NOTE: This Technical Note has been retired. Please see the Technical Notes page for current documentation.

Technical Note AO03
Constructing an AOCE AppleTalk Address

CONTENTS

This Technical Note describes how to create an AOCE `OCERecipient` (`DSSpec`) data structure containing an AppleTalk address for the PowerTalk mail slot on a given machine.

Updated: [February 1995]

Introduction

AOCE uses a single, multi-purpose data structure, the `OCERecipient` or `DSSpec` structure (see *Inside Macintosh:AOCE Application Interfaces* , chapter 7-24 for a description) to represent addresses in a given address space. This technical note describes how to construct an `OCERecipient` structure representing the PowerTalk mail slot (the slot that allows the transfer of letters) for an address on an AppleTalk internet.

You can construct this kind of address using either direct addressing or indirect addressing techniques, as described in *Inside Macintosh* ::*AOCE Application Interfaces* chapter 7. The direct addressing form of this type of address is called the Direct AppleTalk mail address. The Direct AppleTalk mail address format specifies the address in the extension portion of the `OCERecipient` structure. The record location information (`RLI`) of the `RecordID` is not needed.

The indirect addressing form of this type of address is called the PowerShare mail address. It uses the `RecordID` portion of the `OCERecipient` structure to specify an entity that is to receive a message - and the extension part, if a queue other than the default queue needs to be specified.

Given below is a description of how to construct an address using both formats along with sample code.

Back to top

Building a Direct AppleTalk mail address

As a first step in building a Direct AppleTalk mail address you'll need to obtain the address of the given entity on the AppleTalk network. This can be accomplished using the AppleTalk Name Binding Protocol (NBP) routines (see *Inside Macintosh:Networking* for the details). The NBP name mapping for a given AppleTalk address is called an entity name and consists of 3 fields: object, type and zone. The `EntityName` data structure is defined to hold this NBP name.

AOCE recognizes an AppleTalk address by the value `'alan'` (use the known constant kOCEalanXtn) in the `extensionType` field of the `OCERecipient` structure. In this case, the extension portion of the `OCERecipient` structure contains the address. The `extensionValue` field of the `OCERecipient` structure is defined as follows:

```
Str32     objectName
Str32     typeName
Str32     zoneName
Str32     queueName    (optional)
```

The first three fields (object, type and zone) are all required, and all are packed. These fields are in the exact format used by the NBP `EntityName` structure. If the NBP routines are not used to construct this structure then it is up to the programmer to make sure that each field starts on a even boundary.

As is usual for AppleTalk, you can specify a zero-length string or the wildcard character * to indicate the local zone. To specify an NBP name for the PowerTalk mail slot on the chosen entity you need to set the `typeName` field to the string "\pMsgReceiver", which is defined by the constant `kIPMWSReceiverNBPType` from the AOCE interface files.

The `queueName` field is optional. It is used to specify the name of the queue to which the message is to be delivered. If it is included, make sure that is starts on an even boundary in the structure. If it is left out, then any messages delivered to this address are delivered to the user's default queue.

The local record identifier (`LocalRecordID)` part of the `OCERecipient` structure, which uniquely identifies a record within a catalog, should be setup as follows: a null `cid` field, the `recordName` field is the `objectName` from the NBP name, and the `recordType` field, which indicates the type of entity that the record represents, is defined by the following Record Attribute String definition:

```
type 'rast' // Record Attribute String.

{

integer = 0; // smRoman for the script.

len: integer = (stop - len)/8 - 2;

string; // stand-alone attribute prefix

longInt; // tag

string; // attribute type

stop:

};
```

Here's how the `LocalRecordID` looks:

```
cid
source 0
seq 0
recordName NBP object name here
recordType 'rast' resource here
```

The attribute prefix in the Record Attribute String is defined by the known constant `kAttributeValueRecTypeBody.` The tag is the value `'alan'`. The attribute type is defined by the known constant `kMailSlotsAttrTypeBody.` Here is the Record Attribute String resource that is used in the sample code below:

```
resource 'rast'(128)

{

kAttributeValueRecTypeBody,

'alan',

kMailSlotsAttrTypeBody

};
```

