The Newsletter for SIR UK Users

SIR REPORTER

Contents

Editorial

Welcome to the first 2005 issue of Reporter – and a belated Happy New Year to all our readers. This year should be a good one: SIR/XS will be released and there is an international conference and training in New York in May. There are details of both in this issue, and early booking is advised.

Looking back over 2004, I was sorry to miss the September UK event at the RSS. My personal thanks go to Adrian Hodgson for his excellent conference write-up in this issue – it's not a substitute for being there, but the next best thing when attendance is simply impossible.

The committee was very pleased to welcome a new member at the AGM: Jon Johnson from the Institute of Education has joined our ranks. Actually 'AGM' is now a misnomer – the motion was carried that we no longer have to meet annually! For details of that and other business, see the minutes of the AGM in this issue of Reporter.

This issue closes with the usual SIR tips – the Silly section will return when space permits!

Kathy Brooks
Kathy@forvus.co.uk

SIR is at http://www.sir.com.au
UK SIR Users Group is at http://www.soton.ac.uk/~sug

Chairman's Chat

Happy New Year to you all.

It appears that I have been elected as chair again so you will be getting my thoughts in Reporter for another year.

This is hopefully the year we see SIR/XS with all the bells and whistles you have always been asking for. You may find there are some things you personally did not ask for but which nevertheless will be very useful I know I am waiting for some features to help me in my work.

The International Conference is in New York this year and incorporates SIR/XS training so maybe you can get a trip abroad as well as getting brought up to date in the product. See the user group website at *http://www.soton.ac.uk/~sug* for more details.

Don't forget to visit the user group website or the SIR website at *http://www.sir.com.au* to get details of new releases.

In the UK we are hoping to have an event once SIR/XS has been released to show you what you are missing and get users impressions of how they will use the extra features.

Dave Doulton

Training and International Conference 9-11 May 2005, New York

The next international SIR conference will be held in New York at Columbia University and will include training in SIR/XS, the forthcoming new release.

The cost per delegate is US$770 or €660 or £440, 2nd & subsequent delegates from the same organization get 25% off.

Programme

May 9th	
10.00	Registration and Coffee
10.30	Welcome and introduction by Mo Reardon SIR and Howard Andrews Columbia Data Centre
11.00-17.00	Introductory training by John Lemon Aberdeen University or advanced training by Tony Reardon and David Baxter SIR, Tom Shriver DataVisor and Dave Doulton Southampton University including break and lunch
18.30	Meet for dinner at Carolines Comedy Theatre on Broadway
May 10th	
09.00 - 16.30	Introductory or advanced training continued including breaks and lunch
18.00	Meet for dinner on board World Yacht for 3 hour cruise at 7pm
May 11th	
09.00 –10.00	SIR/XS What's new in the forthcoming SIR release Tony Reardon SIR
10.00 – 17.00	Training and conference sessions including breaks and lunch

Accommodation

Optional accommodation has been reserved at the Holiday Inn Midtown 57th Street which is conveniently located for famous New York attractions, and easy to get up to the Columbia campus--a two-block walk to the reliable 'A' subway at 59th street, and a 15-minute ride to the 168th Street stop, which is right at Columbia. Just three blocks from Central Park and Carnegie Hall with Lincoln Center nearby. Situated within the Theater District, a Broadway show is a moment away. JFK Airport and LaGuardia Airport are only 45 minutes away from this Manhattan hotel location. The hotel also features a bar, several in-hotel dining options, and room service, as well as a fitness center and a swimming pool on the roof. We have secured a room rate of US$159 per room, per night, double or single occupancy, one king bed or 2 doubles. These rates apply for any night Sunday May 8th through to Wednesday May 11th inclusive. The phone numbers for reservations are 1-800-231-0405 or 1-212-581-8100. Quote Code XCV. Please make your reservations prior to April 15th 2005, or the hotel will not be able to guarantee availability and/or our specialized room rate.

Mo Reardon

SIR Development

SIR Server

SIR/XS includes a brand new module – a SIR server. This is a new executable that allows another standard SIR session to connect as a client and to transmit commands to the server, execute those commands remotely and retrieve output.

