Handling Incoming Session Requests

Use the **PPCAccept** function or the **PPCReject** function to accept or reject an incoming session request.

Warning: If the **PPCInform** function (with the autoAccept parameter set to FALSE) returns a noErr result code, you must call either the **PPCAccept** function or the **PPCReject** function. The computer trying to initiate a session (using the **StartSecureSession** function or the **PPCStart** function) waits (hangs) until the session attempt is either accepted or rejected, or until an error occurs.

The following program illustrates how you use the **PPCAccept** function to accept a session request. This listing reuses the parameter block used in the **PPCInform** function, so the sessRefNum field already contains the session reference number needed by the **PPCAccept** function.

```
// Accepting a session request using the PPCAccept function

// Assuming inclusion of MacHeaders
#include <PPCToolBox.h>

// Prototype your routine like this prior to calling it
void DoPPCAccept(PPCParamBlockPtr);

void DoPPCAccept(PPCParamBlockPtr pb)

{
    OSErr err;
    // the iocompletion routine is prototyped as follows
    pascal void MyAcceptCompProc(PPCParamBlockPtr);

    // accept the session
    pb->acceptParam.ioCompletion = &MyAcceptCompProc;

    // the sessRefNum field is set by the PPCInform function
    err = PPCAccept(&pb->acceptParam, TRUE);  // asynchronous
}
```

For each function that you use asynchronously, you should provide a completion routine. The following program illustrates a completion routine for a **PPCAccept** function. This procedure gets called at interrupt time when the **PPCAccept** function completes. If there are no errors, it sets the global variable gSessionOpen to TRUE. The global variable gPBInUse is set to FALSE to inform the application that the parameter block and the records it points to are no longer in use.

You can use the session reference number in subsequent **PPCWrite**, **PPCRead**, and **PPCEnd** functions once a session is accepted.

```
// Completion routine for a PPCAccept function
```

```
// Assuming inclusion of MacHeaders
#include <PPCToolBox.h>

// global variables used to relay information to the application
Boolean gSessionOpen;
Boolean gPBInUse;

// Prototype the completion as follows prior to using
// Note, that the routine must use pascal calling conventions
// since it will be called from the ToolBox
pascal void MyAcceptCompProc(PPCParamBlockPtr);

// global

pascal void MyAcceptCompProc(PPCParamBlockPtr pb)

{
    if ( pb->acceptParam.ioResult == noErr )
            // accept completed so the session is completely open
        gSessionOpen = TRUE;
            // use a global to tell the application that PPCParamBlockRec
            // and the records it points to can be deallocated
        gPBInUse = FALSE;
}
```

Use the **PPCReject** function to reject an incoming session request. The following program illustrates how you use the **PPCReject** function to reject a session request.

This listing reuses the parameter block used in the **PPCInform** function, so the sessRefNum field already contains the session reference number needed by the **PPCReject** function.

```
// Rejecting a session request using the PPCReject function

// Assuming inclusion of MacHeaders
#include <PPCToolBox.h>

// Prototype the function like this prior to calling it
void DoPPCReject( PPCParamBlockPtr );

void DoPPCReject( PPCParamBlockPtr pb)

{
    OSErr err;

    // prototype your Reject completion Procedure like this:
    pascal void MyRejectCompProc(PPCParamBlockPtr);

            // reject the session}
    pb->rejectParam.ioCompletion = &MyRejectCompProc;
            // the sessRefNum field is set by the PPCInform function
    pb->rejectParam.rejectInfo = -1;
```

```
    err = PPCReject(&pb->rejectParam, TRUE);    // asynchronous
}
```

The following program illustrates a completion routine for a **PPCReject** function. This procedure gets called at interrupt time when the **PPCReject** function completes. In this example, the global variable gPBInUse is set to FALSE to inform the application that the parameter block and the records it points to are no longer in use.

```
// Completion routine for a PPCReject function

// Assuming inclusion of MacHeaders
#include <PPCToolBox.h>

// global variable to relay information to application
Boolean gPBInUse;

// Prototype the completion routine like this
// Note, that since it is called from the toolbox
// it must use Pascal calling conventions
pascal void MyRejectCompProc(PPCParamBlockPtr);

pascal void MyRejectCompProc(PPCParamBlockPtr pb)

{
    // use a global to tell the application that PPCParamBlockRec and
    // the records it points to can be deallocated
    gPBInUse = FALSE;
}
```