible, including the database (or a subset of
it) in EXPORT format, to SIR.
0431 Overlapping or out of order list items.
0432 Range ignored. Common variable has already been defined.
0433 VALID VALUES ignored. Common variable already defined.
0434 CAT VAR specification ignored. Common var already defined
0435 Label record exceeds maximum size, label ignored.
Only 32763 characters of label text is addressable
by SPSS. The variable labels and
value labels defined by the variables passed to the
SPSS system file exceed that limit. The labels for
this and all subsequent variables are ignored.
0436 Missing LDIVAR or DSNVAR variable, cannot open file.
The LDIVAR or DSNVAR variable contains a missing
value and not a valid file name; hence, the file
cannot be opened.
0437 Bad LDIVAR or DSNVAR file name, cannot open file.
The LDIVAR or DSNVAR variable contains a string
that is either zero length or greater than the
maximum filename; hence, the file cannot be opened.
0438 Can't load subroutine referencing a database from PROGRAM.
A subroutine that is called by a PROGRAM must use the
/NODATABASE parameter. Without this parameter a database
is referenced and the routine can only be called from a
RETRIEVAL.
0439 Cannot form member name.
A valid member name could not be formed.
0440 Member must be a text member.
0441 Family password mismatch.
0442 Member password mismatch.
0443 Cannot open proc file.
0444 Family not found.
0445 Member found but replace mode not specified.
The member is opened for write but the member
exists and REPLACE not specified.
0446 Member not found.
0447 Cannot open scratch file to process member.
0448 I/O error while initialising member.
0449 Proc file already open in this retrieval.
Only one member at a given time may be opened
during a retrieval.
0450 Null retrieval or null program.
No executable command found between the
RETRIEVAL (PROGRAM) statement and END RETRIEVAL
(PROGRAM), FINISH or end-of-file.
0451 Subroutine referenced could not be found.
0452 Variable in list previously defined.
0453 Statement not allowed on multi-statement "IF".

```
0454 Block structure statement is invalid in current context.
0455 Invalid PRESET statement.
0456 Variable must be a program variable.
0457 Unknown type for variable  - REAL assumed.
0458 REPEAT factor must be an integer.
0459 Invalid variable list.
0460 RESTORE CIR/REC is not in CASE/RECORD loop.
0461 Current nesting of blocks is invalid.
0462 Errors in attempting to load subroutine, load ignored.
0463 Feature not supported in this version.
0464 END assumed to terminate current block structure.
0465 Compiled member deleted due to schema changes.
0466 PAGE EJECT invalid in PAGE block of REPORT.
0467 The record you want to RESTORE has been deleted.
0468 RESTORE REC in case loop.
0469 PERFORM PROCS statement without procedures or vice versa.
0470 Statement must appear in PROCESS ROW/ROW IS block.
0471 Statement must appear in PROCESS CASE/CASE is block.
0472 Statement must appear in process REC/REC is block.
0473 Attempted to delete a non-existent case/record/row.
0474 Statement is not in proper loop structure.
0475 Invalid "FOR" statement syntax.
0476 INDEX variable must be a numeric program variable.
0477 You cannot change attributes of MOVED variable.
     Attributes of variables moved into the data buffer
     via GET VARS cannot be changed.  Define the
     attributes prior to the GET VARS command.
0478 Comments must begin in column one.
0479 Either the ROW or COL variable must be specified.
0480 Command is invalid in PROGRAM - RETRIEVAL must be used.
0481 Page is too short to print headings and footings.
0482 If no OBS var is specified then a FILENAME is required.
0483 Too few OBS variables to match other specified lists.
0484 Argument in "FOR" statement is not numeric.
0485 Only one "AFTER RETRIEVAL" block allowed.
0486 Command not allowed in REPORT/"AFTER RETRIEVAL" block.
0487 Invalid aggregation function of string variable.
0488 Cannot have more than one "BEFORE REPORT" block.
0489 Cannot have more than one "AFTER REPORT" block.
0490 Cannot have more than one GLOBAL "ON ERROR" block.
0491 Command invalid at this point in report definition.
0492 Duplicate report break level name.
0493 Variable length is invalid.
0494 Variable previously defined.
0495 Variable has no MISSING VALUES defined.
0496 CIR not found upon return to a higher level case loop.
0497 All procedures have been cancelled.
0498 Invalid position for PRESET/DATA statement.
0499 Errors in loading compiled member.
0500 Cannot mix string and numeric variables in operation.
0502 Graph must specify an OBServation variable.
0503 Invalid operator.
0504 Case structure not specified.
     This error occurs when one or more case schema
     definition commands had been processed but the case
     structure of the database remains undefined. This
     is probably due to a missing CASE IS or NO CASES
```

```
     statement. This statement should be inserted before
     any other case schema definition command.
0505 Redefinition of case structure.
     Once a record schema has been defined, the case
     structure of the database cannot be modified in any
     way. If the database is still empty (it contains no
     data), the easiest way around this problem is -
        write the complete schema to a file (WRITE SCHEMA)
        delete the database (PURGE SIR FILE)
        start with a new database
        read the schema created in step 1, modify it, and run it.
0506 Cannot change type of string variable with modify schema.
0507 Cannot decrease MAX INPUT COLS once record types defined.
0508 Invalid redefinition of common variable.
0509 RANGE/VALID VALUE error in COMPUTE/RECODE.
0510 Invalid variable type.
0511 Command/Option/Function/Request is invalid at this point.
0512 Data truncation error in COMPUTE/RECODE.
0513 KEY SIZE is greater than MAX KEY SIZE.
0514 Data truncation error.
0515 Write access to tabfile is required to create index.
0516 READ/WRITE access violation found in COMPUTE/RECODE.
0517 Command/option is invalid for sequential data file.
0518 Record to evict is not IN the database.
0519 INPUT columns overlap rec type columns.
0520 INPUT columns overlap SEQUENCE columns.
0521 Unequal list lengths.
0522 Error in RANGE specification.
0523 Maximum COMMON VARIABLES exceeded.