The basic concept is that the client sends commands to the server, then tells the server to execute those commands and then gets the results back from the server. The commands can include any SIR commands and can run procedures at the server. The client can either wait for the execution to finish or can just continue, possibly checking later on the status of the procedure.

All communication is machine independent and there is no need for any local access by a client to databases or procedures on the server. This means that the server can be implemented on one type of hardware/operating system e.g. SUN Solaris, with clients on a different system e.g. Windows

There are a number of new functions that can be used when running as a client. These allow the client to logon/logoff; send a command string or a buffer to the server; execute previously sent commands; check if execution has finished; find out the number of lines of output to be retrieved and retrieve lines of output. Lines that are not wanted can be skipped.

There are also functions that allow a client to act as a server administrator. General client access to the server can be password protected and administration functions can be separately protected.

When running procedures on the server, any standard output is automatically made available for the client to retrieve. In order to make this easier to control, there are some new commands and functions intended for use when a process is running on the server. (If they are used in a standard SIR session, they have no effect.) The server process can set a 'NOOUTPUT' flag so that standard output is just discarded thus making sure only wanted lines are sent back to the client. There is a new function that writes standard output regardless of the setting of NOOUTPUT so that an application can send exactly the output it wants back to its initiating client. The server process can also set a return code that is passed back to the client on completion of the execution.

From the client point of view processing is as follows:

1) Client logs on to server and gets back a client id.

2) Client sends text SIR commands.
Control usually passes back without any actual transmission taking place - transmission only happens when maximum message size is reached.

3) Client starts execution.
This sends any outstanding buffered commands. It re-initializes any settings or output from a previous execution from the same client, runs the commands and returns a completion code. Any output is waiting on the server. Commands can include all SIR commands, 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.

3.1) If the client chooses not to wait for the execution to finish, then the return code from the execute does not mean anything. Similarly if the execution returns a timeout, it may still be running. The client can subsequently test to find if the execution has completed successfully.

4) Client gets count of number of lines of output. The client gets each text line, which is normally the next line but can 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 the number of lines available at any point.

5) Client can repeat steps 2 - 4.

6) Client logs off.

Sample Program

```
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
```

We have made early copies of SIR/XS available on a limited basis. If you are interested in trying new features such as the SIR Server, please email david@sir.com.au for a download address.

Tony Reardon
tony@sir.com.au

UK Conference 24th September 2004, London

Paper 1 – Making ODBC Easier

Dave Doulton (University of Southampton)

Dave guided us through the initial stages of setting up SIR for ODBC connectivity to other software. The main topics covered were: -

- ODBC import
- ODBC pql
- ODBC members

The initial set-up involves the use of the Administration Tools menu in windows Control Panel. From the Administration Tools menu you need to select Data Sources (ODBC).

The first two tabs under Data Sources are User DSN and System DSN. Choose the User DSN tab to set up ODBC access for a single user; Choose the System DSN tab to set up ODBC access for multiple users. Both are configured in the same way, but using different tabs.

When you click on Add the system loads up drivers (including SIR ODBC drivers) You will need to choose an ODBC source type, click Finish and then a configuration screen appears. The configuration screen will vary according to the source selected (see Daves slides for configuration examples for SIR,csv, Excel, Access, FoxPro, Dbase, Oracle).

ODBC importing into a SIR database is easiest for a caseless SIR database as there is no caseid to worry about. The approach involves selecting Import records from the database menu, and then selecting the ODBC button. You can use row order as a sort id if there is not an obvious choice in the data.

ODBC PQL is the sensible alternative when you want to read and process data from a file in another source, without storing it in a SIR database.

This is based on a set of SIR PQL commands (e.g. to Connect to the ODBC source and Disconnect).

- CONNECT conid SERVER name {DATABASE name | TABFILE name} [USER name] [PASSWORD name] [PREFIX name] [UPDATE | READ] [ERROR errid]
- DISCONNECT conid [ERROR name]

One of the key points is the need to set a User Name and Password.
These are user selected and anything can be used, as long as you use the same names when references to these are required.
e.g. a User Name could be ‘me’ and a password could be ‘mypwd’.

For more information see SIR Help on ODBC.