The sample code shown below first obtains the NBP name of the machine that the code is executed on, and then uses that name to construct the `OCERecipient` structure. To use an entity name of your own choosing, simply replace the code below that builds the entity name with your own (see the `BuildOurEntity` procedure). Lastly, the code packs the contents of the `OCERecipient` structure into an `OCEPackedRecipient` structure.

```
    /* prototypes */

OCEPackedRecipient *CreateDirectAtlkMailAddress(void);

OSErr BuildOCERecipient(Ptr entityBuffer,

OCERecipient  *recipient);

OSErr BuildOurEntity(EntityPtr  entityPtr);

OSErr GetOurZoneName(StringPtr zonePtr);

   /* globals */
#define   gThisZone  "\p*"

#define  kRastId    128
#define  kRastType  'rast'

/*****************************************************************

 *  CreateDirectAtlkMailAddress

 *
 *****************************************************************/

OCEPackedRecipient *CreateDirectAtlkMailAddress()
{

OCERecipient   recipient;

short          size;

Ptr            ocePackedRecipientPtr;

RString32      name;
```

```
RString        type;

PackedRLI      packedRLI;

LocalRecordID  localRID;

CreationID     cid;

RecordID       rid;

char           entityBuffer[sizeof(EntityName)];

EntityName     entityName;

OSErr          err;

err = BuildOurEntity(&entityName);

if (err == noErr)

{

name.dataLength = kRString32Size;

type.dataLength = kRStringMaxBytes;

OCENewLocalRecordID((RStringPtr)&name,

&type,

&cid,

&localRID);

OCENewRecordID(&packedRLI,

&localRID,

&rid);

NBPSetEntity(entityBuffer,

&entityName.objStr[0],

&entityName.typeStr[0],

&entityName.zoneStr[0]);

recipient.entitySpecifier = &rid;

err = BuildOCERecipient(&entityBuffer,

&recipient);
if (err == noErr)

{

size = OCESizePackedRecipient(&recipient);

ocePackedRecipientPtr = NewPtr(size);

if (ocePackedRecipientPtr != NULL)

{

OCEPackRecipient(&recipient,

ocePackedRecipientPtr);

return (OCEPackedRecipient *)ocePackedRecipientPtr;

}

}

}

    return nil;

}

/*****************************************************************

 *  BuildOCERecipient

 *
 *****************************************************************/

OSErr BuildOCERecipient(Ptr entityBuffer,

OCERecipient  *recipient)

{

OSErr err;

Handle recTypeHdl;

RecordIDPtr  ridPtr;

Ptr          namePtr;

short        totalSize,nameSize,i;

ridPtr = recipient->entitySpecifier;

ridPtr->rli = NULL;
```

```
OCESetCreationIDtoNull(&ridPtr->local.cid);

/* recordName is entity->objName */

OCEPToRString(entityBuffer,  /* first item in buffer is objectName */

smRoman,

ridPtr->local.recordName,

kRString32Size);

/* get our recordType resource */

recTypeHdl = GetResource(kRastType,kRastId);

if (recTypeHdl != NULL)

{

HLock(recTypeHdl);

err = MemError();

if (err == noErr)

{

OCECopyRString((RStringPtr)*recTypeHdl,

ridPtr->local.recordType,

ridPtr->local.recordType->dataLength);

recipient->extensionType = kOCEalanXtn;

namePtr = entityBuffer;  /* point to first name */

totalSize = 0;

/* determine size of entityName structure */

for (i=0; i<3; ++i)

{

nameSize = (*namePtr + 1);  /* size of name (object,type then zone)

*/

totalSize = totalSize + nameSize;  /* add to total size of entity structure */

namePtr = namePtr + nameSize;  /* point to next name */

}
/* extension is the entity name */

recipient->extensionSize = totalSize;

recipient->extensionValue = entityBuffer;

HUnlock(recTypeHdl);

}

else

return (err);

}
else

{

err = ResError();

if (err == noErr)

return (resNotFound);

else

return (err);

}

return (err);
}

/*****************************************************************
 *  BuildOurEntity
 *
 *  Builds an AppleTalk NBP EntityName structure for our machine
 *****************************************************************/

OSErr BuildOurEntity(EntityPtr  entityPtr)

{
StringHandle  userName;

OSErr err;
short resError;

err = GetOurZoneName(&entityPtr->zoneStr[0]);

if (err == noErr)

{
```

```
BlockMove(kIPMWSReceiverNBPType,

&entityPtr->typeStr[0],

kIPMWSReceiverNBPType[0]+1);

/* get flagship name */

userName = GetString(-16413);

if (userName != NULL)

{

HLock((Handle)userName);

err = MemError();

if (err == noErr)

{

BlockMove(*userName,

&entityPtr->objStr[0],


*userName[0]+1);

HUnlock((Handle)userName);

}

else

return (err);
}

else

{

resError = ResError();

if (resError == noErr)

return (resNotFound);

else

return (resError);

}

}
```

```
return (err);

}


/*****************************************************************
 *  GetOurZoneName
 *
 *****************************************************************/

OSErr GetOurZoneName(StringPtr zonePtr)
{

XPPParamBlock xpb;

OSErr err;

#define PB (xpb.XCALL)

PB.csCode          = xCall;

PB.xppTimeout      = 4;

PB.xppRetry        = 4;

PB.xppSubCode      = zipGetMyZone;

PB.zipBuffPtr      = zonePtr;

PB.zipInfoField[0] = 0;

PB.zipInfoField[1] = 0;

err = GetMyZone(&xpb, false);

if ( (err == noBridgeErr) ||

(err == reqFailed) )

{

/* no router so specify "this" zone */

BlockMove(&gThisZone,

zonePtr,

(1 + 1));

err = noErr;

}

return err;
```

```
}
```

