Changes to GSM Lock Table Handling

1. Introduction

This document describes a number of changes to the GSM (Windows NT) File Executive, some system sub-routines and system utilities to improve the Lock Table handling. All the File Executive changes described in this note refer to the GSM (Windows NT) implementation. Although these changes could be ported to GSM (Unix) and the GSM (Novell) NLM, it is highly unlikely, and undesirable, to port the changes to the "real mode" +J0EA00 File Executive. Note also, only “BOS/XLAN” (sic) configurations are considered (some of the block sizes quoted below differ between XLAN and non-XLAN configurations).

2. Increasing the number of Lock Table Entries

The limit on the number of Lock Table (LK) entries is currently 200. This limit is imposed by Global Configurator (note that $CUSA limits the number of Lock Table entries to 198). However, the limit is completely artificial; it was imposed to restrict the data returned by the Read Lock Table operation (FDOPC = 25) to fit in a Arcnet buffer.

The Steering Routine module dynamically allocates the Lock Table. Each Lock Table entry is 10 bytes in length. The number of entries is specified in the BTLOCK Field and moved to AZNLKS by the Steering Routine. The table is terminated by 2 bytes containing 0xFE. A pointer to the Lock Table is established in the AZLK field within the File Executive data. This pointer is used by the File Executive, which tests the 2 bytes of 0xFE to check for the end of the table. Although the table size is held in the AZ-block, in AZNLKS, this field is not currently used by the File Executive. Therefore, as far as the Steering Routine and File Executive are concerned the maximum size of the Lock Table is restricted only by the limits of the Memory Allocation function (although this is effectively reduced to a limit of 32Kb-1 by some internal field size limitations).

However, a number of GSM utilities (e.g. $U and $STATUS) invoke the Read Lock Table operation to read the entire Lock Table into a fixed size memory block (typically a PIC X(2048)) with the Data Division. $STATUS uses this operation to expedite the LOC command; $U appears to use this innocuous operation to test for the presence of remote File Executive. The presence of these fixed sized blocks effectively restrict the size of the Lock Table to the following number of entries:

(2048 – 2) / 10 rounded down = 204

The GSM (NT) LAN Executive interface can handle data blocks larger than 2048.

Two changes to the File Executive will be required to increase the number of Lock Table entries beyond 198. Firstly, the existing Read Lock Table operation must be modified to return just the first N entries (where N =< 204, I propose that N=200). This is required to prevent the existing $STATUS and $U from crashing when the Lock Table is extended beyond 2048 bytes. Note that the Data Division of the program will not be overwritten, because a Block Length of 2048 is correctly established in FDBLO. However, the terminator word of 0xFEFE will not be present in the returned Lock Table and SEARCH statements will produce unpredictable results (i.e. typically failing to display locks).

Note that there is a complication with this technique: The File Executive adds entries to the Lock Table starting from the end, so the first 200 entries are likely to be the least interesting. When the File Executive truncates the results of the Read Lock Table operation to just 200 entries, the **last** 200 entries should be returned.

Secondly, an Extended Read Lock Table operation must be supported:

FDOPC	58
FDFAD	Segment of Lock Table to read

Each segment will consist of a maximum of 200 entries, terminated by a word of 0xFEFE. For example, if the Lock Table contains 500 entries, the following results will be returned:

FDOPC	*FDFAD*	*Results*
25	don’t care	Last 200 entries terminated by 0xFEFE
58	0	First 200 entries terminated by 0xFEFE
58	1	Next 200 entries terminated by 0xFEFE
58	2	Last 100 entries terminated by 0xFEFE
58	> 3	Just a word of 0xFEFE

No changes will be required to $U, which can continue to use the Read Lock Table operation to “finger” a file server. However, the code in $STATUS should be changed to:

```
Move 0 to FDFAD
Move 58 to FDOPC
Call LAN/File Executive
ON EXCEPTION
    Move 25 to FDOPC
    Call LAN/File Executive
```

```
        Display possibly truncated Lock Table
    ELSE
        DO WHILE 1st word of Lock Table <> 0xFEFE
            Display up to 200 entries
            ADD 1 TO FDFAD
            Call LAN/File Executive
        ENDDO
    ENDIF
```

Changes to CFUPDATE and $CUSA will be required to remove the limitations for GSM (Windows NT) configuration files. Note that these changes will not be important when the configuration date has been moved to the Windows registry.

Note that the special Return Control Block operation, used by FEDIAG to return a specific entry in the LK block, and $USAGE to monitor the control block usage, will not be affected by these changes.

