This chapter describes the Zone Information Protocol (ZIP) that maintains mappings of zone names to network numbers on internet routers. ZIP is primarily implemented by routers. A small portion of ZIP is implemented on nodes that are not routers to allow you to obtain zone information from a router node. This chapter describes only the portion of ZIP that is implemented on a node that is not a router.

You should read this chapter if you want to obtain

- the name of the zone to which the node belongs that is running your application
- the names of the zones for the local network to which your application's node is connected
- the names of all the zones that exist throughout the AppleTalk internet to which your local network belongs

The portion of ZIP that is implemented on nodes that are not routers uses the AppleTalk Transaction Protocol (ATP) to send requests for zone information to a router node. To better understand how ZIP handles your requests for information and returns to you responses to those requests, you should read the chapter "AppleTalk Transaction Protocol (ATP)" in this book.

For an overview of the Zone Information Protocol and how it fits within the AppleTalk protocol stack, read the chapter "Introduction to AppleTalk" in this book, which also introduces and defines some of the terminology used in this chapter. For a description of the Zone Information Protocol specification, see *Inside AppleTalk,* second edition.

About ZIP

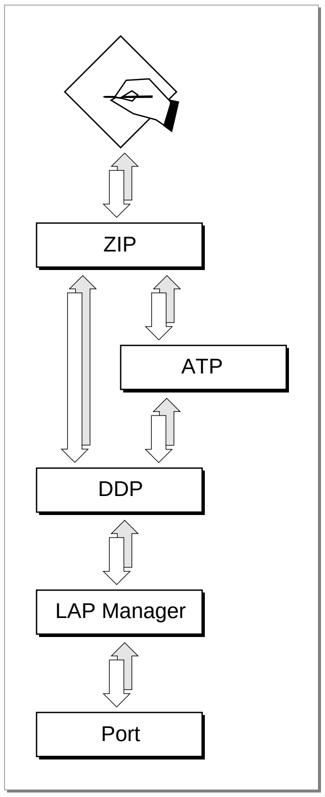
The Zone Information Protocol (ZIP) provides applications and processes with access to zone names. A ***zone*** is a logical grouping of nodes in an AppleTalk internet, and each zone is identified by a name. A zone name is typically used to identify an affiliation between a group of nodes, such as a group of nodes belonging to a particular department within an organization.

ZIP maintains the mapping of networks and the zones they include for all networks belonging to an AppleTalk internet:

- Every node on a network belongs to a zone; a node can belong to only one zone at a time.
- A nonextended network contains only one zone, and all nodes in that network belong to the same zone.
- A single extended network can contain nodes that belong to up to 255 different zones. A single zone can include nodes that belong to different extended networks. Each AppleTalk extended network has associated with it a list of the zones to which its nodes can belong. A node joining the network can select its zone from this list.

On each router node in the internet, ZIP builds a zone information table that includes each network's number (extended networks have network number ranges) in association with the network's list of zones. Nodes that are not routers, such as end-user systems, do not contain a zone information table. However, a portion of ZIP is implemented on each nonrouter node so that applications and processes can gain access to their own node's zone name, names of all the zones on their local network, or names of all the zones throughout the internet. The .XPP driver implements the part of ZIP that is on nonrouter nodes, and it provides an interface that allows an application or process to request zone name information in a transaction-based dialog. ZIP uses the transaction-based services of ATP to transport requests from workstation nodes to router nodes. Figure 4-1 shows ZIP and its underlying protocols. The portion of ZIP that is implemented on nonrouter nodes, such as workstations, uses the services of ATP.

Figure 4-1 The Zone Information Protocol (ZIP) and the underlying AppleTalk protocols



Using ZIP

The Zone Information Protocol provides three functions that you can use to obtain the names of registered zones. You can use these functions to obtain

- the name of the zone to which your application and its node belong
- the names of the zones in your local network or the names of all the zones that exist throughout the AppleTalk internet to which your local network belongs

Applications running on nodes connected to both extended and nonextended networks can use ZIP to get the name of their node's zone. An application running on a node that belongs to an extended network can call ZIP to get a list of all the zone names associated with that network. For example, a network administration application might use ZIP to provide an administrator with a list of the zones for a particular network so that the administrator can select the correct zone for a node when adding nodes to a network.

You can use ZIP in conjunction with NBP. For example, you can use ZIP to look up zones on the network, then use NBP to look up names in each zone.

ZIP sends the `GetMyZone`, `GetLocalZones`, and `GetZoneList` functions as AppleTalk Transaction Protocol (ATP) requests. These requests always ask for a single response. For example, when you call ZIP to request zone name information, the portion of ZIP implemented on the node running your application sends a request using the transaction-based services of ATP to the portion of ZIP implemented on a local router that contains the zone information table; using ATP, ZIP on the router node transmits a response to your request.

