ARA01 : Corrections and additions for the ERS

Written by : ABS Engineering September, 1991
Updated : Benjamin Beasley October, 1992
Updated: Chuq Von Rospach May, 1994

The document "AppleTalk® Remote Access, Application Programming Interface (API) External Reference Specifications" that is distributed with the developers kit has a number of errors and additions necessary to provide developers with a complete example specification. This technical note details the changes necessary to correct the document and its example code.

Minor typos and errors

Page 7

1) Under **LOAD** the word 'to' is missing between "used" and "ensure".

2) The second sentence under **LOAD.** The structure referenced is TRemoteAccessParamHeader, the correct structure is TRemoteAccessParamBlock.

3) In the code example for **LOAD** the TRemoteAccessParamHeader should be changed to TRemoteAccessParamBlock.

4) Under **UNLOAD** the word 'to' is missing between "used" and "release".

5) The second sentence under **UNLOAD.** The struc referenced is TRemoteAccessParamHeader, the correct structure is TRemoteAccessParamBlock.

6) In the code example for **UNLOAD** the TRemoteAccessParamHeader should be changed to TRemoteAccessParamBlock.

7) The correct example code is:

```
#include "RemoteAccessInterface.h"
void UnloadRemoteAccess()
{
    TRemoteAccessParamBlock unloadPB;

    unloadPB.LOAD.csCode = RAM_EXTENDED_CALL;               // extended call
    unloadPB.LOAD.resultStrPtr = nil;                       // result string
    unloadPB.LOAD.extendedType = REMOTEACCESSNAME;          // to remote access
    unloadPB.LOAD.extendedCode = CmdRemoteAccess_Unload;    // try to unload
    PBRemoteAccess(&unloadPB, false);                        // issue sync call
    if (unloadPB.LOAD.ioResult)
           ShowError(unloadPB.LOAD.ioResult);
        } // UnloadRemoteAccess
```

Page 8

1) There should be a space between **guaranteed** & **Access** in the fifth sentence**.** Additionally, **Access** should not be capped.

Page 9

1) In the **DoConnect()** example, TRemoteAccessConnectParam should be changed to TRemoteAccessParamBlock.

2) In the **DoConnect()** example the routine called CopyPStr is not defined. Instead you should use memcpy.

3) The structure pb should be changed to connectPB to make it consistent with the **LOAD** and **UNLOAD** commands. The corrected code would look like this:

```
#include "RemoteAccessInterface.h"
void DoConnect()
{
    TRemoteAccessParamBlock connectPB;
    Str255 PathName = "\pMyhardDisk:Remote Access:Connect Document";

    LoadRemoteAccess();                                      // Get the Remote Access
                                                             // Manager loaded
    connectPB.CONNECT.csCode = RAM_EXTENDED_CALL;            // extended call
    connectPB.CONNECT.resultStrPtr = nil;                    // don't want result strings
    connectPB.CONNECT.extendedType = REMOTEACCESSNAME;       // to remote access
    connectPB.CONNECT.extendedCode = CmdRemoteAccess_DoConnect; // connect command
    connectPB.CONNECT.portGlobalsPtr = nil;                  // use the user port
    connectPB.CONNECT.fileInfo.vRefNum = 0;                  // use the full pathname
    connectPB.CONNECT.fileInfo.parID = 0;                    //
    BlockMoveData (PathName, connectPB.CONNECT.fileInfo.name, PathName[0] + 1);
                                                             // copy the string to the
                                                             // fileInfo.Name

    // Ask for password if needed, use connection document, & show status
    pb.CONNECT.optionFlags = kNSCanInteract | kNSConnectDocument | kNSShowStatus;
    PBRemoteAccess(&pb, false);                              // issue sync call
    if (pb.CONNECT.ioResult)
           ShowError(pb.LOAD.ioResult);               // Do Error reporting and recovery
    UnloadRemoteAccess();
} // DoConnect
```

Page 10

1) The code example for **DoConnect()** shown above should terminate with **} // DoConnect** so that it is consistent with the rest of the code examples.

2) In the **DoDisconnect()** example, TRemoteAccessDisconnectParam should be changed to TRemoteAccessParamBlock.

