

Buzzlock Software: Share a SAS Data Set without SAS/Share®

Phil Busby, President, Live Data Systems, Inc.
Research Triangle Park, North Carolina

Buzzlock Software™ is a concurrent access utility for SAS data sets or other computer system resources. It was written with SAS® and SAS/AF® software version 6.11 under Windows, encrypted with Interbuz® Software version 2.0, and transported to a remote client, a major airline, running SAS/AF® Software version 6.08 under MVS. This paper addresses three areas of interest to SAS users that are illustrated by this application:

- The concurrent access of SAS data set observations is controlled by the Buzzlock program screens and a simple SAS data set in a SAS data library writable by all users;
- Specific techniques of developing software on one platform that is intended to run on another platform are illustrated;
- The transport of SAS data sets and SAS/AF catalogs to the remote site is achieved in a secure and simple manner.

The Problem

A major airline's cargo business has grown to the point where it takes several full-time analysts to manage inventory on over 2000 flights a day. These flights are represented by two sequential external files that are converted at various points into SAS data sets:

The Master flight data set is a control file containing one observation per scheduled flight. It is keyed by flight number, origin, destination, and day-of-week.

The Daily Flight data set contains one observation per flight on each day that the flight is scheduled. It is keyed by flight number, origin, destination, and flight date.

Customers book space for freight on a specific flight, subject to parameters from the Master record for that scheduled flight. Multiple analysts need to access and update these two files concurrently. If one analyst is updating one flight observation, that should not interfere with another analyst who is updating another flight at the same time.

The Existing System

The SAS/AF programs that allow on-line flight selection and update start by making a copy of the flight data set in the user's work library. Each update to a flight is saved in the work data set until the user exits, at which time only those flights that were modified are used to update the main Daily Flight data set. This works fine until two analysts decide to modify the same flight, at which time one's changes overwrite the other's.

The Master file update is even more restrictive: the entire file is converted to a SAS data set each time the user selects the Master file update function, and all changes to the Master file are written back to the external file at exit time. When a specific Master file record is selected for update, the Daily Flight records for that origin, destination, flight number, and day-of-week are also locked, because the Daily Flight update function relies on Master variable values.

The systems need to work this way because of the need to interface with other computer systems where the most dependable interface is a sequential external file. Furthermore, many analysts are familiar with text editors that have enabled them to edit these files directly, before the convenience and control under SAS/AF Software was possible. The appropriate solution is not to rewrite the whole system or require the analysts to change the way they work.

The Buzzlock Solution

On the following screen, the analyst has chosen the Flight Update function, and must now specify which flight to edit (see screen 1). When END is executed, the system must verify that the same flight is not being updated by someone else.

Enter the following data —

FLTN: 2____
 ORG: LAX
 DST: JFK

Press END or ENTER twice to activate
 or
 Press PF6 to quit and process all
 released flights.

Screen 1: Flight Selection

The analyst enters a flight on this screen and presses END; if the flight is not locked, he will go directly to the flight edit screen. The routine called is CHKLOCK, which verifies that there is no matching observation for the flight that is locked by a different user, and if not, it locks the flight for this user by writing an observation to the lock data set for this flight and user.

You may wonder what would happen if two users try to lock a flight at exactly the same time, such that both are trying to access the lock data set. In this situation, CHKLOCK displays its message screen:

Warning: Cannot open data set FLTLOCK.

Enter END to try again or CANCEL to exit.

Screen 2: FLTLOCK contention message

Since the FLTLOCK data set is small, it takes very little time to open it, apply a WHERE clause for a flight, do a fetch, write an

observation to it, close it, and free the libref. Still, users are aware of this possibility and know to just press END to try again.

In the case where there is already another user updating the flight, the user stays on the first screen, the Flight Selection screen, and there is a message at the bottom saying "Flight 2 LAX-JFK locked by xxxxxxxx." where the x's represent the userid of the person with the lock on the flight. The user can call the locking user and inquire when the flight will be unlocked, or more likely, he will just work on a different flight at the moment, and come back to the locked flight later.

Resource Locking

When the analyst is done with his updates to the Daily Flight data set, the system must apply the updates to the main data set from his WORK data set. It is now, and only now, that the system must access the Daily Flight data set itself to perform the update operation. The fast way to do this is a member lock on the entire data set for the duration of the data set update, which calls for a lock on the resource. For this purpose CHKLOCK takes a RESOURCE parameter which locks the entire data set. When another user tries to access the Daily Flight data set during this brief update operation, he gets the following screen:

Warning: Resource DAILYFLT
 locked by user N091145.

Enter END to try again or CANCEL to exit.

Screen 3: RESOURCE contention message

When two users decide to finish their updates, exit and apply them to the main data set at almost exactly the same time, the second user would see this message and either press END or CANCEL. Other functions could be performed while waiting for access to the resource for applying the updates. The updates are not lost unless the user exists out of the SAS session.

The Specs are Changing...

The first day in production, there was a contention problem with the Master file. The system must treat the entire file as a resource since the existing system reads and writes the entire file for each change to a record. Adds and deletes of Master records are possible during the edit of this file, and the system must check for day-of-week overlaps. Frequently one analyst would be editing a Master record when another analyst wanted to edit a different Master record. The second analyst had to wait until the first one had completely finished his update.

Flexible Resources

The solution to this problem was quickly implemented due to the flexibility of user-defined resources: Buzzlock Software locks on unique resource names, but does not know or care what they are. In this case, we simply redefined the Master file into four Master files by geographic region: Domestic, Pacific, Latin, and Atlantic. The four Master files are named masterD, masterP, masterL, and masterA, and split out at the start of the day by region. The end-of-day job merges them together again.