Dave’s templates

Dave has created a template PQL program (available from the SIR Users group web site) to simplify this.
This involves calls to a series of system procedures. (e.g. for Connecting to the ODBC source call sysproc.odbc.connect). With some small changes this can also be used for linking to web applications.

One of the key benefits of ODBC is that it can be used to speed up development work when using a multi-package environment.

For example where corporate reasons dictate the use of tools such as Oracle, or Sql server, data can be exchanged with SIR using ODBC in instances where the preferred programming language is PQL.

As his coupe de grace Dave demonstrated SIR on his Windows XP machine using ODBC to access data held in Oracle on a Linux box on the other side of the room over a wireless network. It all worked a treat!

Paper 2. From Complexity to Simplicity to Complexity

Peter Lauf (Germany)

Peter talked about the origins of SIR to deal with data complexity by looking back to some American proceedings from 1977.

This product is addressed to the researcher and data analyst faced with the need to manage and analyze data sets in the form of complex files. These complex file arises when data is collected from multiple sources over a period of time....“

Essentially, statistic packages were being used to analyse data and these required rectangular data sets. SIR was set up to store and handle complex hierarchical data and to summarise this data easily into rectangular data sets. In the early days it became a logical complement to the analysis packages for storing the data.

Simplicity

Over the last two decades we have seen an explosive growth and spread of IT and a proliferation of desktop PC’s.
This inflationary phase pushed the relational model.
Relational databases became the commodity of choice (e.g. Access, MySQL, ...)
In order to keep pace SIR became a `relational Database`.

Back to Complexity

Peter concluded that data complexity had reappeared, but in a less obvious ‘hidden’ way.

The complexity can be hidden in Excel tables or SAS files or alternatively data can be held inefficiently in Oracle databases.

Peter concluded that SIR had survived because of a Unique Selling Point.

This was not because of its superior relational technology but because of
its superior functionality to handle complex data.

In the 1980s SIR was called a “complex system” compared to programs like SPSS

Today SIR is small and highly efficient compared to many over featured programs.

Paper 3 – The old sweat rides again. (or PQL my favourite bits)

Niels Veldhuijzen, CITO, Netherlands

Niels talked about some of the SIR functions that he has used to help in manipulating data files. He has years of experience of using PQL for manipulating data files. This is necessary because he has had to beg the Database Administrator (on his knees at times) for data in the required format.

Niels’ favourite commands /functions are, in alphabetical order: -

- **BEGIN … END BEGIN**
- **CATegorical VARiableS**
- **Full REPORT generator**
- **REDEFINE ARRAY**
- **REVERSE**
- **SBST**
- **SRST**
- **SRCH** *(my alphabetical order)*

(The last one is a theoretical favourite only. Niels likes the idea of this one and thinks he should be using it somewhere!)

Niels needs to determine the last part of a filename from a string ignoring the rest of the directory structure. This was my own favourite from those that Niels showed as I had been trying to do something similar the week before!

```
e.g. set s
   ('C:\MySirDataBases\LongPath\Myjob')

compute s=  reverse(sbst(reverse(trim(s)),1,

   abs(srst(reverse(trim(s)),'\')) -1))
```

The result is the string ‘Myjob’

Further details for CAT VARS, REDEFINE ARRAYS and BEGIN … END BEGIN are in Niels’ slides (see SUG web site)

Niels saved his absolute favourite to last : FULL REPORT Generator.

He announced that: -

A long time ago, … in a galaxy far away ! .. (reviewers translation) , I declared my love for the full REPORT generator.

It still is the *pièce de résistance* in my data handling work: it can do anything!

He uses it for everything, except for making a Report! (perhaps it makes him a cup of coffee as well ? – reviewers comment)

He uses this for recombining his data in the order he needs it.

Sounds like another demonstration of the flexible uses of SIR PQL.