0524 Maximum record types allowed is 4095.
0525 Maximum number of KEYFIELDS allowed is 4095.
0526 Maximum number of RECORD variables allowed is 4095.
0527 Field width error in DATE/TIME map.
0528 Command or option is invalid in interactive mode.
0529 Record name is not unique across all record types.
0530 Source specification missing in CALL command.
0531 Member specification error in CALL command.
0532 Parameter specification error in CALL command.
0533 "DO REPEAT" repeat statement cannot contain a "call *".
0534 Error in MEMBER/FAMILY specification.
0537 Family security error.
0538 Member security error.
0539 The name of an array matches a function name.
0541 Command is invalid in "CALL" or "DO REPEAT" statements.
0542 Invalid BLOCKS value.
     The number specified for BLOCKS must be a positive
     integer.
0543 Member already exists.
0544 Only DBA can create a new family.
0545 Null utility not executed.
     The utility requested would not create any output
     - SIR FILE LIST for a database without a case structure
     and no CIR variables or with the NOCIR option.
0546 Only DBA can modify passwords.
0547 Invalid security passwords.
0548 Command is invalid for databases without case structure.
     The command  is invalid for databases without
```

```
     case structure.
0549 Option is invalid for databases without case structure.
0550 Member is not a text member.
0551 Variable created.
0552 Feature is no longer supported.
0553 Previously defined N OF CASES ignored.
     N OF CASES greater than 1 for a
     database without a case structure.  This may have
     been caused by an N OF CASES command preceding the
     NO CASE ID command or by defining NO CASE ID after
     the default N OF CASES already assumed.
     N OF CASES is set to 1.
0554 Record count field update error.
     A journal record was not applied because it would cause
     the MAX REC COUNT for that record type to be exceeded.
0555 New KEYFIELD is not in old definition.
0556 New KEYFIELD type differs from old definition.
0557 Variable is not defined in new definition.
0558 Variable type is different in new definition.
0559 Variable length shortened in new definition.
0560 Variable conversion is str/numeric or numeric/str.
0561 Function is invalid at this point.
0562 Statement requires an operand.
0563 Operator is invalid at this point.
0564 Undefined temporary variable.
0565 Request to delete non-empty case  - ignored.
     A CIR journal record to delete a case was
     in the journal file but the record count fields shows
     that the case still contained records.
     The request to delete the case is ignored.
0566 KEYFIELDS option is invalid.
0567 Cannot open file.
0568 Error in global variable definition.
0569 Record count mismatch - PATCHED.
     A CIR journal record contained
     record counts different from computed record counts.
     The record count fields were set to the correct
     numbers before the journal is applied to database.
0570 Function cannot be executed - no Execution Window.
0571 Invalid UPDATE LEVEL/TYPE of record read.
     While reading the binary file, an invalid update
     level or an invalid type of record is found.
     This indicates that there is something wrong with the
     file being examined and causes ITEMIZE to stop.
0572 Error in journal record header.
0573 Cannot find journal which matches update level and date/time
     The journal restore process looks for a set of updates
     which exactly match the current update level and update date
     and time of the database.
0574 I/O error on journal/unload file.
0575 Incomplete record on journal file.
0576 Invalid update level.
0587 Data file became full while creating new CIR.
0588 Data file became full while creating new record.
0589 Case or record limit exceeded.
0590 Data File master index overflow.
0591 Data File is full.
```

```
     The data file is full and cannot be extended.
     Check the disk space allocation for this
     file or for your account.  Make sure that there is
     sufficient disk space available to continue then run
     VERIFY FILE.  VERIFY FILE will attempt to recover.
0592 Database is locked - reload required to unlock record.
0593 Only DBA can use this option.
0595 User interrupt.
0596 Invalid to set MAX INPUT COLS below REC TYPE COLS.
0597 Invalid to set MAX INPUT COLS below SEQUENCE COLS.
0598 Invalid to set MAX INPUT COLS below the system default.
0599 Sort file master index overflow.
0600 SEQUENCE COLS field is too small for some record types.
0601 RECTYPE COL value is too small for the record type.
0602 Premature End-of-Line detected.
0603 Invalid external file name.
0604 Invalid record length/block size.
0605 No current condition.
0606 Invalid conditional expression.
0607 Invalid string syntax.
0608 Conditional type mismatch.
0609 Previously defined variable.
0610 Cannot delete schema that has records in data file.
0611 SAS Blocksize is too small.
0612 User interrupt from utility.
0613 User interrupt from program.
0614 User interrupt from retrieval.
0615 Integer too large, variable type changed to R8.
0616 Out-of-date compiled RETRIEVAL/PROGRAM/SUBROUTINE.
0617 Current procedure type cannot be saved.
0618 Invalid pseudo-variable.
0619 Pseudo-variable already exists.
0620 Invalid number of quantiles.
0621 Invalid value for option.
0622 Cell contents error.
0623 Invalid variable/reserved word in expression found.
0624 Invalid MASK option.
0625 Invalid DECIMAL PLACES option.
0626 I/O failure for binary file read/write
0627 Invalid cell modifier.
0628 Invalid context for cell modifier.
0629 Too many table structure expressions.
0630 Operand invalid.
0631 Invalid expression.
0632 Chunk formatting error.
0633 WAFER/STUB output formatting error.
0634 Row/column variable without range.
0635 Invalid redefinition of a record variable as common.
0636 Invalid name for index.
0637 Index not found.
0638 All indexed fields required for spreadsheet access.
0639 All keyfields must be specified for spreadsheet access.
0652 No permission granted for required operation.
0697 If specifying one control var, must be ROW.
     If specifying two control vars must be ROW & COL.
0701 Contents of member does not match name/type of member.
0703 Called subroutine with wrong number of input arguments.
```

