Macintosh Technical Notes

Developer Technical Support

#305: PBShare, PBUnshare, and PBGetUGEntry

Revised by: Jim Luther
January 1992
Written by: Jim Luther
October 1991

This Technical Note documents three new File Manager routines available on shared local volumes. The Pascal glue code, C glue code, and the assembler equates and macros for the calls are included in this note.

Changes since October 1991: Corrected `ioObjType` input values for `PBGetUGEntry`.

Three new File Manager routines, `PBShare`, `PBUnshare` and `PBGetUGEntry` are available on local volumes that have File Sharing enabled. These three routines are necessary to implement a "Sharing" dialog used to make a volume or directory a "share point" on the network and to set the Owner and User/Group of a shared folder. (For a description of share points, see Macintosh Technical Note #301.) The `PBShare` routine makes a volume or folder a share point. The `PBUnshare` routine undoes the effect of `PBShare`; it makes a share point unavailable on the network. The `PBGetUGEntry` routine lets you access the list of User and Group names and IDs on the local file server.

File Sharing should be on and the volume should be sharable before you call these three routines. You can check to see if File Sharing is turned on

and that the local volume is sharable by calling the `PBHGetVolParms` routine and checking the `bHasPersonalAccessPrivileges` (local File Sharing is enabled) bit returned in the `vMAttrib` field of the `GetVolParmsInfoBuffer`. File Sharing is turned on if local File Sharing is enabled on any mounted volume. A portion of a volume can be shared only if local File Sharing is enabled on that volume. The following two functions can be used for these checks:

```
FUNCTION VolIsSharable (vRefNum: Integer): Boolean;
{See if local File Sharing is enabled on the volume specified by vRefNum}
  VAR
    pb: HParamBlockRec;
    infoBuffer: GetVolParmsInfoBuffer;
    err: OSErr;
BEGIN
  WITH pb DO
    BEGIN
      ioNamePtr := NIL;
      ioVRefNum := vRefNum;
      ioBuffer := @infoBuffer;
      ioReqCount := SizeOf(infoBuffer);
    END;
  err := PBHGetVolParmsSync(@pb);
  IF err = noErr THEN
    IF BTst(infoBuffer.vMAttrib, bHasPersonalAccessPrivileges) THEN
      VolIsSharable := TRUE
    ELSE
      VolIsSharable := FALSE
  ELSE
    VolIsSharable := FALSE;
END;

FUNCTION SharingIsOn: Boolean;
{See if File Sharing is turned on by seeing if any volume has}
{local File Sharing enabled}

  VAR
    pb: HParamBlockRec;
    err: OSErr;
    volIndex: Integer;
    sharing: Boolean;

BEGIN
  sharing := FALSE; {assume File Sharing is off}
  volIndex := 1;
  REPEAT
    WITH pb DO
      BEGIN
        ioNamePtr := NIL;
        ioVolIndex := volIndex;
      END;
    err := PBHGetVInfoSync(@pb);
    IF err = noErr THEN
      sharing := VolIsSharable(pb.ioVRefNum);
    volIndex := volIndex + 1;
  UNTIL (err <> noErr) OR sharing; {stop if error or if a volume has}
                                   {local File Sharing enabled}
  SharingIsOn := sharing;
```

```
END;
```

The Routines

Assembly-Language Note: These routines are called through the `_HFSDispatch` macro with register `A0` pointing to the parameter block and register `D0` containing a routine selector. When your completion routine is called, register `A0` points to the parameter block of the asynchronous call and register `D0` contains the result code. See *Inside Macintosh* Volume IV, pages IV-115 through IV-119, for detailed information.

PBShare

```
FUNCTION PBShare (paramblock: HParmBlkPtr; async: Boolean) :
    OSErr;
```

Trap Macro `_Share`
Routine selector $42

Parameter Block

→	12	`ioCompletion`	long	pointer to completion routine
←	16	`ioResult`	word	result code
→	18	`ioNamePtr`	long	pointer to directory name
→	22	`ioVRefNum`	word	volume specification
→	48	`ioDirID`	long	parent directory ID

`PBShare` makes the directory pointed to by the `ioNamePtr`/`ioDirID` pair on the volume specified by `ioVRefNum` a share point.

Field descriptions

`ioCompletion` Longword input pointer: A pointer to the completion routine.