Paper 4. SIR and the Management of Cancer Trial Data

Kate Motohashi, Gray Cancer Institute

Kate has a biological background and has been involved in a multi-centre randomised trial of treatment methods for Bladder Cancer since May 2001. To give the trial its full name: -

'A multicentre randomised Phase III trial of Radiotherapy with Carbogen and Nicotinamide in the treatment of locally advanced bladder cancer'

Several treatment methods are available surgery, chemotherapy and radiotherapy. Radiotherapy has traditionally not been very successful and one of the potential reasons is low oxygen concentrations. A randomised trial is under way to compare radiotherapy alone with radiotherapy together with the use of breathing apparatus to increase oxygen concentration levels. This adds two modifying agents carbogen (a mixture of 98% Oxygen and 2% carbon dioxide), and nicotinamide.

In order to ensure statistical validity of the trial a minimum of 330 patients over a 4 year period was required. Recruitment commenced in November 2000. Currently 253 patients have been recruited from 14 hospitals with the rate of recruitment being slightly lower than required, so the recruitment period will be extended for another year.

The trial statistician chose SIR as the database because: -

- It is excellent for handling clinical data, particularly where patients have multiple visits
- It is highly compatible with SPSS which was the chosen analysis tool

Kate created a SIR2000 case orientated schema with 13 record types including personal data, morbidity data, cystoscopy results and quality of life measures.

Several complications with data handling have arisen during the study so far as follows: -

- Two different coding scales have been used for recording cystoscopy details and some of these have required manual recoding to ensure consistency
- Multiple missing values have been used as some entries will remain missing always (BLANK) and some may be filled in at a later stage of data collection (Kate has used '9' /'99' /'999' for these.) Care will be needed in distinguishing between these for statistical analysis. The PQL function MISNUM can be used to help with this.
- There is a potential need for the equivalent of wild card searches in the pql programs. The use of the string function pattern may be useful here, using the match anything character "@" (see excerpt from SIR Help below).

```
num = PATTERN (strX , pattern_str)
```

Returns 1 if the pattern specified by pattern_str is in strX, otherwise 0. Both arguments are strings and can be variables, constants or expressions. The pattern can contain the match anything character "@". Undefined is returned if either argument is missing or undefined. For example, the following returns 1.

```
THERE = PATTERN ('Mr. Ralph Jones',
   'Mr.@Jones@')
```

- There is a need for recording the reasons for data editing when data is changed during the cleaning process.

Paper 5. Tips and tricks for producing easily maintainable code in SIR

Fran Williams, University of Essex

Fran has one of the unenviable jobs in the computing industry; that of picking up large programs written by non programmers some time ago that are not adequately documented. Fran has the job of updating and running these programs yearly to incorporate the latest data sets.

This is for the British Household Panel Survey where a cohort of 1500 individuals is re-interviewed yearly. Fran is transforming data for the 13th year at present (2003)

The code is too complex and large to re-write in one go so Fran described her step by step approach. Fran's key messages were: -

- If it aint broke don't fix it, particularly if you don't have much time
- Rewrite so minimal changes required in future
- Make sure changes work
- More can be done next year

Fran described her re-write this year of the derived variables section.

The main commands and functions used in the re-write were: -

DO REPEAT - for condensing repeated sections of IFTHEN ELSE type coding

GLOBAL parameters for values and conditions that might change to replace hard coded values.

Example of existing code

```
ifthen (FICODE eq 1 or
          (FICODE GE 5 and FICODE LE 42))
C     Current condition for state benefit
   payments
      ……..
      ……..
end if
```

Replacement code using a GLOBAL

```
GLOBAL FIBCOND =
   $FICODE eq 1 or (FICODE GE 5 and FICODE LE
   42)$

ifthen (<FIBCOND>)
……..
……...
end if
```

Compiler Directive CIF

CIF is in effect short for Compile IF. It determines whether code is to be compiled or not and is usually used with GLOBALs as in the following example.

Example: Using CIF and Globals to run as a Retrieval update or in test mode

```
GLOBAL UPDATE = UPDATE
GLOBAL DEBUG = 1
GLOBAL WP = M         | Wave Prefix of current
   wave (13) of data

Retrieval <UPDATE>……
  CIF DEF <UPDATE>
    Put vars <WP>VARA = VARA
  CIF FALSE
    CIF EQ <DEBUG>, 1
.     Write "<WP>VARA would be updated to ",
   VARA
    CIF END
  CIF END
```

In this example the user can set the UPDATE GLOBAL to UPDATE in order to run the program as a Retrieval Update and update the <WP>VARA variables in the database. The alternative is to leave the GLOBAL variable UPDATE undefined and to set the DEBUG Global to 1. This runs the program in Retrieval mode and writes a 'debug' statement which documents what values would have been updated.