3) **DoDisconnect()** should be rewritten as follows:

```
#include "RemoteAccessInterface.h"
```

```
void DoDisconnect()
{
    TRemoteAccessParamBlock disconnectPB;

    disconnectPB.DISCONNECT.csCode = RAM_EXTENDED_CALL;       // extended call
    disconnectPB.DISCONNECT.resultStrPtr = nil;               // don't want result strings
    disconnectPB.DISCONNECT.extendedType = REMOTEACCESSNAME;  // to remote access
    disconnectPB.DISCONNECT.extendedCode = CmdRemoteAccess_Disconnect; // disconnect command
    disconnectPB.DISCONNECT.portGlobalsPtr = nil;             // user port
    disconnectPB.DISCONNECT.abortOnlyThisPB = nil;           // don't get tied to any specific pb
    disconnectPB.DISCONNECT.optionFlags = 0 | kNSShowStatus;  // show status while disconnecting
    PBRemoteAccess(&disconnectPB, false);                     // issue sync call
    if (disconnectPB.CONNECT.ioResult)
            ShowError(disconnectPB.LOAD.ioResult);            // Do Error reporting and recovery
} // DoDisconnect
```

Page 11

1) There is no code example for the **IsRemote()** function.

Page 12

1) **GetStatus()** should be rewritten as:

```
#include "RemoteAccessInterface.h"
void GetStatus()
{
    TRemoteAccessParamBlock statusPB;
    Str255 UserName, connectedTo, lastMessage;
    long StatusBits;

    statusPB.STATUS.csCode = RAM_EXTENDED_CALL;                       // extended call
    statusPB.STATUS.resultStrPtr = nil;                               // put results here
    statusPB.STATUS.portGlobalsPtr = nil;                             // do UserPort
    statusPB.STATUS.extendedType = REMOTEACCESSNAME;                  // to Netshare
    statusPB.STATUS.extendedCode = CmdRemoteAccess_Status;            // status command
    statusPB.STATUS.userNamePtr = &UserName;
    statusPB.STATUS.connectedToNamePtr = &connectedTo;
    statusPB.STATUS.theLastStatusMsgPtr = &lastMessage;
    statusPB.STATUS.statusUserNamePtr = nil;
    statusPB.STATUS.statusMsgSeqNum = 0;
    PBRemoteAccess(&statusPB, false);
    if (statusPB.STATUS.ioResult)
            ShowError(statusPB.STATUS.ioResult);       // Do Error reporting and recovery
    else
    {
            // now decode the flag bits into words
            StatusBits = statusPB.STATUS.statusBits;
            if (StatusBits & CctlServerMode)
                    printf("Answer Connection\n");
            if (StatusBits & CctlConnected)
                    printf("Calling connection\n");
            if (StatusBits & CctlConnectionAborting)
                    printf("Cancel in progress\n");
            if (StatusBits & CctlAnswerEnable)
                    printf("Waiting for incoming call\n");
            if (StatusBits & CctlConnectInProg)
                    printf("Connection in progress\n");
    }
} // GetStatus
```

Note: The variable in the original code, **lastSeqNum,** causes a compiler warning because it is not needed in the code fragment **pb** has to be changed to **statusPB** to be consistent with previous code examples, lastly, the code fragment ends with the **} // GetStatus** to remain consistent with the other code fragments.

Page 14

1) **MungePassword()** should be rewritten as:

```
#include "RemoteAccessInterface.h"
void MungePassword(UserName, PassWord)
unsigned char *UserName, *PassWord;
{
    TRemoteAccessParamBlock mungePB;

    mungePB.MUNGEPW.csCode = RAM_EXTENDED_CALL;                      // extended call
    mungePB.MUNGEPW.resultStrPtr = nil;                              // result string
    mungePB.MUNGEPW.extendedType = REMOTEACCESSNAME;                 // to remote access
    mungePB.MUNGEPW.extendedCode = CmdRemoteAccess_PassWordMunger;
    mungePB.MUNGEPW.userNamePtr = (unsigned char *) &UserName;
    mungePB.MUNGEPW.passWordPtr = (unsigned char *) &PassWord;
    PBRemoteAccess(&mungePB, false);                         // issue sync call
                                                             // encrypt the eight bytes
                                                             // pointed to by PassWord
    if (mungePB.MUNGEPW.ioResult)
           ShowError(mungePB.MUNGEPW.ioResult);

} // MungePassword
```

Note: this code is rewritten to be more realistic as the user name and password would probably be passed in to a function like this as opposed to having them hard coded in the function itself. The TRemoteAccessParamBlock variable **MungePB** was changed to **mungePB** to be consistent with the other fragments, and the terminating line ends as **} // MungePassword** to be consistent with other examples.

2) In the **GetCodeHooks** section, the word **are** should be removed from the second sentence, or it should be rewritten.