When you call `GetMyZone` to get the name of your node's zone, ZIP returns the complete zone name in a single ATP response and writes that zone name to the buffer you provide. However, when you want to retrieve a list of zone names belonging either to your local network or to all of the networks forming the internet, ZIP may not always be able to return the complete list of names in a single ATP response. In this case, you need to call the ZIP function repeatedly in a loop in order to retrieve all of the zone names.

The `GetMyZone`, `GetLocalZones`, and `GetZoneList` functions each use a parameter block of type `XPPParamBlock` to contain input and output values for the call. You use the `xCallParam` variant record to the XPP parameter block for the ZIP functions. This parameter block contains an `ioRefNum` field, which the MPW interface sets to the .XPP driver reference number.

The parameter block for each of the three ZIP functions includes a `csCode` field and an `xppSubCode` field. You do not need to set these field values before you call the function; the MPW interface fills in the value for each of these fields. The value for the `csCode` field is always `xCall`. The `xppSubCode` field value identifies the specific ZIP function, and it differs for each of the three functions.

For the three ZIP functions, you specify timeout and retry values that determine the behavior of the ATP transaction that the ZIP call relies on. You need to set values for these fields before you call the ZIP function. You use the parameter block's `xppTimeout` field to set the timeout value and the `xppRetry` field to set the retry value. The timeout tells ATP how long in seconds to wait between each attempt, and the retry value tells it how may retries it should attempt. For information on how ATP uses these values, see the chapter "AppleTalk Transaction Protocol (ATP)" in this book.

For each function, you supply a buffer to hold the returned zone name data and a buffer that ZIP requires for its own use. These two buffers and the `XPPParamBlock` parameter block that you allocate for the function belong to ZIP for the life of the call; you must not manipulate them or alter their contents during the operation. The memory for these buffers and the parameter block belongs to the function until the function completes execution.

If you set the function's `async` Boolean parameter to `TRUE`, either you must provide a completion routine or your application must poll the parameter block's `ioResult` field to determine when the function completes the operation. See the chapter "Introduction to AppleTalk" in this book for a discussion of synchronous and asynchronous execution as it applies to the Boolean parameter.

Getting the Name of Your Application's Zone

Your application can get the name of the zone for the node on which it is running by calling the `GetMyZone` function. The zone name is a data structure of type `Str32`, and the `GetMyZone` function writes the zone name to a buffer that you supply. You set the parameter block's `zipBuffPtr` field to a pointer for a buffer that must be at least 33 bytes in size.

You also supply a buffer that is 70 bytes in size as the value of the `zipInfoField`. You must set the first word of this buffer to 0 before you call the function. This buffer is for ZIP to use.

Listing 4-1 shows the application-defined `DoGetMyZone` function, which illustrates the use of the `GetMyZone` function. The `DoGetMyZone` function declares the parameter block and the return buffer. Then it assigns values to the some of parameter block fields and initializes to 0 the first word of the `zipInfoField` parameter before it calls `GetMyZone`. The MPW interface fills in the XPP parameter block `ioRefNum`, `csCode`, and `xppSubCode` fields, so the `DoGetMyZone` function doesn't need to assign these values.

Listing 4-1 Using the `GetMyZone` function

```
FUNCTION DoGetMyZone(VAR myZoneName: Str32): OSErr;
VAR
   xppPB: XPPParamBlock;
   myZoneName:    ARRAY[1..33] OF Char;
BEGIN
   WITH xppPB DO
      BEGIN
         xppTimeout := 3;           {timeout interval}
         xppRetry := 4;             {retry count for ZIP requests}
         zipBuffPtr := @myZoneName; {buffer for returned zone name}
         zipInfoField[1] := 0;      {initialize first word to 0}
         zipInfoField[2] := 0;
      END;
   DoGetMyZone := GetMyZone(@xppPB, FALSE);
END;
```

If there is no router present in the network, the function returns a function result of noBridgeErr. If the retry count is exceeded before the ATP transaction that ZIP relies on receives a valid response, the function returns a function result of reqFailed. The function returns a function result of tooManyReq when too many concurrent ATP requests have been made. If you receive a function result of tooManyReq, wait a minute or so, and then try again; some transactions can take up to 30 seconds to complete. For the complete list of function results, see the description of the function GetMyZone beginning on page 4-12.

Getting a List of Zone Names for Your Local Network or Its Internet

If your application is running on a node that belongs to an extended network, the application can use the GetLocalZones function to obtain a list of the names of the zones in its node's local network. An application running on a node that belongs to an extended network can also use the GetZoneList function to obtain a list of the names of the zones throughout the AppleTalk internet to which its node's local network belongs. These functions behave similarly.