Building a PowerShare mail address

A PowerShare mail address uses the `RecordID` portion of the `OCERecipient` structure to specify an entity that is to receive a message. The `extensionSize` and `extensionValue` fields of the `OCERecipient` structure are set to 0 and NULL respectively, while the `extensionType` is set to the known constant `kOCEentnDSSpec` (`'entn'`). In this case, mail messages will be delivered to the user's preferred mail queue while non-mail messages will be delivered to the preferred message queue. Here's how it looks:

```
extensionType 'entn'
extensionSize 0
extensionValue NULL
```

If a queue other than the preferred queue needs to be specified, then use the queue-name format for attribute values in the `extensionValue` part of the `OCERecipient` structure as described in *Inside Macintosh ::AOCE Application Interfaces* chapter 7-16.

The record location information structure (RLI) will specify the catalog and catalog node in which the record that describes the entity resides. Set the `directoryName` field to contain the `RString` "AppleTalk". This indicates which catalog the record resides in - namely, the AppleTalk catalog. The `discriminator` structure of the `RLI` will specify the type of the catalog in the signature field. Use `'atlk'` here. Set the `misc` field to the known constant `kDirDSAMKind` (`'dsam'`). The `path` field of the `discriminator` record will indicate the name of all of the catalog nodes on the path from the catalog node in which the record resides, to the catalog root node. Set this to the AppleTalk zone name in which the target machine is located. Here's how the `RLI` looks:

```
directoryName  "AppleTalk"

discriminator

signature 'atlk'

misc 'dsam'

path specify the AppleTalk zone name here
```

The local record identifier (`LocalRecordID` structure) will specify the name and type of the record within the catalog. Set the `recordName` field to be the machine name of the target (flagship name or object name from the NBP Entity Name record), while the `recordType` field should be set to the same Record Attribute String defined earlier ('rast' resource). The creation id will be NULL. Here's how the `LocalRecordID` looks:

```
cid

source 0

seq 0

recordName machine name here

recordType 'rast' resource here
```