`ioResult` Word result value: Result code.

`ioNamePtr` Longword input pointer: Points to the directory name, or NIL if `ioDirID` is the directory ID.

`ioVRefNum` Word input value: The volume specification (volume reference number, working directory reference number, drive number, or 0 for default volume).

`ioDirID` Longword input value: The directory or parent directory specification.

Result codes

`noErr`	0	No error
`tmfoErr`	-42	Too many share points
`fnfErr`	-43	File not found
`dupFNErr`	-48	There is already a share point with this name
`paramErr`	-50	This function is not supported
`dirNFErr`	-120	Directory not found
`afpAccessDenied`	-5000	This folder cannot be shared
`afpObjectTypeErr`	-5025	Object was a file, not a directory
`afpContainsSharedErr`	-5033	The directory contains a share point
`afpInsideSharedErr`	-5043	The directory is inside a shared directory

Pascal glue code for `PBShare`:

```
FUNCTION PBShare (paramBlock: HParmBlkPtr; async: BOOLEAN): OSErr;
  INLINE $101F,  { MOVE.B     (A7)+,D0   }
         $205F,  { MOVEA.L    (A7)+,A0   }
         $6606,  { BNE.S      *+$0008    }
         $7042,  { MOVEQ      #$42,D0    }
         $A260,  { _FSDispatch,Immed     }
         $6004,  { BRA.S      *+$0006    }
         $7042,  { MOVEQ      #$42,D0    }
         $A660,  { _FSDispatch,Sys,Immed }
         $3E80;  { MOVE.W     D0,(A7)    }

FUNCTION PBShareSync (paramBlock: HParmBlkPtr): OSErr;
  INLINE $205F,  { MOVEA.L    (A7)+,A0   }
         $7042,  { MOVEQ      #$42,D0    }
         $A260,  { _FSDispatch,Immed     }
         $3E80;  { MOVE.W     D0,(A7)    }

FUNCTION PBShareAsync (paramBlock: HParmBlkPtr): OSErr;
  INLINE $205F,  { MOVEA.L    (A7)+,A0   }
         $7042,  { MOVEQ      #$42,D0    }
         $A660,  { _FSDispatch,Sys,Immed }
         $3E80;  { MOVE.W     D0,(A7)    }
```

MPW C v3.1 glue code for `PBShare`:

```
pascal OSErr PBShare (HParmBlkPtr paramBlock, Boolean async)
  = {0x101F,   /* MOVE.B     (A7)+,D0   */
     0x205F,   /* MOVEA.L    (A7)+,A0   */
     0x6606,   /* BNE.S      *+$0008    */
     0x7042,   /* MOVEQ      #$42,D0    */
     0xA260,   /* _FSDispatch,Immed     */
     0x6004,   /* BRA.S      *+$0006    */
     0x7042,   /* MOVEQ      #$42,D0    */
     0xA660,   /* _FSDispatch,Sys,Immed */
     0x3E80};  /* MOVE.W     D0,(A7)    */

pascal OSErr PBShareSync (HParmBlkPtr paramBlock)
```

```
    = {0x205F,   /* MOVEA.L    (A7)+,A0   */
       0x7042,   /* MOVEQ      #$42,D0    */
       0xA260,   /* _FSDispatch,Immed     */
       0x3E80};  /* MOVE.W     D0,(A7)    */

  pascal OSErr PBShareAsync (HParmBlkPtr paramBlock)
    = {0x205F,   /* MOVEA.L    (A7)+,A0   */
       0x7042,   /* MOVEQ      #$42,D0    */
       0xA660,   /* _FSDispatch,Sys,Immed */
       0x3E80};  /* MOVE.W     D0,(A7)    */
```