The database variable names consist of a root-name which is consistent over time and a wave prefix (which is defined as a GLOBAL <WP>) which changes each year. Thus wave 1 variables are prefixed by A, wave 2 by B etc and the current wave (wave 13) by M.

Fran also described the hidden time-bombs (other than year 2000) that can be built in to computer code without proper planning.

One procedure calculates the value of a variable which indicates where the latest job information for a respondent can be found. It has a root-name of JLID and a database name of <WP>JLID. However, in this procedure a temporary variable, LJLID, was calculated. The code worked fine until last year, wave 12, when the database variable name was also LJLID! This resulted in incorrect values being assigned to the database.

Even more time bombs would have appeared at wave 14 where N<rootvar> was used as a counter in many places. However, all the code has now been rewritten in such a way as to ensure that the names for temporary variables are different in format to database variables so that at no wave can they have the same name. All temporary variable names now contain an underscore which is a character that is not used for any database variable.

Fran concluded that for applications that need running for different time periods: -

- Make use of SIR compiler directives to produce data driven systems
- Rewrite sections so that they will never need rewriting again (i.e to remove hard coded values embedded within the code.
- No need to do everything at once if re-writing a system that works
- Name temporary variables carefully to avoid any potential clashes with normal database variables now or in the future eg X_TEMPV if underscores never used for database variable names.

Paper 6 Recovery from the Backslash? Experiences of a German SIR User 1983-2004.

Klaus R. Allerbeck JW Goethe University Frankfurt, Germany

Klaus presented an interesting historical perspective on the fading use of SIR in Germany over the years. He described four key factors in this decline.

The disappearance of responsive, german-speaking, problem-oriented social science support .

In the early 1980's SIR was a common package in use in German universities. Although traditional computer centres mainly adopted a snobbish approach to packages like SPSS and SIR, the academic computing teams were more open minded, and even interested in how SIR handled system dependencies.
Its use was helped and supported by a company called STATUS who were able to provide German speaking support for the package and co-ordinated the exchange of information and ideas between the different users.
STATUS explained the neat, compact demonstrations that English speaking SIR programmers were so fond of, in German, and made them accessible to social scientists. Existing users could survive without the support of STATUS once they had figured out the interface of the SPERRY operating system, its file structure and the Conversational Time Sharing system.
However, Once STATUS was gone, there was no support for new German users

The decline of mainframe computing.

In the mainframe era when computer resources were centralised there was a central budget available for software licenses. As computing became more de-centralised hardware budgets were made available at a more local level, but the issue of funding local software was conveniently ignored in many cases. As student status is ill-defined in Germany, site licences for software were out of the question. Issues such as illegal copying of software ("Raubkopien") and the danger of spreading viruses both placed a restriction on sharing resources.

Another ,'hidden' benefit of the mainframe era was the informal sharing of ideas that occured whilst users gathered to collect the output from overnight batch runs. Klaus described this as a `great opportunity to chat up girls as well' !

The German adaptation of Personal Computing

Once the PC started to spread like wildfire the union became interested in the health requirements, for the workers and the need to ensure that equipment was standardised according to Deutches Industrial Norms (DIN). For keyboards this meant conformity to DIN 2137: the uniform German IBM clone keyboard. The origin of this norm revolved around keyboard layouts being the same as those on typewriters.
The huge disadvantage with this was the lack of the backslash (\ ) used for separating the directory folder part of filenames.

Klaus quoted an example from Stuttgart. A SIR enthusiast had used PQL to solve a known computer problem. However when new staff arrived with a Windows background they soon phased SIR out using the argument that it was too complicated for them and not user friendly.

However on a brighter note there is a forthcoming merger of the Welfare and Labour Offices next year which may offer some scope for SIR enthusiasts as there will be a need to check records in historical systems.

