Open Transport Reference

Carbon > Networking

2005-07-07

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

Tables

Open Transport Reference

Framework:	CoreServices/CoreServices.h
Declared in	OpenTransport.h OpenTransportProtocol.h OpenTransportProviders.h queue.h types.h

Overview

Open Transport is the Mac OS 8 and 9 API for accessing TCP/IP networks, such as the Internet, at the transport level. For Mac OS X, Apple provides Open Transport as a compatibility library to ease migration of legacy applications. As such, Mac OS X does not support the entire Open Transport API.

In new Mac OS X applications you should not use Open Transport but should instead use BSD Sockets or, when possible, higher-level Core Services and Core Foundation APIs such as CFNetwork, CFURL, CFSocket, and CFStream. You can also use Cocoa networking classes such as NSURL, NSURLHandle, and NSNetService.

In Mac OS X, Open Transport provides limited support for endpoints and port access, and no support for the XTI or UNIX STREAMS interfaces. If you want your application to run in Mac OS 8 and 9 and in Mac OS X, use Open Transport for your Mac OS 8 and 9 version and Apple's newer APIs for your Mac OS X version.

For more information about Open Transport, see:

http://developer.apple.com/macos/opentransport/

Mac OS X supports only these Open Transport providers:

- TCP, UDP, and Raw IP Endpoints
- TCP/IP Services Providers and TCP/IP Mapper Providers (for the Domain Name Resolver protocol)
- DDP endpoints, AppleTalk Services Providers, and AppleTalk Mappers (for the Name Binding Protocol)
- OT/PPP endpoints

Mac OS X does not support ADSP, ATP, ASP, PAP, or serial endpoints.

You may have to revise your code if it uses Open Transport in one of the following ways:

- Your application uses a function that directly gains access to a network port. Ports are read-only in Mac OS X. Code that communicates directly with network interfaces must use the IOKit API.
- Your application uses the transaction-based endpoint feature of Open Transport. This feature is not supported in Mac OS X. Removal of this capability should affect only users of AppleTalk protocols such as ASP.

- Your application uses Open Transport's XTI interfaces or UNIX STREAMS interfaces. Mac OS X will not support these interfaces. You can obtain similar functionality using supported high-level functions.
- In Mac OS X, one cannot assume that Open Transport deferred tasks and notifiers procedures run at deferred task level. They may be preempted by the main event loop or another Mac OS X thread. You should always use atomic operations to access data shared between deferred tasks and notifiers and main system tasks.

Mac OS X does not support functions for:

- accessing Open Transport hash lists
- accessing the Open Transport port name or icon
- directly manipulating CFM or ASLM libraries

Client context parameters have been added to a number of OT functions. (An OT client is an application or a shared library.) Each client of Open Transport now has its own client context so that OT can track resources it allocates on behalf of the client. OT resources are objects like endpoints, timer tasks, and blocks of memory. To find out more about Open Transport resources management, see *"Understanding Open Transport Asset Tracking"* at:

http://developer.apple.com/technotes/tn/tn1173.html

Mac OS X introduces a new type, `OTClientContextPtr`, that represents the OT client context. This new type is passed as an extra parameter to functions that allocate OT resources. Before Mac OS X, the OT client context was determined by the Open Transport static libraries that you linked to your application. Now the OT client context is determined explicitly. The same Carbon binary can run on Mac OS 8/9 and Mac OS X, and you do not have to link your application to the static libraries.

You can use `InitOpenTransportinContext` to replace `InitOpenTransport`. It functions identically except that it also takes a client context pointer and a flags parameter to indicate whether you are initializing OT for an application or a shared library. When your application or shared library is done using Open Transport you should call `CloseOpenTransportInContext` to dispose of the Open Transport resources allocated for the client.