ZIP returns a single ATP response per request. Because the complete list of zone names may not fit in a single ATP response, you need to make repeated calls to either GetLocalZones or GetZoneList until you receive all of the zone names. You must allocate a buffer to hold the zone names data that the ZIP function returns and point to that buffer from the function's zipBuffPtr parameter block field. This buffer must be 578 bytes in size, large enough to hold an entire ATP response. ZIP returns the zone names into this buffer as a packed array of packed Pascal strings.

The zipNumZones field returns the actual number of zone names that ZIP placed in the buffer. You must set the zipLastFlag field to 0 before you execute the GetZoneList or GetLocalZones function. If the zipLastFlag parameter is still 0 when the command has completed execution, then ZIP is waiting to return more zone names. In this case you must empty the buffer, or allocate a new one, and call the GetZoneList or GetLocalZones function again immediately. When there are no more zone names to return, ZIP sets the zipLastFlag field to a nonzero value. The zipInfoField field is a 70-byte data buffer that you must allocate for use by ZIP. The first time you call any of these functions, you must set the first word of this field to 0. You must not change any values in this field subsequently.

Listing 4-2 shows the application-defined DoGetZoneList function, which illustrates how to use the GetZoneList function. The GetLocalZones function operates in exactly the same fashion.

This DoGetZoneList function allocates a buffer for zone names and repeatedly calls the GetZoneList function to get a list of zone names. If GetZoneList returns a function result of noErr, then the DoGetZoneList code calls the application-defined MyZIPExtract function, shown in Listing 4-3, to remove a zone name from the GetZoneList buffer and place it in the application's buffer. The DoGetZoneList code in Listing 4-2 does not show the application-defined MyAddToZoneList that writes the zone name to the application's buffer.

Listing 4-2 Using `GetZoneList` to retrieve names of zones throughout the AppleTalk internet

```
FUNCTION DoGetZoneList: OSErr;
CONST
   kZoneBufferSize = 578;              {required size of zone list buffer}
VAR
   xppPB: XPPParamBlock;
   result: OSErr;
   zoneBuffer: Ptr;
   index: Integer;
   zoneName: Str32;
BEGIN
      {Allocate buffer for returned zone names.}
   zoneBuffer := NewPtr(kZoneBufferSize);
   IF zoneBuffer = NIL THEN
      result := MemError
   ELSE
   BEGIN
      WITH xppPB DO
      BEGIN
         xppTimeout := 3;              {timeout interval}
         xppRetry := 4;                {retry count}
         zipBuffPtr := zoneBuffer;     {zone names returned here}
         zipLastFlag := 0;             {set to 0 first time through}
         zipInfoField[1] := 0;         {first word of zipInfoField must be }
         zipInfoField[2] := 0;         { initialized to 0 the first time}
      END;

      {Loop to get all of the zone names.}
      REPEAT
         result := GetZoneList(@xppPB, FALSE);
         IF (result = noErr) THEN
            FOR index := 1 TO xppPB.zipNumZones DO
               IF MyZIPExtract(zoneBuffer, xppPB.zipNumZones, index,
                           zoneName) = noErr THEN
                  MyAddToZoneList(zoneName);
      UNTIL (xppPB.zipLastFlag <> 0) OR (result <> noErr);
      DisposPtr(zoneBuffer);           {release memory}
   END;
   DoGetZoneList := result;
END;
```

When you call the `GetZoneList` function or the `GetLocalZones` function to obtain a list of zone names, ZIP returns the zone names as a packed array of packed Pascal strings. Your application must include a routine to extract the zone names that you want from the buffer.

Listing 4-3 shows an application-defined function called `MyZipExtract` that extracts a particular zone name from the buffer of packed zone names returned by either `GetZoneList` or `GetLocalZones`.

The `MyZipExtract` function takes a `numInBuf` input parameter that specifies the number of zone names in the buffer pointed to by the `theBuffer` parameter. For the `numInBuf` parameter, you specify the value that ZIP returned in the `zipNumZones` field of the XPP parameter block used for the `GetZoneList` or `GetLocalZones` function.

You use the `whichOne` input parameter to identify the zone name to extract. The `MyZIPExtract` function returns the zone name in the `zoneName` string parameter.

The `MyZIPExtract` function returns a result of `paramErr` if `whichOne` is 0 or `whichOne` is greater than the number of zones in the buffer. Otherwise, the function returns a function result of `noErr`.

Listing 4-3 Extracting a zone name from the list of zone names returned in the buffer

```
FUNCTION MyZIPExtract (theBuffer: Ptr; numInBuf: Integer; whichOne: Integer;
                     VAR zoneName: Str32): OSErr;
VAR
   result: OSErr;
   zonePtr: Ptr;
BEGIN
   {preflight the input parameters}
   IF (whichOne = 0) OR (whichOne > numInBuf) THEN
      result := paramErr
   ELSE
   BEGIN
      zonePtr := theBuffer;
      {Look for whichOne}
      REPEAT
         whichOne := whichOne - 1;
         IF whichOne <> 0 THEN
               {move pointer to next zone name}
            zonePtr := Ptr(ORD4(zonePtr) +
               Length(StringPtr(zonePtr)^) + 1);
      UNTIL whichOne = 0;
               {return the zone name}
      BlockMove(zonePtr, @zoneName,
               Length(StringPtr(zonePtr)^) + 1);
      result := noErr;
   END;
   MyZIPExtract := result;
END;
```

ZIP Reference

This section describes the data structure and the functions that are specific to the Zone Information Protocol (ZIP). The "Data Structures" section shows the Pascal data structure for the XPP parameter block. The "Routines" section describes the ZIP functions.