3. Associating the User Number & Node-id with Shared Locks

The File Executive does not currently support a “Get Shared Lock” operation. A Shared Lock is distinguished from an Exclusive Lock by setting the User Number to 0. This is done by an explicit, and highly dangerous, “MOVE 0 to $$USER” statement within the SLOCK$ and SULOC$ sub-routines. The reasons for this methodology have been lost in the mists of time (and GSM/BOS history) but the technique of “collapsing” all the User Numbers for a Shared Lock to a single user number (i.e. user number 0) does simplify the Lock Table manipulation routine (EABX in the File Executive).

In order to differentiate between a Shared Lock and a genuine lock by user 0, there is code in the EABX routine of the File Executive to map the Channel Number, from FDCHA in the FD, to a User Number via the associated DC-block. However, this code is not totally reliable because the FDCHA field is the modulo 250 channel number. Consequently, each 250th entry in the DC-block array must be tested until the FDDEV and FDBLO fields in the current FD match the current Lock Table entry (NB. The FDBLO field holds the file label number which is held into the Lock Table).

In a pathological case, where user number *A* has a Shared Lock asserted on a file opened on Channel Number *X*; and user number *B* is trying to assert a Shared Lock the same file opened on Channel Number (*X*+250) the "is this a genuine user 0" algorithm will return unpredictable results because the 1st DCUSER field encountered will contain *A* rather than the expected *B*. This is not normally a problem because a simple Boolean test is performed on the DCUSER field and a User Number of 0 is only encountered when specialised programs (i.e. relocatable intercept routines) are issuing File Executive operations. However, the algorithm that maps a Channel Number back to a User Number is not accurate enough to be used for general use when **different** non-zero User Numbers must be differentiated.

Consequently, new File Executive Shared Lock operations will be required that use the real, unadulterated User Number. The following operations must be added to the File Executive:

Get Shared Lock	FDOPC	62
	FDFAD	Lock region
	FDBLOLabel number	
	FDDEV	Device number
	FDCHA	Channel number
	FDUSER	User number
Release Shared Lock	FDOPC	64
	FDFAD	Lock region
	FDBLOLabel number	
	FDDEV	Device number
	FDCHA	Channel number
	FDUSER	User number

Important note: There is no requirement for a Shared Lock/Wait operation.

Crucial to the implementation of these new operations is a new Shared Lock control block, SK, that maps an LK-block index to a User Number, Node-id pair. The format of the SK-block is as follows:

```
Int       lkindex;       /* Index of LK entry */
bos_b     usernum;       /* User number */
bos_b     skfiller;      /* Filler - not used */
bos_w     nodeid;        /* Node-id */
```

The relationship between the LK and SK blocks is similar to that between the DF and DC blocks (i.e. a single LK/DF block is linked to multiple SK/DC blocks via a primary block index number).

For an Exclusive Lock entry in the Lock Table, both the User Number and Node-id are held in the Lock Table so no linked SK-block is required.

For a Shared Lock entry in the Lock Table, the User Number is 0 and the Node-id is set to the Node-id of the 1st user who asserted the Shared Lock. For every user who has asserted a Shared Lock, an SK-block is used to hold the actual User Number and Node-id. A simple LK-block index number within the SK-block allows an SK-block to be mapped to its parent LK-block; and an LK-block to be mapped to one, or more, child SK-blocks.

The new Get Shared Lock and Release Shared Lock operations will perform similar functionality to the existing Lock and Unlock operations but will assume a User Number of 0 when considering the LK-block; and move the real User Number, from FDUSER, only when considering the SK-block. **Important Note**: Internally mapping the real FDUSER to a User Number of 0 when considering the LK-block is vital to maintain backwards compatibility with the existing "kludged" Shared Lock handling. Other schemes, such as holding a Shared Lock vs. Exclusive Lock flag in the LK-block, could lead to incompatibilities and complications when implementing the Read Lock Table operation.

3.1 New Registry options to support the new Shared Lock handling

Two new options in the registry have been added:

+NumberOfSharedLockTableEntries

This option simply specifies the number of entries in the Extended Shared Lock table (i.e. the
Number of entries in the SK-block). This option is logically part of the Configuration Information (note the "+" prefix) but has been added directly to the registry to adhere to the conventions described in IN181. Note that this option occurs in 3 places in the registry hierarchy (once under the Global Client key, and twice under the Global Server key - see IN181 for further details).
The default value for this registry setting is 2000.

ExtendedSharedLocks

This option informs the client software that Extended Shared Locks are supported by the Global Client **AND ALL GLOBAL SERVERS WHICH IT ACCESSES**. Enabling this option has the effect of setting the #80 bit in the SYBIF1 flag in the System Area. This flag should be examined by the Shared Lock routines to decide which type of Shared Lock operation (i.e. old or new) to invoke.

The default value for this registry setting is “Off”.