The following functions now take a client context:

- `CloseOpenTransportInContext` (page 361)
- `OTAllocInContext` (page 368)
- `OTAllocMemInContext` (page 369)
- `OTAsyncOpenAppleTalkServicesInContext` (page 369)
- `OTAsyncOpenEndpointInContext` (page 370)
- `OTAsyncOpenInternetServicesInContext` (page 370)
- `OTAsyncOpenMapperInContext` (page 371)
- `OTCreateDeferredTaskInContext` (page 388)
- `OTCreateTimerTaskInContext` (page 389)
- `OTOpenAppleTalkServicesInContext` (page 431)
- `OTOpenEndpointInContext` (page 432)

- OTOpenInternetServicesInContext (page 433)
- OTOpenMapperInContext (page 433)

As a convenience, applications may pass a null pointer to these routines and Open Transport will use the context that was passed to InitOpenTransport. However, shared libraries must always pass a valid OTClientContextPtr.

If you want to keep your application source code compatible with pre-Mac OS X systems, you may define the C preprocessor constant OTCARBONAPPLICATION to 1 to use the old routine names without the "InContext" suffix.

Mac OS X applications must pass UPPs instead of procedure pointers for Open Transport callback routines. You can use these new functions to create UPPs:

OTNotifyUPP replaces OTNotifyProcPtr

OTProcessUPP replaces OTNotifyProcPtr

OTListSearchUPP replaces OTListSearchProcPtr

You can use these functions to allocate and free UPPs:

- NewOTNotifyUPP (page 365)
- DisposeOTNotifyUPP (page 362)
- NewOTProcessUPP (page 365)
- DisposeOTProcessUPP (page 362)
- NewOTListSearchUPP (page 364)
- DisposeOTListSearchUPP (page 361)

These functions have been modified to take an OTNotifyUPP UPP instead of a procedure pointer:

- OTAsyncOpenAppleTalkServicesInContext (page 369)
- OTAsyncOpenInternetServicesInContext (page 370)
- OTInstallNotifier (page 420)
- OTAsyncOpenEndpointInContext (page 370)
- OTAsyncOpenMapperInContext (page 371)

These functions have been modified to take an OTProcessUPP UPP instead of a procedure pointer:

- OTCreateTimerTaskInContext (page 389)
- OTCreateDeferredTaskInContext (page 388)

These functions have been modified to take an OTListSearchUPP UPP instead of a procedure pointer:

- OTFindLink (page 398)
- OTFindAndRemoveLink (page 397)

Functions by Task

Initializing and Closing Open Transport

`CloseOpenTransportInContext` (page 361) Deprecated in Mac OS X v10.4
Unregisters your application or code resource connection to Open Transport.

`InitOpenTransportInContext` (page 362) Deprecated in Mac OS X v10.4
Initializes the parts of Open Transport for use by the application or code resource.

Creating, Cloning, and Disposing of a Configuration Structure

`OTCloneConfiguration` (page 381) Deprecated in Mac OS X v10.4
Copies an `OTConfiguration` structure.

`OTCreateConfiguration` (page 387) Deprecated in Mac OS X v10.4
Creates a structure defining a provider's configuration.

`OTDestroyConfiguration` (page 392) Deprecated in Mac OS X v10.4
Deletes an `OTConfiguration` structure.

Opening and Closing Providers

`OTCloseProvider` (page 382) Deprecated in Mac OS X v10.4
Closes a provider of any type—endpoint, mapper, or service provider.

Controlling a Provider's Modes of Operation

`OTAckSends` (page 366) Deprecated in Mac OS X v10.4
Specifies that a provider make an internal copy of data being sent and that it notify you when it has finished sending data.

`OTCancelSynchronousCalls` (page 380) Deprecated in Mac OS X v10.4
Cancels any currently executing synchronous function for a specified provider.

`OTDontAckSends` (page 394) Deprecated in Mac OS X v10.4
Specifies that a provider copy data before sending it.

`OTIsAckingSends` (page 422) Deprecated in Mac OS X v10.4
Determines whether a provider is acknowledging sends.

`OTIsSynchronous` (page 423) Deprecated in Mac OS X v10.4
Returns a provider's current mode of execution.

`OTSetAsynchronous` (page 449) Deprecated in Mac OS X v10.4
Sets a provider's mode of execution to asynchronous.

`OTSetBlocking` (page 450) Deprecated in Mac OS X v10.4
Allows a provider to wait or block until it is able to send or receive data.