0704 Called subroutine with wrong number of output arguments.
0705 External common areas cannot INCLUDE other areas.
0706 Input argument type mismatch on subroutine call.
The type of the value passed as a subroutine input argument does not match the type declared in the subroutine.
0707 Type mismatch of subroutine output (returning) argument.
The type of the value passed as a subroutine output argument does not match the type declared in the subroutine.
0708 RETURN statement is not valid in a RETRIEVAL or PROGRAM.
0710 Missing END EXTERNAL VARIABLE AREA statement assumed.
0711 Cannot find specified EXTERNAL VARIABLE BLOCK to load.
0712 Missing or invalid window size declaration.
0713 Invalid External Variable Block. Possibly old version.
An invalid EXTERNAL VARIABLE BLOCK was found during loading. The block may need to be re-compiled.
0714 Duplicate EXTERNAL VARIABLE BLOCK definition.
0715 EXTERNAL VARIABLE BLOCK variable is currently defined.
0716 TABFILE referenced in subroutine has wrong timestamp.
0717 EXTERNAL VARIABLE BLOCK password mismatch.
0718 EXTERNAL VARIABLE BLOCK creation timestamp mismatch.
0720 Expected keyword missing.
0721 Required fixed position parameter is missing.
0722 Error while attempting to copy a random file.
0723 Output file exist, but REPLACE option not specified.
0724 Variable specified is the wrong type.
0725 Input and output files are the same.
0726 Mismatch between tabfile used at execution & compile time
The tabfile used at execution time must match the name and creation date of the tabfile used at compile time.
0727 Mismatch between Table used at execution & compile time.
The table used at execution time must match name and creation date of the table used at compile time.
0728 Mismatch between Index used at execution and compile time
The Index used at execution time must match name and creation date of index used at compile time.
0729 Cannot copy file to a SIR created file.
The destination file in a SIR FILE COPY command cannot be one of the database files.
0730 Missing subprocedure definition.
0731 END SUBPROCEDURE missing.
0732 Statements not allowed after END SUBPROCEDURE statement.
0733 Duplicate SUBPROCEDURE definition.
0734 Statement allowed only within a SUBPROCEDURE.
0735 END PROCEDURE when not in a PROCEDURE
0736 String value required.
0737 Cannot store into variable because of incorrect lock.
0738 Cannot store into variable because record/cir is locked.
0739 Invalid lock specification.
0740 Lock expression must be numeric.
0741 Command not valid when using Master.
0742 Only a constant or variable name allowed.
0743 Record processed by EXPORT is locked.
0744 Case processed by EXPORT is locked.
0745 X,Y value specified is missing/undefined/out of range.
0747 Subroutine updates database - caller not in UPDATE mode.

```
0748 Invalid or missing EDIT buffer name.
0749 Line number missing or invalid.
0750 Specified line to process is missing.
0751 Edit buffer currently exists.
0752 Edit buffer is unknown.
0753 SREAD/NREAD - No columns left on line to perform read.
     The prompt for the read fills the line, therefore read
     cannot be performed on the current line.
0754 Command needs more parameters
0755 A block mode request for non-block mode machine.
     The BLOCKMODE keyword should be specified on the
     RETRIEVAL/PROGRAM/SUBROUTINE command to enable block mode
     commands.
0756 A non-block mode request for block mode machine.
0757 No menu to process
0758 Unknown stack command code
0759 Page width altered to fit table
0760 Page length altered to fit table
0761 Item Id required
0762 No menu to end
0763 Cannot define a window at this point
0764 No window or menu for title
0765 No dialog for title
0766 Cannot define menu item at this point
0767 Cannot define dialog item at this point
0768 Cannot define message processing block at this point
0769 Invalid type of message processing block
0770 No dialog for message processing block
0771 EXECUTE DBMS cannot be used in CASE/RECORD/ROW blocks
0772 Tabfile restore failed
0773 Can only use FORMS commands in a FORM
0774 Must have screen name
0775 END screen command missing
0776 Duplicate screen name
0777 Cannot define a screen at this point
0778 FIELD must have name
0779 Cannot have key field on screen more than once
0780 Must have all key fields on screen
0781 Command only valid in SCREEN block
0782 CALL SCREEN ... AUTO cannot be first command on screen
0783 PQL CONNECT DATABASE cannot be used in CASE/RECORD blocks
0784 Duplicate value label value
0785 No variables selected
0786 No data sources selected
0787 Could not find selected data source
0788 Could not find selected variable
0789 Cannot specify key for first data file
0790 Specified keys for data files must be in sequence
0791 Specified keys for data files not yet defined
     Key values on the DATA FILES command need to be defined
     either in COMMON VARS or in RECORD definitions
0792 Record schema not defined - can't modify variables
0793 Global substitution too long for input - global truncated
     Input length restricted to 254 charcaters
2000 Bad select value in Host routine
2001 Specified name for database does not match
2002 Invalid password for database
```