The following combinations of client and server options are possible:

Global Client ExtendedSharedLocks	Global Server NumberOfSharedLockTableEntries	
Off	0	Normal Shared Locks
Off	N	Normal Shared Locks (the SK-block will be unused)
On	0	Extended Shared Locks (these must degenerate to act as Normal Shared Locks)
On	N	Extended Shared Locks

3.2 Adding a new Shared Lock entry to the LK-block

When a new LK-block entry is used to add a Shared Lock (i.e. with a current usage count of 0), the following extra SK-block manipulation must be included:

- Clear all entries in the SK-block for the current LK-index (an SK-block entry is maked as "not in use" by setting sk->lkindex fo 0xFFFFFFFF);
- Find the 1st free entry (sk->lkindex = 0xFFFFFFFF) in the SK-block;
- Move the current LK-block index to sk->lkindex;
- Move FDUSER to sk->usernum;
- Move the Node-id to sk->nodeid.

Important Note 1: This new logic will only be enabled if the +NumberOfSharedLockTableEntries is non-zero.

Important Note 2: Any failure to add an entry to the SK-block will return an error "T" i.e. TOO MANY OPEN FILES.

3.3 Adding a existing Shared Lock entry to the LK-block

When an existing LK-block entry is used to increase the usage count of a Shared Lock (i.e. with a current nonzero usage count), the following extra SK-block manipulation must be included:

- Find the 1st free entry (sk->lkindex = 0xFFFFFFFF) in the SK-block;
- Move the current LK-block index to sk->lkindex;
- Move FDUSER to sk->usernum;
- Move the Node-id to sk->nodeid.

3.4 Removing an existing Shared Lock entry from the LK-block

When an existing LK-block entry is used to decrease the usage count of a Shared Lock the following extra SK-block manipulation must be included:

- Scan the SK-block searching for an entry with the following parameters:

sk->lkindex	Current LK-block index;
sk->usernum	FDUSER
sk->nodeid	Node-id

- If an entry with those parameters can't be found then some type of internal error has occurred and no further action should taken;
- If an entry is found then it should be removed from the SK-block by setting sk->lkindex to 0xFFFFFFFF.

4. Removing Shared Locks on File Closes

The Lock Table (i.e. LK-block) is accessed by the Close File operation. Because this operation simply scans the LK-block for matching User Numbers, Node-id's and File Block parameters, a Close will never remove Shared Locks (i.e. where the User Number is 0).

The Close File operation must be modified to process Shared Locks (i.e. when LKUSER = 0) in a special manner: The SK-block must be scanned to looking for matching LK-index, User and Node-id parameters. When such an entry is found, the SK-block entry must be removed, Lock Count in the LK-block must be decreased, and the Lock removed if the count reaches zero.

5. Removing Shared Locks on $MONITOR Resets and $STATUS Restarts

The Lock Table (i.e. LK-block) is accessed by Reset User operation. Because this operation simply scans the LK-block for matching User Numbers and Node-id's a Reset User will never remove Shared Locks (i.e. where the User Number is 0).

The Reset User operation (when called by a $MONITOR reset and by a $STATUS Restart) must be modified to process Shared Locks (i.e. when LKUSER = 0) in a special manner: The SK-block must be scanned to looking for matching LK-index, User and Node-id parameters. When such an entry is found, the SK-block entry must be removed, the Lock Count in the LK-block must be decreased, and the Lock removed if the count reaches zero.

6. Returning Shared Lock information to $STATUS

The current Read Lock Table operation issued by $STATUS will not return the Shared Lock information. Consequently, the following new File Executive operation will be required to access entries from the SK-block:

FDOPC	59
FDBLO2	
FDMEM	Pointer to 2 byte buffer
FDFAD01	LK-index value (1 to N) i.e. based on 1 **NOT** 0
FDFAD23	First/next count (start from 0)

If the operation is successful, the single-byte User Number and single-byte Node-id will be returned to the buffer.

The following errors may be returned in FDRES:

'N'	LK-index too high
'O'	No SK-block allocated
'8'	SK-block first/next count too high
'9'	LK-block not in use

The logic in $STATUS should be of the following format:

```
Read Lock-Table (possibly in several segments, see above)
FOR every "in-use" LK-block entry
        Move LK-block index to FDFAD01 (based on 1)
        DO FOR FDFAD23 = 0 TO 9999
                Invoke File Exec operation 59
                ON EXCEPTION
                    IF FDRES = '8' FINISH       * Normal termination
                    Handle hard-error (may be ignored)
                ELSE
                    Convert User Number/Node to User Name/Partition
                END
        ENDDO
ENDDO
```

7. Returning User Number & Node-id when a Lock Operation fails

In addition to displaying the list of users with a Shared Lock from within $STATUS, the logic in the 32-bit Speedbase Database Manager that displays the LOCK message must also display the User Name/Partition with an Exclusive Lock, or the first User Name/Partition with a Shared Lock. This will be achieved using the new GETLK$ routine (see SJ248).