`OTSetNonBlocking` (page 455) Deprecated in Mac OS X v10.4
Disallows a provider from waiting if it cannot currently complete a function that sends or receives data.

`OTSetSynchronous` (page 456) Deprecated in Mac OS X v10.4
Sets a provider's mode of execution to synchronous.

Using Notifier Functions with Providers

`OTEnterNotifier` (page 395) Deprecated in Mac OS X v10.4
Limits the notifications that can be sent to your notifier.

`OTInstallNotifier` (page 420) Deprecated in Mac OS X v10.4
Installs a notifier function.

`OTLeaveNotifier` (page 423) Deprecated in Mac OS X v10.4
Allows Open Transport to resume sending primary and completion events.

`OTRemoveNotifier` (page 445) Deprecated in Mac OS X v10.4
Removes a provider's notifier function.

`OTUseSyncIdleEvents` (page 465) Deprecated in Mac OS X v10.4
Allows synchronous idle events to be sent to your notifier.

Sending Module-Specific Commands to Providers

`OTIoctl` (page 421) Deprecated in Mac OS X v10.4
Sends a module-specific command to an Open Transport protocol module.

Creating Endpoints

`OTAsyncOpenEndpointInContext` (page 370) Deprecated in Mac OS X v10.4
Opens an endpoint and installs a notifier callback function for the endpoint.

`OTOpenEndpointInContext` (page 432) Deprecated in Mac OS X v10.4
Opens an endpoint that operates synchronously.

Binding and Unbinding Endpoints

`OTBind` (page 378) Deprecated in Mac OS X v10.4
Assigns an address to an endpoint.

`OTUnbind` (page 464) Deprecated in Mac OS X v10.4
Dissociates an endpoint from its address or cancels an asynchronous call to the `OTBind` function.

Obtaining Information About an Endpoint

`OTGetEndpointInfo` (page 403) Deprecated in Mac OS X v10.4
Obtains information about an endpoint that has been opened.

`OTGetEndpointState` (page 404) Deprecated in Mac OS X v10.4
Obtains the current state of an endpoint.

`OTGetProtAddress` (page 407) Deprecated in Mac OS X v10.4
Obtains the address to which an endpoint is bound and, if the endpoint is currently connected, obtains the address of its peer.

`OTLook` (page 426) Deprecated in Mac OS X v10.4
Determines the current asynchronous event pending for an endpoint.

`OTResolveAddress` (page 446) Deprecated in Mac OS X v10.4
Returns the protocol address that corresponds to the name of an endpoint.

Allocating Structures for Endpoints

`OTFree` (page 400) Deprecated in Mac OS X v10.4
Frees memory allocated using the OTAlloc function.

Determining if Bytes Are Available for Endpoints

`OTCountDataBytes` (page 386) Deprecated in Mac OS X v10.4
Returns the amount of data currently available to be read.

Functions for Connectionless Transactionless Endpoints

`OTRcvUData` (page 440) Deprecated in Mac OS X v10.4
Reads data sent by a client using a connectionless transactionless protocol.

`OTRcvUDErr` (page 441) Deprecated in Mac OS X v10.4
Clears an error condition indicated by a `T_UDERR` event and returns the reason for the error.

`OTSndUData` (page 459) Deprecated in Mac OS X v10.4
Sends data using a connectionless transactionless endpoint.

Establishing Connection for Endpoints

`OTAccept` (page 366) Deprecated in Mac OS X v10.4
Accepts an incoming connection request.

`OTConnect` (page 385) Deprecated in Mac OS X v10.4
Requests a connection to a remote peer.

`OTListen` (page 425) Deprecated in Mac OS X v10.4
Listens for an incoming connection request.

`OTRcvConnect` (page 438) Deprecated in Mac OS X v10.4
Reads the status of an outstanding or completed asynchronous call to the`OTConnect` function.

Functions for Connection-Oriented Transactionless Endpoints

`OTRcv` (page 436) Deprecated in Mac OS X v10.4
Reads data sent using a connection-oriented transactionless protocol.