2003 Password needed to attach database
2004 No CIRs/records found within specified restrictions
2005 No more CIRs/records within specified restrictions
2006 Maximum record count exceeded for current CIR
2008 Data file empty
2009 Invalid date string
2010 Invalid time string
2011 No. of days outside valid date range
2012 Data file is full
2013 In use flag set - run VERIFY utility
2027 Data file is empty after last delete
2028 Retrieval update mode required to add new CIR/record
2029 Specified record type locked or undefined
2030 Maximum case limit exceeded - restructure
2031 Data file master index overflow - restructure
2032 Attempted to position beyond end of data file
2033 Database previously opened but not closed
2034 Retrieval update mode required to delete case/record
2040 Cannot open Journal file
2041 Database not initialized
2042 Database closed - must reopen
2043 Invalid Database Reference
2044 Cannot locate update level on reload file
2045 Cannot close database currently used by some stream
2046 Cannot open reload input file
2047 Invalid HOST password specified to use database
2048 Must be DBA to Modify Schema
2049 Invalid lock flag specified
2053 Database already exists
2054 Update level overflow - unload and reload the database
2055 In use by other process
2056 Not found on specified disk
2057 Access forbidden
2058 Problem with open routine
2059 Bad Master Index - must rebuild database
2060 Not created by current version of SIR - export & import
2061 Too many data files defined for index
3001 Invalid user number
3002 Invalid sequence of function calls
3003 Security can only be set prior to any other calls
3004 HOST not properly initialized prior to call
3005 Count total value is invalid
3006 Count increment value is invalid
3007 Count start value is invalid
3008 Function call must be in an active case/record block
3009 Next is invalid for case/record is block
3010 Exit is only valid in an active case/record block
3011 Delete is only valid in an active case/record block
3012 Write is only valid in an active case/record block
3014 Key type already defined in current block
3015 Invalid order of key creation routines
3016 Attempted to create too many key fields
3017 Too many values entered for current key
3018 Numeric value specified for character key
3019 Character value specified for numeric key
3020 Value outside valid range for numeric key
3022 Security level insufficient to read record/CIR

```
3023 Security level insufficient to write/delete record/CIR
3024 CIR/record already exists in database
3025 CIR/record not found in database
3026 Incompatible lock exists on specified CIR/record
3027 Specified CIR/record is available for read only
3028 Current LOCK type is not sufficient to MODIFY/WRITE data
3029 CIR/record must be locked exclusively to delete
3030 CIR/record must be locked to write
3031 Deleted CIR/record cannot be restored
3032 Specified record name not found
3034 Case operation performed on record block
3035 Record operation performed on case block
3036 Deleted CIR/record cannot be written to database
3038 Record processing must be within a case block
3039 Cannot add or delete records unless the case is locked
3043 Current LOCK type is not sufficient to READ data
3044 DBA read and write security required to delete case
3045 Sample value must be between 0 and 1
5001 Attempted to transfer a string to a numeric variable
5002 Attempted to transfer a number to a string variable
5003 Security level insufficient to read variable
5004 Security level insufficient to write variable
5005 Undefined value transfered
5006 Missing value 1 transfered
5007 Missing value 2 transfered
5008 Missing value 3 transfered
5009 Specified variable name not found in record/CIR
5010 Specified record type/database not currently in stack
5011 Specified record type/CIR not currently valid
5012 Only integer lengths of 1,2 and 4 allowed
5013 Only real lengths of 4 and 8 allowed
5014 Maximum string length of 4094 allowed
5015 Invalid variable number in descriptor
5016 Invalid level specifier
5017 Item of length 'n' must be on an 'n' byte boundary
5018 Blank is not a defined missing value for variable
5019 Key variable cannot be modified at current time
7001 Cannot open tabfile
7002 Cannot read from tabfile
7003 Cannot write to tabfile
7004 Cannot create a table with the same name as the tabfile
7005 Cannot open Journal file
7006 Cannot read from Journal file
7007 Cannot write to Journal file
7008 Tabfile is already connected, cannot reopen or create
7009 Tabfile is not initially empty
7010 Specified file is not a Journal file
7011 Attempted to add duplicate entry to unique
7012 File currently open for read only access
7013 File is not a SIR tabfile
7014 Attempted to create a duplicate index
7015 No permission granted to create table
7016 Table currently exists on tabfile
7017 No permission granted for requested operation
7018 Table does not exist
7019 Table currently in use for a non-compatible mode
7021 Cannot drop internal table/index
```

```
7022 Required index on system table is missing
7023 Tabfile is corrupt run VERIFY TABFILE utility
7024 Group/User name unknown for tabfile/table
7025 New Journal file must be empty
7026 Row previously dropped
7027 No more rows found for request
7028 Row exceeds maximum size for tabfile
7029 Numeric request for string column
7030 String request for numeric column
7031 Integer overflow converting real to integer
7032 Column name not found in table
7033 Index already open in incompatible mode
7034 Index not open
7036 Tabfile index required more than 6 levels
7037 JOURNAL/BACKUP file does not contain a valid header
7038 Null values are not allowed for specified column
7039 Invalid blocksize value
7040 Specified table number is invalid or table is not open
7041 Invalid column name or number of columns
7042 Numeric argument out of valid range
7043 Inconsistence between field widths of a real number
7044 Invalid setting for string column
7046 Arguments for routine are invalid at this time
7047 Invalid missing/valid range number or too many defined
7048 Valid and missing ranges should be disjunct
7049 Tabfile is unknown or closed
7050 Column already exists
7051 A table should have at least one column defined
7052 Categorical variable cannot have range
7053 Journal restore - tabfile already exists
7054 Tabfile creation date/time differs from Journal
7055 Journal file applies to a later version of the tabfile
7056 Journal file ended without tabfile close entry
7057 Expected journal entry to open tabfile not found
7058 Table creation date/time differs from Journal
7059 Attempted to open table for closed tabfile
7060 Invalid data while trying to add row
7061 Operation not valid for VIEW
7062 Excess information at end of Journal record
7063 Tabfile not found on Backup
7064 Full dump cannot be restored in pieces
7065 Cannot restore full backup to existing tabfile
7066 Block number referenced but not in original list
7067 Cannot write to backup file
9001 Master could not attach to new path
9002 Master could not Re-attach to old path
9003 Master could not delete an old path
9004 Master cannot open database datafile
9005 Master error deleting a complete case
9006 Error releasing case and/or lower record locks
9007 Master CIR rewrite failure
9008 Master CIR lock release failure
9009 Master CIR lock convert failure
9010 Master logout failed
9011 Master database data file close failure
9012 Master database data function failed
9013 Master record deletion failed
```

```
9014 Record lock release failure
9015 Master record rewrite failure
9016 Record lock convert failed
9017 Could not send message to Master
9018 Error opening current process input channel
9019 Error opening Master message echo file
9020 Cannot establish link with Master
9030 Master database record access failed
9031 Cannot establish required record type lock
9032 Cannot copy difference file
9033 Master closing - cannot accept logins
9034 Master shutdown
9035 Master password did not match
9036 Exceeded number of licensed users
```