Data Structures

This section describes the XPP parameter block that you use to provide information to and receive it from ZIP.

The XPP Parameter Block for ZIP

The Zone Information Protocol's `GetMyZone`, `GetLocalZones`, and `GetZoneList` functions implemented by the .XPP driver use the `xCallParam` variant record to the XPP parameter block defined by the `XPPParamBlock` data type. Your application uses this parameter block to specify input values to and receive output values from a ZIP function. This section defines the parameter block fields that are common to all of the ZIP functions and that are filled in by the MPW interface or returned by the function; your application does not need to fill in these fields. This section does not define reserved fields, which are used either internally by the .XPP driver or not at all. The fields for the `xCallParam` variant record are defined in the function descriptions.

```
TYPE XPPParamBlock =
      PACKED RECORD
         qLink:              QElemPtr;   {reserved}
         qType:              Integer;    {reserved}
         ioTrap:             Integer;    {reserved}
         ioCmdAddr:          Ptr;        {reserved}
         ioCompletion:       ProcPtr;    {completion routine}
         ioResult:           OSErr;      {result code}
         cmdResult:          LongInt;    {reserved}
         ioVRefNum:          Integer;    {reserved}
         ioRefNum:           Integer;    {driver reference number}
         csCode:             Integer;    {primary command code}
         CASE XPPPrmBlkType OF
            xCallParam
               xppSubCode:   Integer;    {secondary command code}
               xppTimeout:   Byte;       {.XPP timeout period}
               xppRetry:     Byte;       {retry count}
               filler1:      Integer;    {reserved}
```

```
        zipBuffPtr:    Ptr;        {returned zone names}
        zipNumZones:   Integer;    {number of zones returned}
        zipLastFlag:   Byte;       {nonzero when all zone names }
                                   { have been returned}
        filler2:       Byte;       {reserved}
        zipInfoField:  PACKED ARRAY[1..70] OF Byte;
                                   {reserved}
END;
XPPParmBlkPtr = ^XPPParamBlock;
```

Field descriptions

`ioCompletion`	A pointer to a completion routine that you can provide. When you execute a function asynchronously, the .XPP driver calls your completion routine when it completes execution of the function if you specify a pointer to the routine as the value of this field. Specify `NIL` for this field if you do not wish to provide a completion routine. If you execute a function synchronously, the .XPP driver ignores the `ioCompletion` field. For information about completion routines, see the chapter "Introduction to AppleTalk" in this book.
`ioResult`	The result of the function. When you execute the function asynchronously, the function sets this field to 1 and returns a function result of `noErr` as soon as the function begins execution. When the function completes execution, it sets the `ioResult` field to the actual result code.
`ioRefNum`	The .XPP driver reference number. The MPW interface fills in this field.
`csCode`	The command code of the XPP command to be executed. The MPW interface fills in this field.

Routines

This section describes the ZIP functions. The ZIP functions allow you to

- obtain the name of the zone to which the node belongs that is running your application
- obtain a list of all the zones for the local network of the node that is running your application
- obtain a list of all the zones associated with the internet that the node running your application belongs to

An arrow preceding a parameter indicates whether the parameter is an input parameter, an output parameter, or both:

Arrow	Meaning
→	Input
←	Output
↔	Both

Obtaining Zone Information

This section describes the Zone Information Protocol (ZIP) functions: `GetMyZone`, `GetLocalZones`, and `GetZoneList`. The `GetMyZone` function returns the name of the zone that your application's node belongs to. The `GetLocalZones` function returns a list of zone names on the local network that your application's node belongs to. The `GetZoneList` function returns a complete list of zones on the internet that your application's node belongs to.

Assembly-language note

The .XPP driver functions all use the same value (`xCall`, which is equal to 246) for the `csCode` parameter to the XPP parameter block. The `xCall` routine uses the value of the `xppSubCode` parameter to distinguish between the functions, as follows:

Function	xppSubCode	Value
GetMyZone	zipGetMyZone	7
GetLocalZones	zipGetLocalZones	5
GetZoneList	zipGetZoneList	6 ◆

GetMyZone

The `GetMyZone` function returns the zone name of the node on which your application is running.

```
FUNCTION GetMyZone (thePBptr: XPPParmBlkPtr;
                    async: Boolean): OSErr;
```

`thePBptr`	A pointer to an XPP parameter block.
`async`	A Boolean that indicates whether the function should be executed asynchronously or synchronously. Specify `TRUE` for asynchronous execution.