Contention is no longer a problem because the analysts divide the work by region, so their updates are within their own region's Master file. Each Master file update locks only the regional Master file.

From Windows to MVS

Procs CPORT and CIMPORT are the most straightforward way to convert SAS/AF catalogs across operating systems. The output transport file was encrypted using Interbuz Software version 2.0, producing a secured file to take to the remote site either by diskette or FTP. Note that the encrypted file's extension is .BUZ, but when it is decrypted the original .XPT extension is restored on the remote PC.

For SAS batch jobs, source code developed under Windows is written as variable-length records, and when converted to fixed-length records in a MVS partitioned data set, all leading blanks on every line are lost. To preserve indentation we simply started every line with a white-space comment:

```
/**/ PROC REPORT DATA=RPTDLY HEADSKIP PANELS=2 PSPACE=16 LS=165 PS=56
/**/ SPLIT='/' SPACING=1 MISSING NOWD;
/**/ TITLE "ALLOCATION RESULTS FROM CARGO - PRODUCTS 6-7";
/**/ COLUMN UNIDATE ORG DST FLTN
/**/ UNIEQP DPTR AVLPAYL PAYLFT BELLCAP BELLCUBE
/**/ VOL60 CAN60 CUBE60 VOL70 CAN70 CUBE70;
/**/ DEFINE UNIDATE / DISPLAY FORMAT=DAY2. "DT";
/**/ DEFINE ORG / DISPLAY FORMAT=$3. "ORG";
/**/ DEFINE DST / DISPLAY FORMAT=$3. "DST";
/**/ DEFINE FLTN / DISPLAY FORMAT=8. WIDTH=4 "FLNO";
/**/ DEFINE DPTR / DISPLAY FORMAT=8. WIDTH=4 "DPTR";
/**/ DEFINE AVLPAYL / DISPLAY FORMAT=8. WIDTH=6 "AVLPAY";
/**/ DEFINE PAYLFT / DISPLAY FORMAT=8. WIDTH=6 "LBSLFT";
/**/ DEFINE BELLCAP / DISPLAY FORMAT=8. WIDTH=5 "BLCAP";
/**/ DEFINE BELLCUBE / DISPLAY FORMAT=8. WIDTH=5 "BLCUB";
/**/ DEFINE VOL60 / DISPLAY FORMAT=8. WIDTH=6 "FUVOL";
/**/ DEFINE CAN60 / DISPLAY FORMAT=8. WIDTH=5 "FUCAN";
/**/ DEFINE CUBE60 / DISPLAY FORMAT=8. WIDTH=5 "FUCUB";
/**/ DEFINE VOL70 / DISPLAY FORMAT=8. WIDTH=6 "FLVOL";
/**/ DEFINE CAN70 / DISPLAY FORMAT=8. WIDTH=5 "FLCAN";
/**/ DEFINE CUBE70 / DISPLAY FORMAT=8. WIDTH=5 "FLCUB";
/**/ RUN;
```

The Libref/Fileref Problem

A bigger problem is testing on the PC: all of the librefs and filerefs are different. To simplify the conversion, we created a SAS macro variable in the startup program to define the operating system:

```
CALL SYMPUT('OPSYS', 'WINDOWS');
```

Then, throughout the SCL code, library assignments look like this:

```
LENGTH OPSYS $ 8;
OPSYS = SYMGET('OPSYS');
IF OPSYS = 'WINDOWS'
THEN NSLIB = 'c:\noshow\';
ELSE NSLIB = 'CARGO.NOSHOW.SAS';
SUBMIT CONTINUE;
  LIBNAME NSLIB '&NSLIB';
ENDSUBMIT;
```

This later was refined to set all of the filerefs and librefs directly in the startup program:

```
IF OPSYS = 'WINDOWS' THEN
  CALL SYMPUT('NSLIB', 'c:\noshow\');
ELSE CALL SYMPUT('NSLIB',
  'CARGO.NOSHOW.SAS');
```

followed by this construct in later SCL code:

```
LENGTH NSLIB $ 64;
NSLIB = SYMGET('NSLIB');
SUBMIT CONTINUE;
  LIBNAME NSLIB '&NSLIB';
ENDSUBMIT;
```

Unit Testing Made Easy

To test the locking functionality on a one-user system, we just edited the FLTLOCK data set directly, simulating the existence of resource locks by other users by creating appropriate observations in the lock data set.

Of course, the test data sets and files still needed to be ported to the development machine from the remote site. In the case of SAS data sets or binary files from one PC to another, however, these files were compressed and encrypted directly by Interbuz Software, without conversion into transport files or UU-encoded printable characters.

Conclusion

The SAS/AF program screens and SAS data set of Buzzlock Software allow dynamic locking of specific records in this example of an airline's flight data base. This solution fits the needs of a group of about a dozen analysts who divide the work by region. The software development process illustrated security and portability techniques applicable to the development of software for a remote site running under a different operating system. The system is easily expanded to allow concurrent access of any system resource.

SAS and SAS/AF are registered trademarks of SAS Institute Inc. in the USA and other countries.

® indicates USA registration.

INTERBUZ is a registered trademark of Live Data Systems, Inc.

Other product names are trademarks of their respective companies.

Phil Busby
Live Data Systems, Inc.
P.O. Box 13144
Research Triangle Park, NC 27709
800/252-3102
livedata@unidial.com
<http://www.interbuz.com>