MPW C v3.2 glue code for `PBShare`:

```
  pascal OSErr PBShare (HParmBlkPtr paramBlock, Boolean async)
    = {0x101F,   /* MOVE.B     (A7)+,D0   */
       0x205F,   /* MOVEA.L    (A7)+,A0   */
       0x6606,   /* BNE.S      *+$0008    */
       0x7042,   /* MOVEQ      #$42,D0    */
       0xA260,   /* _FSDispatch,Immed     */
       0x6004,   /* BRA.S      *+$0006    */
       0x7042,   /* MOVEQ      #$42,D0    */
       0xA660,   /* _FSDispatch,Sys,Immed */
       0x3E80};  /* MOVE.W     D0,(A7)    */

  #pragma parameter __D0 PBShareSync(__A0)
  pascal OSErr PBShareSync (HParmBlkPtr paramBlock)
    = {0x7042,   /* MOVEQ      #$42,D0    */
       0xA260};  /* _FSDispatch,Immed     */

  #pragma parameter __D0 PBShareAsync(__A0)
  pascal OSErr PBShareAsync (HParmBlkPtr paramBlock)
    = {0x7042,   /* MOVEQ      #$42,D0    */
       0xA660};  /* _FSDispatch,Sys,Immed */
```

Assembler equate and macro for `_Share`:

```
  selectShare         EQU   $42

                      macro
                      _Share &async1,&async2
                            DoHFSDispatch selectShare,&async1,&async2
                      endm
```

PBUnshare

```
FUNCTION PBUnshare (paramblock: HParmBlkPtr; async: Boolean) :
      OSErr;
```

Trap Macro	`_Unshare`
Routine selector	$43

Parameter Block

→	12	`ioCompletion`	long	pointer to completion routine
←	16	`ioResult`	word	result code
→	18	`ioNamePtr`	long	pointer to directory name
→	22	`ioVRefNum`	word	volume specification
→	48	`ioDirID`	long	parent directory ID

`PBUnshare` makes the share point pointed to by the `ioNamePtr`/`ioDirID` pair on the volume specified by `ioVRefNum` unavailable on the network; it undoes the effect of `PBShare`.

Field descriptions

`ioCompletion` Longword input pointer: A pointer to the completion routine.

`ioResult` Word result value: Result code.

ioNamePtr	Longword input pointer: Points to the directory name, or NIL if ioDirID is the directory ID.
ioVRefNum	Word input value: The volume specification (volume reference number, working directory reference number, drive number, or 0 for default volume).
ioDirID	Longword input value: The directory or parent directory specification.

Result codes

noErr	0	No error
fnfErr	-43	File not found
dirNFErr	-120	Directory not found
afpObjectTypeErr	-5025	Object was a file, not a directory, or this directory is not a share point

Pascal glue code for PBUnshare:

```
FUNCTION PBUnshare (paramBlock: HParmBlkPtr; async: BOOLEAN): OSErr;
  INLINE $101F,  { MOVE.B     (A7)+,D0   }
         $205F,  { MOVEA.L    (A7)+,A0   }
         $6606,  { BNE.S      *+$0008    }
         $7043,  { MOVEQ      #$43,D0    }
         $A260,  { _FSDispatch,Immed     }
         $6004,  { BRA.S      *+$0006    }
         $7043,  { MOVEQ      #$43,D0    }
         $A660,  { _FSDispatch,Sys,Immed }
         $3E80;  { MOVE.W     D0,(A7)    }

FUNCTION PBUnshareSync (paramBlock: HParmBlkPtr): OSErr;
  INLINE $205F,  { MOVEA.L    (A7)+,A0   }
         $7043,  { MOVEQ      #$43,D0    }
         $A260,  { _FSDispatch,Immed     }
         $3E80;  { MOVE.W     D0,(A7)    }

FUNCTION PBUnshareAsync (paramBlock: HParmBlkPtr): OSErr;
  INLINE $205F,  { MOVEA.L    (A7)+,A0   }
         $7043,  { MOVEQ      #$43,D0    }
         $A660,  { _FSDispatch,Sys,Immed }
         $3E80;  { MOVE.W     D0,(A7)    }
```

MPW C v3.1 glue code for PBUnshare:

```
pascal OSErr PBUnshare (HParmBlkPtr paramBlock, Boolean async)
  = {0x101F,   /* MOVE.B     (A7)+,D0   */
     0x205F,   /* MOVEA.L    (A7)+,A0   */
     0x6606,   /* BNE.S      *+$0008    */
     0x7043,   /* MOVEQ      #$43,D0    */
     0xA260,   /* _FSDispatch,Immed     */
     0x6004,   /* BRA.S      *+$0006    */
     0x7043,   /* MOVEQ      #$43,D0    */
     0xA660,   /* _FSDispatch,Sys,Immed */
     0x3E80};  /* MOVE.W     D0,(A7)    */

pascal OSErr PBUnshareSync (HParmBlkPtr paramBlock)
  = {0x205F,   /* MOVEA.L    (A7)+,A0   */
     0x7043,   /* MOVEQ      #$43,D0    */
     0xA260,   /* _FSDispatch,Immed     */
     0x3E80};  /* MOVE.W     D0,(A7)    */

pascal OSErr PBUnshareAsync (HParmBlkPtr paramBlock)
  = {0x205F,   /* MOVEA.L    (A7)+,A0   */
     0x7043,   /* MOVEQ      #$43,D0    */
     0xA660,   /* _FSDispatch,Sys,Immed */
     0x3E80};  /* MOVE.W     D0,(A7)    */
```