Below is some sample code that builds a PowerShare mail address for the machine that the code is executed on. To specify a machine of your own choosing, simply change the BuildOurEntity() routine so that it builds an Entity Name structure containing the desired target machine.

```
/* prototypes */

PackedDSSpec* CreatePowerShareMailAddress();

OSErr BuildLocalRID(LocalRecordIDPtr localRID,

RStringPtr recordType,

RStringPtr recordName,

StringPtr machineName);

PackedRLI* BuildPackedRLI(RLIPtr rli,

StringPtr zoneName);

DirectoryName kDirectoryName  = {smRoman,

9,

"AppleTalk"};

#define kDirATLKKind 'atlk'

/*****************************************************************

*  CreatePowerShareMailAddress

*

*****************************************************************/

PackedDSSpec* CreatePowerShareMailAddress()

{

OSErr err;

DSSpec dSSpec;
```

```
RecordID rid;

LocalRecordID localRID;

RLI rli;

RString recordType;

RString recordName;

EntityName entityName;

Ptr packedRLI;

Ptr packedDSSpec;

unsigned short size;

packedDSSpec = NULL;

packedRLI = NULL;

/* get AppleTalk address of target machine */

err = BuildOurEntity(&entityName);

if (err == noErr)

{

/* construct the local record identifier */
err = BuildLocalRID(&localRID,

&recordType,

&recordName,
&entityName.objStr);

if (err == noErr)

{

/* construct the record location information structure */

packedRLI = (Ptr)BuildPackedRLI(&rli,

&entityName.zoneStr);
if (packedRLI != NULL)

{

/* create a RecordID structure */

OCENewRecordID((PackedRLI*)packedRLI,
&localRID,

&rid);

/* fill in all fields of the OCERecipient structure

*/

dSSpec.entitySpecifier = &rid;

dSSpec.extensionType = kOCEentnDSSpec;

dSSpec.extensionSize = 0;
dSSpec.extensionValue = nil;

size = OCEPackedDSSpecSize(&dSSpec);

packedDSSpec = NewPtr(size);

if (packedDSSpec != NULL)

{

/* build a packed DSSpec */

err = OCEPackDSSpec(&dSSpec,

(PackedDSSpec *)packedDSSpec,

size);

}

}

}

}

if (packedRLI != NULL)

{

DisposePtr((Ptr)packedRLI);

}

return ((PackedDSSpec *)packedDSSpec);
}

/*****************************************************************

*  BuildLocalRID
```

```
*

***************************************************************/

OSErr BuildLocalRID(LocalRecordIDPtr localRID,

RStringPtr recordType,

RStringPtr recordName,

StringPtr machineName)

{

OSErr err;

Handle recTypeHdl;

/* get our recordType resource */

recTypeHdl = GetResource(kRastType,kRastId);

if (recTypeHdl != NULL)

{

HLock(recTypeHdl);

err = MemError();

if (err == noErr)

{

OCECopyRString((RStringPtr)*recTypeHdl,

recordType,

kRStringMaxBytes);

OCEPToRString((ConstStr255Param)machineName,

smRoman,

recordName,

kRStringMaxBytes);

OCENewLocalRecordID(recordName,

recordType,

OCENullCID(),

localRID);
HUnlock(recTypeHdl);

}

else

return (err);

}

else

{

err = ResError();

if (err == noErr)

return (resNotFound);

else

return (err);

}

return noErr;

}

/****************************************************************

* BuildPackedRLI

*

****************************************************************/

PackedRLI* BuildPackedRLI(RLIPtr rli,

StringPtr zoneName)

{

DirDiscriminator dirDiscrim;

OSErr err;

RString zone;

unsigned short size;

Ptr packedPath;
Ptr packedRLI;

RStringPtr path[1];

packedRLI = NULL;

dirDiscrim.signature = kDirATLKKind;

dirDiscrim.misc = kDirDSAMKind;

OCEPToRString((ConstStr255Param)zoneName,

smRoman,

&zone,

kRStringMaxBytes);

path[0] = (RStringPtr)&zone;

/* set path to target AppleTalk zone name */

size = OCEPackedPathNameSize(path, 1);

packedPath = NewPtr(size);

if (packedPath != NULL)

{

err = OCEPackPathName(path,

1,

(PackedPathName *)packedPath,

size);

if (err == noErr)

{

OCENewRLI(rli,

&kDirectoryName,

&dirDiscrim,

0,

(PackedPathName*)packedPath);

size = OCEPackedRLISize(rli);

packedRLI = NewPtr(size);
if (packedRLI != NULL)

{

err = OCEPackRLI(rli, (PackedRLI*)packedRLI, size);

}

}

}

if (packedPath != NULL)

{

DisposePtr((Ptr)packedPath);

}

return (PackedRLI *)packedRLI;

}
```

Back to top

References

Inside Macintosh :AOCE Application Interfaces

Inside Macintosh :Networking

Back to top

Downloadables

	Acrobat version of this Note ()	Download

Back to top