Parameter block

→	ioCompletion	ProcPtr	A pointer to a completion routine.
←	ioResult	OSErr	The function result.
→	csCode	Integer	Always `xCall` for this function.
→	xppSubCode	Integer	Always `zipGetMyZone` for this function.
→	xppTimeout	Byte	The retry interval in seconds.
→	xppRetry	Byte	The retry count.
→	zipBuffPtr	Ptr	A pointer to data buffer.
→	zipInfoField	PACKED ARRAY	A data buffer for use by ZIP; first word set to 0.

Field descriptions

`xppSubCode`	A routine selector. This field is automatically set by the MPW interface to `zipGetMyZone` for this function.
`xppTimeout`	The amount of time, in seconds, that the .ATP driver should wait between attempts to obtain the data. A value of 3 or 4 seconds for the `xppTimeout` field is usually sufficient.
`xppRetry`	The number of times the .ATP driver should attempt to obtain the data before returning the request failed (`reqFailed`) result code. A value of 3 or 4 is usually sufficient.
`zipBuffPtr`	A pointer to a 33-byte data buffer that you must allocate. ZIP returns the zone name into this buffer as a Pascal string.
`zipInfoField`	A 70-byte data buffer that you must allocate and initialize for use by ZIP. You must set the first word of this buffer to 0 before you call the `GetMyZone` function.

DESCRIPTION

Before you call `GetMyZone`, you must allocate a buffer that is 33 bytes in size and set the `zipBuffPtr` parameter block field to point to this buffer. ZIP writes the zone name that it retrieves to this buffer that you supply. You must also supply a buffer that is 70 bytes in size as the value of the `zipInfoField` field. This buffer is for ZIP to use. An application running on a node on either an extended or a nonextended network can use this function to retrieve the node's zone name.

SPECIAL CONSIDERATIONS

The memory that you allocate for the parameter block and the two buffers required by the `GetMyZone` function belongs to the .XPP driver until the function completes execution. You can reuse the memory or dispose of it after the operation completes.

ASSEMBLY-LANGUAGE INFORMATION

To execute the `GetMyZone` function from assembly language, call the `_Control` trap macro with a value of `xCall` in the `csCode` field of the parameter block and a value of `zipGetMyZone` in the `xppSubCode` field of the parameter block. To execute this function from assembly language, you must also specify the .XPP driver reference number.

RESULT CODES

`noErr`	0	No error
`noBridgeErr`	–93	No router is available
`reqFailed`	–1096	Request to contact router failed; retry count exceeded
`tooManyReqs`	–1097	Too many concurrent requests
`noDataArea`	–1104	Too many outstanding ATP calls

SEE ALSO

For the `XPPParamBlock` data type, see "The XPP Parameter Block for ZIP" beginning on page 4-10.

To get the correct reference number for the .XPP driver, you can use the Device Manager's `OpenDriver` function, which returns the driver reference number. For information about the `OpenDriver` function, see the chapter "Device Manager" in *Inside Macintosh: Devices*.

GetLocalZones

The `GetLocalZones` function returns a list of all the zone names on the local network—that is, the network that includes the node on which your application is running.

```
FUNCTION GetLocalZones (thePBptr: XPPParmBlkPtr;
                        async: Boolean): OSErr;
```

`thePBptr`	A pointer to an XPP parameter block.
`async`	A Boolean that indicates whether the function should be executed asynchronously or synchronously. Specify `TRUE` for asynchronous execution.

Parameter block

→	`ioCompletion`	`ProcPtr`	A pointer to a completion routine.
←	`ioResult`	`OSErr`	The function result code.
→	`csCode`	`Integer`	Always `xCall` for this function.
→	`xppSubCode`	`Integer`	Always `zipGetLocalZones`.
→	`xppTimeout`	`Byte`	The retry interval in seconds.
→	`xppRetry`	`Byte`	The retry count.
→	`zipBuffPtr`	`Ptr`	A pointer to data buffer.
←	`zipNumZones`	`Integer`	The number of names returned.
←	`zipLastFlag`	`Byte`	A flag that is nonzero if there are no more names.
→	`zipInfoField`	`PACKED ARRAY`	A data buffer for use by ZIP; first word set to 0.

Field descriptions

`xppSubCode`	A routine selector. This field is automatically set by the MPW interface to `zipGetLocalZones` for this function.
`xppTimeout`	The amount of time, in seconds, that the .ATP driver should wait between attempts to obtain the data. A value of 3 or 4 seconds for the `xppTimeout` field is usually sufficient.
`xppRetry`	The number of times the .ATP driver should attempt to obtain the data before returning the request failed (`reqFailed`) result code. A value of 3 or 4 is usually sufficient.