MPW C v3.2 glue code for PBUnshare:

```
pascal OSErr PBUnshare (HParmBlkPtr paramBlock, Boolean async)
  = {0x101F,   /* MOVE.B     (A7)+,D0   */
     0x205F,   /* MOVEA.L    (A7)+,A0   */
     0x6606,   /* BNE.S      *+$0008    */
     0x7043,   /* MOVEQ      #$43,D0    */
     0xA260,   /* _FSDispatch,Immed     */
     0x6004,   /* BRA.S      *+$0006    */
     0x7043,   /* MOVEQ      #$43,D0    */
     0xA660,   /* _FSDispatch,Sys,Immed */
     0x3E80};  /* MOVE.W     D0,(A7)    */

#pragma parameter __D0 PBUnshareSync(__A0)
pascal OSErr PBUnshareSync (HParmBlkPtr paramBlock)
  = {0x7043,   /* MOVEQ      #$43,D0    */
     0xA260};  /* _FSDispatch,Immed     */

#pragma parameter __D0 PBUnshareAsync(__A0)
pascal OSErr PBUnshareAsync (HParmBlkPtr paramBlock)
  = {0x7043,   /* MOVEQ      #$43,D0    */
     0xA660};  /* _FSDispatch,Sys,Immed */
```

Assembler equate and macro for `_Unshare`:

```
selectUnshare         EQU   $43

                    macro
                    _Unshare &async1,&async2
                          DoHFSDispatch selectUnshare,&async1,&async2
                    endm
```

PBGetUGEntry

```
FUNCTION PBGetUGEntry (paramblock: HParmBlkPtr; async: Boolean) :
      OSErr;
```

Trap Macro	`_GetUGEntry`
Routine selector	$44

Parameter Block

→	12	`ioCompletion`	long	pointer to completion routine
←	16	`ioResult`	word	result code
→	26	`ioObjType`	word	object type function code
→	28	`ioObjNamePtr`	long	ptr to returned user/group name
↔	32	`ioObjID`	long	user/group ID

`PBGetUGEntry` asks the local file server for the next user or group in its list. `PBGetUGEntry` returns the user or group name and the user or group ID.

Field descriptions

`ioCompletion` Longword input pointer: A pointer to the completion routine.

`ioResult` Word result value: Result code.

`ioObjType` Word input value: Determines the type of object to be returned, as follows:

$0000	return next user
$FFFF	return next group

`ioObjNamePtr` Longword input pointer: Points to a result buffer where the user or group name is to be returned. If the pointer is NIL, then no name is returned. The name is returned as a Pascal string with a maximum size of 31 characters (Str31).

`ioObjID` Longword input/result value: The server will return the first user or group whose name is alphabetically next from the user specified by `ioObjID`. Setting `ioObjID` to 0 will return the first user or group. On return, `ioObjID` will be the user or group's ID.

You can enumerate the user or group list in alphabetical order by calling this routine again and again without changing the parameter block until the result code `fnfErr` is returned.

Result codes

`noErr`	0	No error
`paramErr`	-50	The `ioObjID` is negative or this function is not supported
`fnfErr`	-43	There are no more users or groups to return