Paper 7 Demonstration of SIR/XS.

Randy Banks, University of Essex

Randy started by reminding us of the new features introduced in SIR2002.

For data management Sir2002 improvements have included multiple databases, integrated master modules, sql server for ODBC, secondary indices and long strings.

For application development the advances were GUI based, screen painters, unlimited pql program size, a working spreadsheet mode, PQLForms and internet enabled.

The SIR2002 software development environment now uses a single generic source on Windows XP, with releases for Windows, Sun, HP, IBM and Linux. Required fixes to software are now fixed on demand and revisions provided on the internet for download. The latest version of SIR 2002 at the time of the conference was 20.35.

Then we were treated to tantalising glimpses of the exciting new developments in progress or planned for SIR/XS. These included 32 character

names, improved journaling and recovery, the facility to store the sr3 data file in multiple physical files, GUI improvements, XML support, SQL in PQL, and a SIR server environment.

Thirty two character names are being introduced everywhere that names are required (i.e. database and record names, variable names , index names, passwords, tabfiles, families, members, sub-routines and sub-procedures, labels, and buffer names).

Standard 32 character names will need to start with an alpha character and can consist of letters, numbers and four special characters ($ # @ _).
Non standard names will also be possible consisting of 1 to 30 characters enclosed in curly braces { }. These will be able to contain any characters and will be stored without the braces for sort order purposes.

The journaling part of SIR has been re-written in SIR/XS in order to facilitate roll back, roll forward, auto recovery, transaction processing and incremental unload.
The journal will now contain a file header, an update header and an update record.
Journal processing has been introduced with PQL access to record values.

```
e.g.
PROCESS JOURNAL  [ FROM = updlevel |  START =
   date  [,time] ]
 JOURNAL RECORD IS
.. pql access to record variables
 END JOURNAL RECORD IS
END PROCESS JOURNAL
```

The facility to hold the sr3 data file in multiple physical files has been introduced to help with storage and backup requirements for large databases. Data can now be split across multiple devices and in a situation where some parts of the database (e.g. older records) don't change, it will be possible to split the database so that regular backups only need to be performed for the data that is changed. The split has been developed based on caseid ranges for case structured databases and record type or key values for caseless databases.

GUI Improvements include the facility to replace the main SIR menu on the fly,display a tipbox, new clipboard copy and paste functions, right mouse click, mouse over messages, and tool bar buttons on dialogs.

XML support involves the need to introduce functionality to be able to handle XML data file structures (hierarchical text files with Tags enclosing data) and XML names.

An example of XML structure involving start Tags (e.g <name>) and finish Tags (e.g. </name>) is shown below.

```
<people>
   <person>
      <name>John D Jones</name>
      <salary>2150</salary>
      <born-year>1956<born-year>
   </person>
   <person>
    …..  next persons details
    …. etc
```

SIR /XS will include a new PQL procedure to produce XML files and will also include features to assist in creating generic tagged files, such as identifying the output file as a tagged file on the OPEN command, a WRITE command which will surround each output element with start and finish tags, and optional control of tags to default to variable names.
It will be relatively easy to create an interface that can cope with reading simple XML produced by the SIR XML output routines. It will be less straightforward to create reading routines that are able to deal with the standards for reading XML under all circumstances.

Another new development is the facility to undertake some basic SQL from within PQL using a PROCESS SELECT program block.

```
PROCESS SELECT variable_list | * | ALL
    FROM     [tabfile.   ]  table_name
                [database. ] record_name

    [INDEXED BY index_name]

    [WHERE condition ]

    …..   other commands

END PROCESS SELECT
```

This will allow the use of Joins and the WHERE clause and access to all PQL conditions and expressions to return one named set of variables at a time. There are no plans to include aggregation , GROUP BY or UNION.

To close the presentation Randy talked about SIR Server.
This will be similar in concept to SQLServer in that it will be started on the network 'host server' machine, and it will wait for users to logon using their own 'client' workstations. The SIRServer will process requests and return the results as standard output to the users 'client' workstation.

e.g.

```
CONNECT  conid  SIRSERVER   'b1'
   USER  'myuserid'
   PASSWORD 'mypaswd'
   ERROR   'errid'

SEND REQUEST
     CONNECT  conid
     COMMAND  'myprocs.report1'
      RESULT    lines
```

The myprocs.report1 family contains the PQL program of the user that will be submitted to the SIRServer.