`zipBuffPtr`	A pointer to a 578-byte data buffer that you must allocate. ZIP returns the zone names into this buffer as a packed array of Pascal strings.
`zipNumZones`	The number of zone names that ZIP placed in the data buffer.
`zipLastFlag`	A value that indicates if there are more zone names for your network beyond those that ZIP returned in the `zipBuffPtr` field. The .XPP driver sets this field to 1 if there are no more zone names for your network.
`zipInfoField`	A 70-byte data buffer that you must allocate for use by ZIP. You must set the first word of this buffer to 0 before you call the `GetLocalZones` function the first time through the loop, and you must not change the contents of this field thereafter.

DESCRIPTION

A single extended network can have more than one zone associated with it. Your application can use the `GetLocalZones` function to retrieve the list of zones for its node's local network. The `GetLocalZones` function uses ATP to retrieve the zone information. The buffer that you allocate to hold the returned zone names is the size of a single ATP response. You must call the `GetLocalZones` function repeatedly until all of the zones for the local network have been returned.

Your application must check the `zipLastFlag` field to determine if there are more zone names for your network. If the value of this field is 1, there are no more zone names for your local network. If the value of this field is still 0 when the `GetLocalZones` function completes execution, you must empty the data buffer pointed to by the `zipBuffPtr` parameter and immediately call the `GetLocalZones` function again without changing the value in the `zipInfoField` parameter.

If you receive a `GetLocalZone`s function result of `tooManyReqs`, wait a minute or so, and then try again; some transactions can take up to 30 seconds to complete.

This function works for extended networks only. If the node that is running your application is on a nonextended network and you want the name of that node's zone, use the `GetMyZone` function.

SPECIAL CONSIDERATIONS

The memory that you allocate for the parameter block and the two buffers required by the `GetLocalZones` function belongs to the .XPP driver until the function completes execution. You can reuse the memory or dispose of it after the operation completes.

ASSEMBLY-LANGUAGE INFORMATION

To execute the `GetLocalZones` function from assembly language, call the `_Control` trap macro with a value of `xCall` in the `csCode` field of the parameter block and a value of `zipGetLocalZone`s in the `xppSubCode` field of the parameter block. To execute this function from assembly language, you must also specify the .XPP driver reference number.

RESULT CODES

`noErr`	0	No error
`noBridgeErr`	–93	No router is available
`reqFailed`	–1096	Request to contact router failed; retry count exceeded
`tooManyReqs`	–1097	Too many concurrent requests
`noDataArea`	–1104	Too many outstanding ATP calls

SEE ALSO

For the `XPPParamBlock` data type, see "The XPP Parameter Block for ZIP" beginning on page 4-10.

To get the correct reference number for the .XPP driver, you can use the Device Manager's `OpenDriver` function, which returns the driver reference number. For information about the `OpenDriver` function, see the chapter "Device Manager" in *Inside Macintosh: Devices*.

GetZoneList

The `GetZoneList` function returns a complete list of all the zone names on the internet.

```
FUNCTION GetZoneList (thePBptr: XPPParmBlkPtr;
                      async: Boolean): OSErr;
```

`thePBptr`	A pointer to an XPP parameter block.
`async`	A Boolean that indicates whether the function should be executed asynchronously or synchronously. Specify `TRUE` for asynchronous execution.

Parameter block

→	`ioCompletion`	`ProcPtr`	A pointer to a completion routine.
←	`ioResult`	`OSErr`	The function result.
→	`csCode`	`Integer`	Always `xCall` for this function.
→	`xppSubCode`	`Integer`	Always `zipGetZoneList` for this function.
→	`xppTimeout`	`Byte`	The retry interval in seconds.
→	`xppRetry`	`Byte`	The retry count.
→	`zipBuffPtr`	`Ptr`	A pointer to data buffer.
←	`zipNumZones`	`Integer`	The number of names returned.
←	`zipLastFlag`	`Byte`	A flag that is nonzero if there are no more names.
→	`zipInfoField`	`PACKED ARRAY`	A data buffer for use by ZIP; first word set to 0.

Field descriptions

`xppSubCode`	A routine selector. This field is automatically set by the MPW interface to `zipGetZoneList` for this function.
`xppTimeout`	The amount of time, in seconds, that the .ATP driver should wait between attempts to obtain the data. A value of 3 or 4 seconds for the `xppTimeout` field generally gives good results.
`xppRetry`	The number of times the .ATP driver should attempt to obtain the data before returning the request failed (`reqFailed`) result code. A value of 3 or 4 is usually sufficient.
`zipBuffPtr`	A pointer to a 578-byte data buffer that you must allocate. ZIP returns the zone names into this buffer as a packed array of Pascal strings.
`zipNumZones`	The number of zone names that ZIP placed in the data buffer.
`zipLastFlag`	A value that indicates if there are more zone names for your network beyond those that ZIP returned in the `zipBuffPtr` field. The .XPP driver sets this field to 1 if there are no more zone names for your network.
`zipInfoField`	A 70-byte data buffer that you must allocate for use by ZIP. Typically, you call `GetZoneList` repeatedly from within a loop. You must set the first word of this buffer to 0 before you call the `GetZoneList` function the first time through the loop, and you must not change the contents of this field thereafter.