SIR/XS Files

The system uses various files predetermined files listed below:

```
    sirforms.err   Forms error output list
    sirsql.fmt     SQL messages file (binary)
    sir.ini        Initialisation file in the user's My Documents
directory.
                   Under Unix the file .sir is in the user's home
directory.
    sirforms.kmp   Forms default keymapping (binary)
    sir.lcn        Licence Control File
    sirout.lis     Output from a SIR/XS batch run
    sirsql.slg     SQL session log file (text)
    sirproc.srp    System Procedure File
    sirsql.wsp     SQL workspace file (binary)
```

The system also uses various extensions as standard as listed below:

```
.cmp              Forms commands
.exp              A text export of a file
.frm              Forms compiled file (binary)
.kmp              Keymap file (binary)
.lis              Generic listing (text output)
.pql              Text source of a VisualPQL program
.pwr              Output from a PWrite - text with PROCEDURE commands
.sch              Schema definition commands
.sr1 to .sr6      Database files 1 to 6. The files all have the
database name followed by a suffix
                  .srn where:
                  1 & 2 are the description of the database;
                  3 is the data in the database;
                  4 is the procedures;
                  5 is the journal;
                  6 is any secondary indexes
.tbf              Tabfile
.txt              Generic text
```

SIR/XS reads and writes files in several directories.

Home directory

This is the directory where you install executable files (for example, `C:\Sir` under Windows or `/bin/Sir` under Unix).

SIR/XS reads most of the internal data files from the home directory and loads help pages from the `help` subdirectory of its home directory. If you write programs that use the SirAPI or SIR/XS Z-library (Host), you must place your executable into the home directory, because the software looks for its internal data files in the same directory where the main module is located.

Normally SIR/XS does not create or modify files in the home directory except when the licence file is created.

Add the SIR/XS home directory to your list of PATH directories if you prefer to start SIR/XS from the command line.

Under Unix the `sirdir` environment variable must point to the home directory to run SIR/XS properly.

You can change the file where the SIR/XS settings are stored. This is normally .sir or sir.ini, depending on the operating system. The execution parameter `-ini:`*filename*. If this parameter is used then it must be specified first.

You can specify an alternate location for the licence file. To do this you need to define an operating system environment variable named SIRLCN that has a filename as its value. This filename contains the licence details.

Current working directory

This is the directory that is current when you start SIR/XS from the command line or specified as a starting directory associated with the icon or menu item you use to start SIR/XS.

You are strongly discouraged from using the home directory as the current working directory.

It may be convenient to use your database directory as the current directory.

The current working directory is the default place to load and save files that are referred by local names. It is also used for temporary data files that are deleted automatically on exit.

Temporary and Scratch Files

During execution SIR/XS creates and deletes scratch files in the directory named in an operating system environment variable **`SIRTEMP`**, or, if that is not defined then in an operating system environment variable **`TEMP`**. If neither of these variables are defined then scratch files are written to the current directory.

SirForms Key Mapping

If you redefine and save the SirForms keymap, it is saved into the current working directory. When the program starts, it looks for the keymap file in the current directory. If there is no local keymap, SIRForms uses the system keymap from the home directory.

SQL workspace

SirSQL stores the workspace file in the current working directory.

sir.ini or .sir initialisation file

The initialisation file contains a set of configurable parameters organised in sections. Each parameter is specified as a name followed by a value. The order of sections and the parameters in the file is not significant.

Each parameter has a default value that is used if the parameter is not defined. If a parameter name is not recognised, it is ignored.

Where a boolean value is expected, 1 stands for **true**, 0 - for **false**.

Some parameters can be altered during a session and the file is saved at the end of a session. If the file does not exists on startup, it is created at the end of the session. A single file is used by all SIR/XS products.

In the multi-user environment each user has his own initialisation file. Under MS-Windows the file `sir.ini` is created in the user's *My Documents* directory. Under Unix the file `.sir` is created in the user's home directory.

You can start sir using a different file by using the parameter `-ini:`*`filename`* on start up.

The file is in a format used for initialisation files on the specific operating system that is slightly different between Unix and Windows.

The following sections and parameters can be specified:

[sir]

General settings.

- `title` Text displayed in the main window title bar.
- `spsec` Number of seconds to display the startup bitmap. (Windows only)
- `mwndp` Specify 1 to save the position of the main window each time SIR/XS is closed.
- `mwndx` X coordinate for position of main window on startup. Specify mwndp=0 to ensure these values is not overwritten.
- `mwndy` Y coordinate for position of main window on startup.
- `mwndw` Width of main window on startup.
- `mwndh` Height of main window on startup.
- `fsize` Character size of fonts used in the main window and dialogs. (Windows only)
- `fnamm` Name of fonts used in the main window menus.
- `fnamn` Name of fonts used in the dialogs.

- `fnamt` Name of fixed width fonts used in the main window text area.
- `logfm` Specify 1 to record a log file.
- `logfn` The name of the log file.
- `lpcmd` The name of the print command (not Windows) eg "lp -c".
- `txted` The name of your prefered text editor executable.
- `htmlv` The name of your html browser executable for viewing help files.

[master]

Master Settings.

- `timeout` Specify timeout in seconds for Master send/receive. The connect timeout is twice this value.

[sql]

SQL Settings.