Adrian Hodgson

Annual General Meeting

Minutes of AGM held on 24th September 2004, London

Apologies for absence
None

Minutes of Previous Meeting
Agreed.

Matters Arising
None.

Chair's Report
We have held a UK Conference in September 2004 – this one. There had been two issues of Reporter produced (November 2003 and May 2004) – thanks to Kathy and Fran for this.

Dave reported that SIR 2002 is now version .35. There is also a new Unix update. A demo version of SIR XS is available.

Treasurer's Report
Randy gave us a statement of account. We have as of today £987.78 in the current account, £806.25 in the business reserve and £5038.16 in the Bonus Saver account giving an overall balance of £6832.19. The fees for the London conference amounted to £280 and the costs to £600

Motions
"That Article 4A of the Bylaws of the SIR UK Users Group be amended to read: A general meeting of the group shall be held at least once in **two** years. Written notification of the time and place shall be sent to all full members at least thirty days in advance".
This was proposed by Dave Doulton and seconded by Randy Banks. Jon Johnson argued that 2 years is too long to go without any SIR training. He accepted that it is constitutionally OK but that the intention should be to have a conference each year.
The motion was passed with two abstensions.

Elections
Jon Johnson volunteered to join the committee. There being no further nominations Jon Johnson was elected together with Randy Banks, Patrick Brown, Dave Doulton, Adrian Hodgson and John Lemon as members of the SIR UK User Group committee.

Any Other Business
None.

Frances Williams
Secretary

SIR Tips

Tips and Tricks using SIR 'Any Time' Commands

One of the best features of SIR PQL is its 'any time' commands. Using these can lead to code which is much easier to write and maintain.

Imagine a survey where the same group of people are interviewed every year and one of the questions that is asked is whether the respondent has obtained any educational qualification since they were last interviewed and if so, what it was. You then want to create a derived variable which holds the highest qualification that the respondent has. This will be either the highest qualification the respondent had last time they were interviewed (not necessarily the previous year) or this year's qualification if it is higher.
Imagine we are looking at the third year of the survey and that records and variables have the same name each year other than that they have a different year prefix – A for year 1, B for year 2 etc.

```
Retrieval update
Integer H_QUAL | temporary variable to hold
   highest qualification
.  process rec CEDREC
.     get vars PID      | personal identifier
   - remains the same each year
.     compute H_QUAL = CQUAL  |  variable
   containing this year's qualification will
   be '0' if none
.     old rec is BEDREC (PID)
.        ifthen (BHQUAL gt H_QUAL)
c           highest qual from last year is
   higher that this year's qual
.           compute H_QUAL = BHQUAL
.        end if
.     end rec
.     ifthen (system(16) eq 0)
c        not interviewed last year, try year
   one
.        old rec is AEDREC (PID)
.           ifthen (AHQUAL gt H_QUAL)
.              compute H_QUAL = AHQUAL
```

```
.           end if
.        end rec
.     end if
.     put vars CHQUAL = H_QUAL
c     update this year's highest
   qualification to date
.  end rec
end retrieval
```

This works fine, but each year it needs a lot of modification and imagine how long the code will become after several years!

```
GLOBAL UPDATE = UPDATE  | comment this out
   for testing purposes
GLOBAL YP = C  | the current year's prefix -
   C for year 3
GLOBAL RYP = $B A$   | reverse prefixes for
   all previous years
C
Retrieval <UPDATE>      | this will read
   'retrieval UPDATE if the global is defined
C             and 'retrieval' if it is
   commented out
Integer HQUAL
.  process rec <YP>EDREC   | the compiler
   expands this to CEDREC
.     get vars PID
.     compute H_QUAL = <YP>QUAL
.     begin
.        do repeat pref = <RYP>
c           the compiler expands this to pref
   = B A
.           old rec is pref!EDREC
c              the ! ensures that the
   compiler expands pref to give BEDREC first
   time through the loop and then AEDREC
.              ifthen (pref!HQUAL gt H_QUAL)
.                 compute H_QUAL = pref!HQUAL
.                 exit begin
c                 once we get H_QUAL we don't
   want to look at any other records so we
   jump out of the begin/end begin loop
.              end if
.           end rec
.        end repeat
.     end begin
.  end rec
C
CIF eq "<UPDATE>" "UPDATE"    |  in update
   mode
.  put vars <YP>!HQUAL = H_QUAL
c  this line will only be compiled if
   <UPDATE> is defined
CIF END
C
End retrieval
```