Pascal glue code for `PBGetUGEntry`:

```
FUNCTION PBGetUGEntry (paramBlock: HParmBlkPtr; async: BOOLEAN): OSErr;
  INLINE $101F,  { MOVE.B     (A7)+,D0   }
         $205F,  { MOVEA.L    (A7)+,A0   }
         $6606,  { BNE.S      *+$0008    }
         $7044,  { MOVEQ      #$44,D0    }
         $A260,  { _FSDispatch,Immed     }
         $6004,  { BRA.S      *+$0006    }
         $7044,  { MOVEQ      #$44,D0    }
         $A660,  { _FSDispatch,Sys,Immed }
         $3E80;  { MOVE.W     D0,(A7)    }

FUNCTION PBGetUGEntrySync (paramBlock: HParmBlkPtr): OSErr;
  INLINE $205F,  { MOVEA.L    (A7)+,A0   }
         $7044,  { MOVEQ      #$44,D0    }
         $A260,  { _FSDispatch,Immed     }
         $3E80;  { MOVE.W     D0,(A7)    }

FUNCTION PBGetUGEntryAsync (paramBlock: HParmBlkPtr): OSErr;
  INLINE $205F,  { MOVEA.L    (A7)+,A0   }
         $7044,  { MOVEQ      #$44,D0    }
         $A660,  { _FSDispatch,Sys,Immed }
         $3E80;  { MOVE.W     D0,(A7)    }
```

MPW C v3.1 glue code for `PBGetUGEntry`:

```
pascal OSErr PBGetUGEntry (HParmBlkPtr paramBlock, Boolean async)
  = {0x101F,   /* MOVE.B     (A7)+,D0   */
     0x205F,   /* MOVEA.L    (A7)+,A0   */
     0x6606,   /* BNE.S      *+$0008    */
     0x7044,   /* MOVEQ      #$44,D0    */
     0xA260,   /* _FSDispatch,Immed     */
     0x6004,   /* BRA.S      *+$0006    */
     0x7044,   /* MOVEQ      #$44,D0    */
     0xA660,   /* _FSDispatch,Sys,Immed */
     0x3E80};  /* MOVE.W     D0,(A7)    */

pascal OSErr PBGetUGEntrySync (HParmBlkPtr paramBlock)
  = {0x205F,   /* MOVEA.L    (A7)+,A0   */
     0x7044,   /* MOVEQ      #$44,D0    */
     0xA260,   /* _FSDispatch,Immed     */
     0x3E80};  /* MOVE.W     D0,(A7)    */

pascal OSErr PBGetUGEntryAsync (HParmBlkPtr paramBlock)
  = {0x205F,   /* MOVEA.L    (A7)+,A0   */
     0x7044,   /* MOVEQ      #$44,D0    */
     0xA660,   /* _FSDispatch,Sys,Immed */
     0x3E80};  /* MOVE.W     D0,(A7)    */
```

MPW C v3.2 glue code for `PBGetUGEntry`:

```
pascal OSErr PBGetUGEntry (HParmBlkPtr paramBlock, Boolean async)
  = {0x101F,   /* MOVE.B     (A7)+,D0   */
     0x205F,   /* MOVEA.L    (A7)+,A0   */
     0x6606,   /* BNE.S      *+$0008    */
     0x7044,   /* MOVEQ      #$44,D0    */
     0xA260,   /* _FSDispatch,Immed     */
```

```
        0x6004,   /* BRA.S      *+$0006    */
        0x7044,   /* MOVEQ      #$44,D0    */
        0xA660,   /* _FSDispatch,Sys,Immed */
        0x3E80};  /* MOVE.W     D0,(A7)    */

#pragma parameter __D0 PBGetUGEntrySync(__A0)
pascal OSErr PBGetUGEntrySync (HParmBlkPtr paramBlock)
  = {0x7044,   /* MOVEQ      #$44,D0    */
     0xA260};  /* _FSDispatch,Immed     */

#pragma parameter __D0 PBGetUGEntryAsync(__A0)
pascal OSErr PBGetUGEntryAsync (HParmBlkPtr paramBlock)
  = {0x7044,   /* MOVEQ      #$44,D0    */
     0xA660};  /* _FSDispatch,Sys,Immed */
```

Assembler equate and macro for `_GetUGEntry`:

```
selectGetUGEntry    EQU   $44

                    macro
                    _GetUGEntry &async1,&async2
                          DoHFSDispatch selectGetUGEntry,&async1,&async2
                    endm
```

Further Reference:

- *Inside Macintosh*, Volume IV, The File Manager
- *Inside Macintosh*, Volume V, File Manager Extensions In a Shared Environment
- *Inside Macintosh*, Volume VI, The File Manager
- *Inside AppleTalk*, AppleTalk Filing Protocol
- Technical Note #301, File Sharing and Shared Folders