DESCRIPTION

The `GetZoneList` function returns a complete list of all the zone names on the internet to which the local network of the node running your application belongs. The `GetZoneList` function uses ATP to retrieve the zone information. The buffer that you allocate to hold the returned zone names is the size of a single ATP response. You must call the `GetZoneList` function repeatedly until all of the zones for the local network have been returned.

Your application must check the `zipLastFlag` field to determine if there are more zone names for your network. If the value of this field is 1, there are no more zone names for your local network. If the value of this field is still 0 when the `GetZoneList` function completes execution, you must empty the data buffer pointed to by the `zipBuffPtr` parameter and immediately call the `GetZoneList` function again without changing the value in the `zipInfoField` parameter.

If you receive a `GetZoneList` function result of `tooManyReqs`, wait a minute or so, and then try again; some transactions can take up to 30 seconds to complete.

To obtain a list of only the zone names on the local network, use the `GetLocalZones` function instead. If you use the `GetZoneList` function on a nonextended network, the function returns the `reqFailed` result code.

SPECIAL CONSIDERATIONS

The memory that you allocate for the parameter block and the two buffers required by the `GetZoneList` function belongs to the .XPP driver until the function completes execution. You can reuse the memory or dispose of it after the operation completes.

ASSEMBLY-LANGUAGE INFORMATION

To execute the `GetZoneList` function from assembly language, call the `_Control` trap macro with a value of `xCall` in the `csCode` field of the parameter block and a value of `zipGetZoneList` in the `xppSubCode` field of the parameter block. To execute this function from assembly language, you must also specify the .XPP driver reference number.

RESULT CODES

`noErr`	0	No error
`noBridgeErr`	–93	No router is available
`reqFailed`	–1096	Request to contact router failed; retry count exceeded
`tooManyReqs`	–1097	Too many concurrent requests
`noDataArea`	–1104	Too many outstanding ATP calls

SEE ALSO

For the `XPPParamBlock` data type, see "The XPP Parameter Block for ZIP" beginning on page 4-10.

To get the correct reference number for the .XPP driver, you can use the Device Manager's `OpenDriver` function, which returns the driver reference number. For information about the `OpenDriver` function, see the chapter "Device Manager" in *Inside Macintosh: Devices*.

Summary of ZIP

Pascal Summary

Constants

```
CONST
   {csCode for .XPP extended calls}
   xCall             =     246;

   {.XPP driver unit and reference number}
   xppUnitNum        =     40;
   xppRefNum         =     -41;

   {routine selectors}
   zipGetLocalZones  =      5;      {routine selector for local zone names}
   zipGetZoneList    =      6;      {routine selector for internet zone list}
   zipGetMyZone      =      7;      {routine selector for node's zone name}
```

Data Types

The XPP Parameter Block for ZIP

```
TYPE XPPParamBlock =
      PACKED RECORD
         qLink:               QElemPtr;   {reserved}
         qType:               Integer;    {reserved}
         ioTrap:              Integer;    {reserved}
         ioCmdAddr:           Ptr;        {reserved}
         ioCompletion:        ProcPtr;    {completion routine}
         ioResult:            OSErr;      {result code}
         cmdResult:           LongInt;    {reserved}
         ioVRefNum:           Integer;    {reserved}
         ioRefNum:            Integer;    {driver reference number}
         csCode:              Integer;    {primary command code}
      CASE XPPPrmBlkType OF
         xCallParam
            xppSubCode:    Integer;    {secondary command code}
            xppTimeout:    Byte;       {timeout period for .XPP}
            xppRetry:      Byte;       {retry count}
            filler1:       Integer;    {reserved}
            zipBuffPtr:    Ptr;        {returned zone names}
            zipNumZones:   Integer;    {number of zones returned}
            zipLastFlag:   Byte;       {nonzero when all zone }
                                       { names have been returned}
            filler2:       Byte;       {reserved}
            zipInfoField:  PACKED ARRAY[1..70] OF Byte;
                                       {reserved for use by .XPP}
   END;
   XPPParmBlkPtr = ^XPPParamBlock;
```

Routines

Obtaining Zone Information

```
FUNCTION GetMyZone          (thePBptr: XPPParmBlkPtr; async: Boolean): OSErr;
FUNCTION GetLocalZones      (thePBptr: XPPParmBlkPtr; async: Boolean): OSErr;
FUNCTION GetZoneList        (thePBptr: XPPParmBlkPtr; async: Boolean): OSErr;
```