In this version of the code, the only modifications which are required each year is to change the value of the two globals, YP and RYP. So for year 4 you would have:

```
GLOBAL YP = D
GLOBAL RYP = $C B A$
```

And the rest of the code stands as is. If the global <UPDATE> is defined the code to be runs in updating mode; if it is commented out it runs in test (non-updating) mode.

Fran Williams

CIF tf

When I used to teach CIF commands on the Advanced SIR course I could never understand how CIF TF (means True or False) and CIF TRUE could be useful: I'd only ever used Cif, Cif false and Cif end.

But last year it came to me in a blinding flash…..

Until then I had a lot of code like:

```
Global UPD = 0 | Default to scan mode, change
   to 0 to 1 for update mode

Cif eq <UPD>,1
Retrieval update upstat
Cif false
Retrieval
Cif end

. Process cases
….. Commands various

Cif eq <UPD>,1
.   put vars ….
Cif end

. End case
End retrieval
```

And some of these programs had loads of Cif blocks because they did complicated stuff. Now they look much neater with just one Cif block:

```
Cif eq <UPD>,1
Retrieval update upstat
Cif false
Retrieval
Cif tf

. Process cases
….. Commands various

Cif true
.   put vars ….
Cif tf

. End case
End retrieval
Cif end
```

Enjoy!

Kathy Brooks

SIR UK User Group Committee Members 2004/2005

Randy Banks (Treasurer)
ISER
University of Essex
Wivenhoe Park
Colchester CO4 3SQ
Tel: 01206 873067
Fax: 01206 873151
email: randy@essex.ac.uk

Kathy Brooks (Reporter)
Forvus
53 Clapham Common
South Side
London SW4 9BX
Tel: 020 7819 1012
Fax: 020 7819 1010
email: kathy@forvus.co.uk

Patrick Brown
Institute of Occupational Medicine
Research Park North
Riccarton
Edinburgh EH14 4AP
Tel: 0131 449 8031
Fax: 0870 850 5132
email: Patrick.Brown@iomhq.org.uk

Dave Doulton (Chair)
University of Southampton
Computing Services
Highfield
Southampton SO17 1BJ
Tel: 023 8059 3541
Fax: 023 8059 3131
email: D.C.Doulton@soton.ac.uk

Adrian Hodgson
ORC International
5th Floor City Point
701 Chester Road
Stretford
Manchester M32 0RW
Tel: 0161 877 6781 (switchboard)
Fax: 0161 872 3997
Email: adrian.hodgson@orc.co.uk

Jon Johnson
Room 641
Centre for Longitudinal Studies
Institute of Education
20 Bedford Way
London WC1H 0AL
Tel: 020 7612 6571
Fax: 020 7612 6686
Email: jj@cls.ioe.ac.uk

John S. Lemon
Aberdeen University Computing Centre
Edward Wright Building
Dunbar Street
Aberdeen AB24 3QY
Tel: 01224 273350
Fax: 01224 273372
Email: j.s.lemon@abdn.ac.uk

Mo Reardon/Tony Reardon
SIR PTY LTD
312 Mona Vale Road
Terrey Hills
NSW 2084, Australia
Tel: 00612 9450 2354
Fax: 00612 9475 1430
email: mo@sir.com.au
email: tony@sir.com.au

Frances Williams (Secretary)
ISER
University of Essex
Wivenhoe Park
Colchester CO4 3SQ
Tel: 01206 873568
Fax: 01206 873151
email: fwill@essex.ac.uk

You can email the whole committee by addressing your message to **sug.comm@isermail.essex.ac.uk**