- `spsec` Number of seconds to display the startup bitmap. (Windows only)
- `mwndp` Specify 1 to save the position of the main window each time SIR/XS is closed.
- `mwndx` X coordinate for position of main window on startup. Specify mwndp=0 to ensure these values is not overwritten.
- `mwndy` Y coordinate for position of main window on startup.
- `mwndw` Width of main window on startup.
- `mwndh` Height of main window on startup.
- `fsize` Character size of fonts used in the main window and dialogs. (Windows only)
- `fnamm` Name of fonts used in the main window menus.
- `fnamt` Name of fixed width fonts used in the main window text area.
- `logfm` Specify 1 to record a log file.
- `logfn` The name of the log file.
- `nosys` Specify 1 to hide system tables in table lists.
- `autos` Specify 1 to automatically save the workspace on exit.

[odbc]

ODBC Settings.

- `ctout` Specify timeout in tenths of a second for ODBC connect.
- `stout` Specify timeout in tenths of a second for ODBC send.
- `rtout` Specify timeout in tenths of a second for ODBC receive.

Running in Batch

You can run SIR/XS in batch mode by supplying an input file that contains commands. The commands on the file are read and executed and, when the end of the file is reached, execution terminates and control is returned to the operating system.

The `IN=` parameter on the execution command specifies the input file and determines that this is a batch run. You can run sets of commands from files in normal SIR/XS in which case control is returned to the user when end of file is reached.

The IN= filename can be `stdin` in which case the input is taken from the standard input stream. This could be the keyboard, a pipe or redirected input using "<". If input is to come from the keyboard you can also use the `PROMPT` execution parameter which will put a prompt to the screen.

Output generated by batch runs is written to files. Output includes the listing of commands, remarks and any messages. The `OUT=` parameter on the execution statement specifies the default output file. If a default output file is not named, then a standard file called `SirOut.lis` is created. For example;

```
SIR IN=PQLPROG.CMD  OUT=PQLPROG.OUT
```

Commands and procedures that do not specify filenames write to the output file. Commands that specify a filename write output to the specified file. Output can be assigned to a file in normal SIR/XS and everything that is normally written to the scrolled output window is written to the assigned output file.

The output produced may include listings of remarks and/or commands and this is controlled by `PRINT BACK` settings. The default settings in batch list remarks, messages and commands.

The OUT= filename can be `stdout` in which case the output is sent to the standard output. This could be the screen, a pipe to another command or redirected output using ">".

The standard SIR/XS executable can run in this batch fashion and, if this is done, it runs with a gui window and remarks are written to the scrolled output window so that you have an idea of progress.

A non-gui version of SIR/XS is also supplied as a separate executable - `SIR`**`batch`**. This runs in a non-gui mode and can operate without a terminal window at all. All output is to

files.
(N.B. Some functions of the system are expressly designed to operate interactively in a graphical environment and may cause problems if operated in any other way. If a batch run fails to operate in the expected fashion, run the same input file in standard SIR/XS to ascertain if this reveals the source of the problem.)

If a batch run executes VisualPQL programs that contain commands that require interactive input or output, the program is not executed. Similarly, commands that require use of an editor are flagged as an error in a batch run.

Login Procedures

A login procedure can be executed when the system starts or when you attach to a new database. This is typically used to define an initial working environment. The commands in the login procedure might run one or more programs and define system options.

Login procedures are not run in batch executions.

Login procedures are members in the procedure file in the `SYSTEM` family. When the system is started in a normal fashion, if there is a `START` member in the system family, it is automatically executed before the main menu system starts.

To execute an alternative login procedure, specify the `EX=` execution parameter. The specified procedure must be in the `SYSTEM` family. If an `EX=` parameter is specified, the default login procedure is not executed.

If the initial procedure creates a `WINDOW`, it is then controlling the system. If there is no initial procedure or the initial procedure does not create a window, the system is started with the menu from the system procedure file `SYSTEM.START`. Since initial procedures are run before the windowing gui is active, they may not use dialogs. If you wish to have an initial dialog, invoke it from the `INITIAL` section of the `WINDOW` creation.

Execution Parameters

The SIR/XS command line has various optional keywords and parameters. If an unrecognised parameter is found, it is taken to be a filename to be processed. This is held as the value of the global variable `INFILE` that can be accessed by your applications. Action is taken depending on the file extension as follows:
`.srn` is taken to be a database file and the file is opened
`.exp` is taken to be an Export file and the database is imported
`.pql` is taken to be a VisualPQL procedure and is run. Note that this is run **after** the menu system is started and cannot be used to define alternate menus.

`.srm` is taken to be a VisualPQL procedure and is run. Note that this is run **before** the menu system is started and can be used to define alternate menus. If the program does not define an alternate menu system then it cannot include gui commands or functions.

If a parameter starting with a `*` is found, it is taken to be a global variable and the following value is assigned to that global variable e.g `*user=fred` results in a global variable `USER` with the value "fred". If the value contains blanks or other special characters, enclose it in quotes.

Following is an alphabetical list of all the execution parameters.

`-ini:filename`
: Sets user preferences file name (usually sir.ini or .sir). Note, if this parameter is used then it must be the first one on the command line.

`CENY=yyyy`
: Sets the year used for calculation of the century when two digit years are input. When a two digit year is input, the system compares the input year with this parameter. If the parameter is less than the input, the century is set to the specified century. If the parameter is greater than or equal to the input, the century is set to the specified century plus 1.

  The default is 1930.

  To illustrate the calculation, assume the default setting of 1930. Then any input year in the range 00 - 29 is assigned a century of 20nn; any input year in the range 30 - 99 is assigned a century of 19nn.

  All variables defined as dates are held internally as a number of days since the start of the Julian calendar on Oct 15 1582.

`DB=database_name`
: Sets the initial database. If this parameter is omitted in an interactive run, the system starts without a database. If it is omitted in a batch run, the `CONNECT DATABASE`, or `CREATE DATABASE` command as appropriate can be used in the input command stream.

`EL or ERRORS=n`
: Sets the task error limit. This determines how many error messages are displayed in a particular task. The default is 50.