C Summary

Constants

```
/*MPP parameter constants*/
#define MPPioCompletion MPP.ioCompletion
#define MPPioResult MPP.ioResult
#define MPPioRefNum MPP.ioRefNum
#define MPPcsCode MPP.csCode

enum {                             /*.XPP csCode*/
   xCall            =     246};    /*csCode for .XPP extended calls*/

enum {                             /*.XPP driver unit and reference */
                                   /* numbers*/
   xppUnitNum       =     40,      /*XPP unit number */
   xppRefNum        =     -41};    /*XPP reference number */
```

```
enum {                        /*XPP routine selectors*/
   zipGetLocalZones  =  5,    /*routine selector for local zone names*/
   zipGetZoneList    =  6,    /*routine selector for internet zone list*/
   zipGetMyZone      =  7};   /*routine selector for node's zone name*/
```

Data Types

The XPP Parameter Block for ZIP

```
#define XPPPBHeader
   QElem          *qLink;           /*reserved*/\
   short          qType;            /*reserved*/\
   short          ioTrap;           /*reserved */\
   Ptr            ioCmdAddr;        /*reserved*/\
   ProcPtr        ioCompletion;     /*completion routine*/\
   OSErr          ioResult;         /*result code*/\
   long           cmdResult;        /*reserved*/\
   short          ioVRefNum;        /*reserved*/\
   short          ioRefNum;         /*driver reference number*/
   short          csCode;           /*primary command code*/

typedef struct {
   XPPPBHeader
      short       xppSubCode;       /*secondary command code*/
      char        xppTimeout;       /*retry interval in seconds*/
      char        xppRetry;         /*retry count*/
      short       filler1;
      Ptr         zipBuffPtr;       /*pointer to buffer of 578 bytes*/
      short       zipNumZones;      /*number of zone names in response*/
      char        zipLastFlag;      /*nonzero if no more zones*/
      char        filler2;          /*filler*/
      char        zipInfoField[70]; /*initial call, set first word to 0*/
}XCallParam;
```

Routines

Obtaining Zone Information

```
pascal OSErr GetMyZone      (XPPParmBlkPtr thePBptr, Boolean async);
pascal OSErr GetLocalZones  (XPPParmBlkPtr thePBptr, Boolean async);
pascal OSErr GetZoneList    (XPPParmBlkPtr thePBptr, Boolean async);
```

Assembly-Language Summary

Constants

XPP csCode

```
xCall            EQU     246   ;csCode for XPP extended calls
```

XPP Driver Unit Reference Number

```
xppUnitNum       EQU     9     ;XPP unit number
```

XPP xCall Subcodes for ZIP Commands

```
ZGetMyZone       EQU     7     ;selector for GetMyZone command
ZGetZoneList     EQU     8     ;selector for GetZoneList command
ZGetLocalZones   EQU     9     ;selector for GetLocalZones command
```

Data Structures

XPP Parameter Block Common Fields for ZIP Routines

0	`qLink`	long	reserved
4	`qType`	word	reserved
6	`ioTrap`	word	reserved
8	`ioCmdAddr`	long	reserved
12	`ioCompletion`	long	address of completion routine
16	`ioResult`	word	result code
18	`cmdResult`	long	reserved
22	`ioVRefNum`	word	reserved
24	`ioRefNum`	word	driver reference number

GetMyZone

28	`xppSubCode`	word	always `zipGetZoneList` for this function
30	`xppTimeout`	byte	retry interval in seconds
31	`xppRetry`	byte	retry count
34	`zipBuffPtr`	long	pointer to data buffer
42	`zipInfoField`	70 bytes	data buffer for use by ZIP; first word set to 0

GetLocalZones

28	`xppSubCode`	word	always `zipGetLocalZones` for this function
30	`xppTimeout`	byte	retry interval in seconds
31	`xppRetry`	byte	retry count
34	`zipBuffPtr`	long	pointer to data buffer
38	`zipNumZones`	word	number of names returned
40	`zipLastFlag`	byte	nonzero if no more names
42	`zipInfoField`	70 bytes	data buffer for use by ZIP; first word set to 0

GetZoneList

28	`xppSubCode`	word	always `zipGetZoneList` for this function
30	`xppTimeout`	byte	retry interval in seconds
31	`xppRetry`	byte	retry count
34	`zipBuffPtr`	long	pointer to data buffer
38	`zipNumZones`	word	number of names returned
40	`zipLastFlag`	byte	nonzero if no more names
42	`zipInfoField`	70 bytes	data buffer for use by ZIP; first word set to 0

Result Codes

`noErr`	0	No error
`noBridgeErr`	–93	No router is available
`reqFailed`	–1096	Request to contact router failed; retry count exceeded
`tooManyReqs`	–1097	Too many concurrent requests
`noDataArea`	–1104	Too many outstanding ATP calls