The GETLK$ routine will require the following new File Executive operation to be supported:

FDOPC	63
FDBLO	As for LOCK operation
FDDEV	As for LOCK operation
FDFAD	As for LOCK operation
User No.	As for LOCK operation
Node-id	As for LOCK operation

On successful completion, the following information is returned via the FD:

FDFAD0	User Number with the Lock (or 1st with a Shared Lock)
FDFAD1	Node-id with the Lock (or 1st with a Shared Lock)
FDFAD2	0 = Shared Lock
	1 = Exclusive Lock

If the Lock Table entry is not in use, an error '7' will be returned. Note that this can happen if an Unlock, issued by another User/Node, clears the Lock, or Shared Lock, between the previous failed Lock operation and the Get Lock Status operation.

An error '7' will also be returned if no SK-block is allocated or an entry with the required LK-block index can't be found in the SK-block.

8. Internal changes to the Lock Table

During the implementation two changes have been made to the structure of the Lock Table, LK-block.

An additional 32-bit LK-block index field (i.e. a value ranging from 0 to N-1, where N is the number of Lock Table entries) has been added to the LK-block. This has been added to avoid implementing non-portable (particularly to 64-bit architectures) pointer arithmetic in "C".

The size of the Lock Table User Count field has been increased from 1 byte to 4 bytes (i.e. bos_b to int). Unfortunately, this field is returned, as a signed-byte value, to $STATUS by the Read Lock Table operation. Prior to these changes, any attempt to increment the User Count in the Lock Table beyond 127 would result in an error "T". The field was returned to $STATUS unmodified (i.e. a byte value between 1 and 127). Since the File Executive changes, the User Count field can increase to 2**31 (i.e. effectively unlimited). When the field is returned to $STATUS by the Read Lock Table operation(s) (i.e. see section 2) it is rounded down to 127.
The issue that the $STATUS LOC operation with report a Lock Count of 127 if the Lock Count is 127, **or higher**, is not considered a serious problem. The $STATUS Extended Lock Table command (i.e. LKE) will always report an accurate list of Shared Locks).

9. Issues with the SYLANF flag

The $STATUS RES and CAN commands attempt to close all the files and clear all the locks for the target user. This is achieved by issuing a File Executive Reset operation, for each partition for that user, on all the file-servers that have been accessed by those partitions. This technique uses the partition-specific SYLANF PIC 9(9) COMP variable that is held in the System Area for each partition. The SYLANF flag contains a bit set for each node, between "A" to "Z", that has been accessed by a particular partition. This flag is used to avoid attempting Reset operations on servers that have not been accessed and/or are non-existent. However, for reasons that are far from clear, the actual SYLANF used by the $STATUS RES and CAN commands is the flag for the partition running $STATUS instead of the various partitions of the User being restarted.

Rather than modify $STATUS to use the appropriate SYLANF flag (which is technically very difficult), a new option has been introduced to hold this variable, on a partition basis, within GLOBAL.EXE. This new option is enabled by setting the following registry setting:

```
..\Global\Client\IgnoreSYLANF
```

Setting this registry option to "on" will cause the LAN handling within GLOBAL.EXE to use its local, and accurate, set of "Server Accesses Flags" rather than the inaccurate set passed via $STATUS.

10. Automatic Resets on Client Connect, Disconnect and Reconnect

None of the advances described above handle the situation that occurs when a Global Client disconnects unexpectedly from one, or more, Global Servers. Unless a "clean" $BYE is expedited there is a risk of leaving locks and open files on the server(s) that have been accessed by that Global Client.

A new option has been added to GLSERVER to automatically "Reset" all the users of a specified Global Client when that Global Client connects to, reconnect to, or disconnects from the Global Server.

The following registry options may be enabled in any combination:

..\Global\Servers\x\FileExecResetOnConnection
Automatically reset all users for node-id *y* when the Global Client for node-id *y* connects to server *x*.

..\Global\Servers\x\FileExecResetOnReconnection
Automatically reset all users for node-id *y* when the Global Client for node-id *y* explicitly reconnects to server *x*.
..\Global\Servers\x\FileExecResetOnDisconnection
Automatically reset all users for node-id *y* when the Global Client for node-id *y* disconnects from server *x*.

The following options apply to all servers that do not have an equivalent server-specific option:

..\Global\Servers\FileExecResetOnConnection
..\Global\Servers\FileExecResetOnReconnection
..\Global\Servers\FileExecResetOnDisconnection

All client restarts are logged in the Global Server log file.