`EX=member_name`
: Specifies that the named member of the `SYSTEM` family is executed as soon as the system starts. Does not apply to batch runs.

`F=family_name`
: Sets the default family; otherwise the `SYSTEM` family is the default.

`FAMP=family_password`
: Specifies the default family password.

`GRP=group_name`
: Specifies the group name used when connecting the tabfile specified by `TFL=`.

`GPW=password`
: Specifies the group password used when connecting the tabfile specified by `TFL=`.

`IN=filename`
: Specifies the name of the input command file for a batch run. Specifies that this is a batch run.

`JOURNAL={ ON | OFF }`
: Turns journaling on or off for the database specified on the `DB =` parameter. If not specified, the journaling status set for the database is used. This option requires DBA security level.

`M=member_name`
: Sets the default member name.

`MEMP=member_password`
: Specifies the default member password.

`MST=name`
: Specifies that this is a concurrent session and names the `MASTER` process to use. See Master for details on concurrency.

`NAC`
: (**N**o **A**uto **C**onnect) stops the automatic connection of the tabfile specified by `TFL=`.

`OUT=filename`
: Specifies the output file for batch run. Specifies that this is a batch run and must also have an `IN =` parameter.

`P='directory'`
: Specifies a **P**refix to use to locate the database. Specify the prefix in quotes terminated by a backslash. This accesses databases in other directories. This only applies to the database files. Other files, unless otherwise specified, are read from or written to the default directory.

```
PB =
[ON|OFF]
   [YES|NO]
   [[NO]CALL]
```

```
[[NO]COMMANDS]
[[NO]REMARK]
[[NO]REPEAT]
[[NO]SKIPPED]
[[NO]TASK]
[[NO]USER]
```

Specifies print back options. These are `OFF` by default for all options except `REMARK` and `USER`. In batch runs, these are `ON` by default for all options except `TASK`.

Specify `PB` multiple times to select or deselect multiple options. For example:

`PB=CALL/ PB=REPEAT`
`ON or YES` sets the default print back options `ON`. `OFF or NO` turns print back off completely.

`CALL` echoes all the lines included by `CALL`, `INCLUDE` and `RUN` commands.

`COMMANDS` echoes all commands.

`REMARK` writes remarks as tasks are executed.

`REPEAT` echoes all the commands generated by a `DO REPEAT`.

`SKIPPED` echoes commands that are skipped in `CIF` blocks.

`TASK` writes statistics of CPU usage and other activity. The default is `NOTASK`.

`USER` writes remarks detailing the SIR/XS attribute created when a new filename is encountered.

`PROC=filename`

Specifies a procedure file as the initial procedure file. The default is the procedure file of this database or `SYSPROC` if a database is not attached. Specify either a filename, such as `'C:\myproj\myproj1.sr4'` or use `SYSPROC` to attach the SIR/XS system procedure file.

Note: When SIR/XS starts it runs the member `SYSTEM.START` from the procfile specified by this keyword. `SYSTEM.START` calls a main menu program such as `SYSPROC.SYSTEM.MAINMENU` to start an interactive session.

`PS=n`

Sets the number of lines per page when output is assigned to be written to a file. Paging can be suppressed by specifying `PS = 0`. The default is 60.

`PW=database_password`
Specifies the database password in conjunction with the `DB=` parameter.

`PWD=n`
Specifies the output width in characters. The default is 120.

`QUIET`
Suppresses initialisation and termination remarks.

`RS=read_security_password`
Used in conjunction with the `DB =` parameter, this specifies the database read security password. A read security level of 0 (zero) is assigned if no password or an incorrect password is supplied.

`SIRUSER=name`
Sets an alternate name which is logged to the journal as database updates are done. If this is not specified then the user name is got from system environment variable `USERNAME` or `USER`. Could be used as the basis for audit trails by individual users who update the database. Specify a SIR/XS name.

`SL=n`
Sets the default size of string variables implicitly declared during VisualPQL compilation or Schema definition. The default is 32.

`SRTN=n`
Sets the default number of records assumed by a sort. The number of records in a sort is assumed to be the number of records in the attached database. Specify an appropriate number if you are running without a database, using large tabfiles or generally need to set a different value.

`TFL=tabfile_name`
: Specifies a tabfile to connect during initialisation. This becomes the default tabfile.

`TFFN=filename`
: Specifies the operating system filename of the tabfile specified with `TFL=`. Specify this where the operating system filename is not the tabfile name with a `.tbf` suffix.

`UPW=password`
: Specifies the user password used when connecting to the tabfile specified by `TFL`.

`USER=name`
: Specifies the user name used when connecting to the tabfile specified by `TFL`.

`WD=line_width`
: Sets the line width of the output file. The default is 121.

`WL=n or WARNINGS=n`
: Sets a limit on the number of warning messages displayed by a task. The default is 50.

`WS=write_security_password`
: Specifies the database write security level password. If no write security password is supplied or if an incorrect password is supplied, a write security level of 0 (zero) is assigned.

Running from a web server

Web Server Support

This interface capability allows any web server that supports CGI to communicate with SIR/XS and invoke VisualPQL programs. These can then request data from the web server and send output back to the server that then appears on the user's web page. A format for some procedures, notably `TABULATE`, allows these to produce output in html format and send this directly back to the web. Output directed to the web server is written to a local file if the programs are run locally so that programs intended for use over the web can easily be developed and tested.

The following information is intended to provide some help for the users who have no prior experience with Web server software but want to use SIR/XS on their Web server.

Web servers' configuration includes a system of virtual directories available to the Web users. Sometimes this virtual file system may exactly match a subtree of your real file system, but it doesn't have to. Here is the example of a possible configuration under Windows NT:

```
Dir          Virtual     Real path
-------------------------------------
Root         /           d:\www\home
Images       /images     d:\www\images
Scripts      /cgi-bin    d:\www\programs
```

Note, that in the virtual file space, the /images directory is a subdirectory of the root directory, but on this particular server the two physical directories have a common parent.

Assume that you have installed SIR/XS into `c:\SIR/XS` on your Web server. Include this directory into the `PATH` of a special user account used by the Web server to run the programs on the Web browser's requests or include it into the system-wide `PATH`, so it is available for all users.

After installation, the SIR/XS Web executable `sirweb.cgi` is installed into the SIR/XS's home directory. `sirweb.cgi` can be used on all platforms. However, if you use the MS Windows version of SIR/XS, you can also use `sirweb.isa`. Use the isa version instead of cgi if your Web server supports the ISAPI protocol, because it is faster than the traditional CGI access. Use the appropriate name in the `default.htm` (see below).

Make sure that the sirweb's access permissions allow it to be executed by the Web server process (or by anyone).

You may need to rename the file if your Web server insists on a particular file extension for a particular type of Web application. You need to specify to your web server that sirweb.cgi is a binary executable file (as opposed to Perl and shell scripts). For example, under MS Windows, it has a format of the normal Win32 EXE file. The second, Windows specific file, sirweb.isa has the Win32 DLL format.

Set up an appropriate html file in the appropriate directory. For example, `default.htm` in the Web server's root directory. The HTML references the installed sirweb executable as a simple link (URL) or as an action handler of the user input form. e.g.

```
<a href="http://b1/scripts/sirweb.isa">Click here to try the SIR
Internet Database Server
```

or

```
<FORM ACTION="/scripts/sirweb.isa" METHOD=POST>
```

The sirweb executable supports GET and POST request methods and takes arbitrary number of CGI parameters. The `sirapp` parameter specifies the name of the VisualPQL program to run. This VisualPQL program can use the CGI functions to get any number of named, additional parameters. If you do not specify a value for `sirapp` the system uses `SYSPROC.CGI.DEFAULT` as the default member to run.

Other Issues

SYSPROC

If you are going to develop a system for CGI access, decide where your standard procedures are to be stored. When the interface starts up, you need to be able to run your initial procedures and there is no database attached. The most convenient place for this is `SYSPROC.CGI.DEFAULT` but be sure to have this saved as a file because the system procedure file is recreated when you re-install versions of SIR/XS.

Procedure References

Do not rely on the current state of the system for default procedure files, families or members. Other procedures may run and reset these so use fully qualified names.

Database and Tabfile Access

Do not rely on databases or tabfiles being connected or being the default. Specifically connect, disconnect databases or tabfiles as required and fully qualify all references.

Caching

To optimise performance, various components of the system maintain caches of previously accessed components. For the web server, these are pages, for SIR/XS, these are procedures. If you are developing and testing, then be aware that you may need to force the system to use a new copy of some procedure or page that you have just changed.

Directory and File References

In your VisualPQL programs, you create html that you send to the web server e.g.

```
program
write (CGI) '<CENTER>'
write (CGI) '<img src="/images/sirsplash.gif">'
write (CGI) '<h1>Welcome to the SIR Internet Database Server</h1>'
write (CGI) '</CENTER>'

write (CGI) '<FORM ACTION="/scripts/sirweb.isa" METHOD=POST>'
write (CGI) '<INPUT TYPE=HIDDEN NAME="sirapp" VALUE="sysproc.cgi.go">'

write (CGI) 'Application to run: <SELECT NAME="sirtxt">'
write (CGI) '<OPTION VALUE="hello" SELECTED>Simple Text'
write (CGI) '<OPTION VALUE="entry">Data Entry'
write (CGI) '<OPTION VALUE="excel">Excel'
write (CGI) '<OPTION VALUE="spss">SPSS'
write (CGI) '<OPTION VALUE="tables">Tabulations'
write (CGI) '<OPTION VALUE="splash">Welcome'
write (CGI) '<OPTION VALUE="freport">Report'
write (CGI) '</SELECT>'


write (CGI) '<INPUT TYPE=SUBMIT NAME= "sirmem" VALUE="Run"><p>'
write (CGI) '<INPUT TYPE=SUBMIT NAME= "sirmem" VALUE="fcache">'
write (CGI) '<INPUT TYPE=SUBMIT NAME= "sirmem" VALUE="entry">'
write (CGI) '<INPUT TYPE=SUBMIT NAME= "sirmem" VALUE="excel">'
write (CGI) '<INPUT TYPE=SUBMIT NAME= "sirmem" VALUE="Register">'
write (CGI) '</FORM>'
write (CGI) '<CENTER>'
write (CGI) '<img src="/images/SIRXS3.jpg" alt="SIR/XS">'
write (CGI) '</CENTER>'
end program
```

In the html, you may have references to other files such as images or output files that are accessed by other programs. Be aware that these references must be resolvable by the web server.

Any images that are referenced are typically in an `images` sub-directory from the web server root so ensure that any predefined images that you reference are in that directory. (If you use the `DESCRIPTIVE` procedure, this uses an image `red.gif` that can be found in the sir - images sub-directory and can be copied to the web server images directory.)

If you are writing files for your users to browse with other packages you need to send them to an appropriate directory and reference this in your html. e.g.

```
retrieval
process cases
. record is employee
c   Find last name
.   compute revname = trimlr(reverse (name))
.   compute len     = abs(srst(revname,' '))
.   compute lname   = reverse(sbst(revname,1,len-1))
. end record is
. process rec occup
.   get vars position startdat startsal division
.   perform procs
. end process rec
end cases
excel save file filename = 'd:\wwwroot\testex.dbf'
    / variables = lname division position startdat startsal
end retrieval
program
write (cgi) 'Your data is <a href="/testex.dbf"> here </a>'
end program
```

Content Type

By default the cgi output content type will be text/html. This can be changed by passing the variable "Content" containing the alternate type. For example:
`http://www.sir.com.au/cgi-bin/sirweb.cgi?sirapp=cgi.hello&Content=text/plain`