`OTSnd` (page 456) Deprecated in Mac OS X v10.4
Sends data to a remote peer.

Tearing Down an Endpoint Connection

`OTRcvDisconnect` (page 438) Deprecated in Mac OS X v10.4
Identifies the cause of a connection break or of a connection rejection, acknowledges and clears the corresponding disconnection event.

`OTRcvOrderlyDisconnect` (page 439) Deprecated in Mac OS X v10.4
Acknowledges a request for an orderly disconnect.

`OTSndDisconnect` (page 458) Deprecated in Mac OS X v10.4
Tears down an open connection (abortive disconnect) or rejects an incoming connection request.

`OTSndOrderlyDisconnect` (page 458) Deprecated in Mac OS X v10.4
Initiates or completes an orderly disconnection.

Checking Synchronous Calls

`OTCanMakeSyncCall` (page 381) Deprecated in Mac OS X v10.4
Checks whether you can call a synchronous function.

Working With Timer Tasks

`OTCancelTimerTask` (page 380) Deprecated in Mac OS X v10.4
Cancels a task that was already scheduled for execution.

`OTCreateTimerTaskInContext` (page 389) Deprecated in Mac OS X v10.4
Creates a task to be scheduled.

`OTDestroyTimerTask` (page 393) Deprecated in Mac OS X v10.4
Disposes of a timer task.

`OTScheduleTimerTask` (page 448) Deprecated in Mac OS X v10.4
Schedules a timer task to be executed at the specified time.

Working With Deferred Tasks

`OTCreateDeferredTaskInContext` (page 388) Deprecated in Mac OS X v10.4
Creates a reference to a task that can be scheduled to run at deferred task time.

`OTDestroyDeferredTask` (page 393) Deprecated in Mac OS X v10.4
Destroys a deferred task created with the `OTCreateDeferredTask` function.

`OTScheduleDeferredTask` (page 447) Deprecated in Mac OS X v10.4
Schedules a task for execution at deferred task time.

Creating Mappers

`OTAsyncOpenMapperInContext` (page 371) Deprecated in Mac OS X v10.4
Creates an asynchronous mapper and installs a notifier function for the mapper provider.

`OTOpenMapperInContext` (page 433) Deprecated in Mac OS X v10.4
Creates a synchronous mapper provider and returns a mapper reference.

Registering and Deleting Names with Mappers

`OTDeleteName` (page 390) Deprecated in Mac OS X v10.4
Removes a previously registered entity name.

`OTDeleteNameByID` (page 391) Deprecated in Mac OS X v10.4
Removes a previously registered name as specified by its name ID.

`OTRegisterName` (page 443) Deprecated in Mac OS X v10.4
Registers an entity name on the network.

Looking Up Names for Mappers

`OTLookupName` (page 427) Deprecated in Mac OS X v10.4
Finds and returns all addresses that correspond to a particular name or name pattern, or confirms that a name is registered.

Determining and Changing Option Values

`OTOptionManagement` (page 434) Deprecated in Mac OS X v10.4
Determines an endpoint's current or default option values or changes these values.

Finding Options

`OTFindOption` (page 398) Deprecated in Mac OS X v10.4
Finds a specific option in an options buffer.

`OTNextOption` (page 430) Deprecated in Mac OS X v10.4
Locates the next `TOption` structure in a buffer.

Getting Information About Ports

`OTFindPort` (page 399) Deprecated in Mac OS X v10.4
Obtains information about a port that corresponds to a given port name.

`OTFindPortByRef` (page 400) Deprecated in Mac OS X v10.4
Obtains information about a port that corresponds to its given port reference.

`OTGetBusTypeFromPortRef` (page 402) Deprecated in Mac OS X v10.4
Extracts the value of the bus type from a port reference.

`OTGetDeviceTypeFromPortRef` (page 402) Deprecated in Mac OS X v10.4
Extracts the value of the hardware device type from a port reference.

`OTGetIndexedPort` (page 405) Deprecated in Mac OS X v10.4
Iterates through the ports available on your computer.

`OTGetSlotFromPortRef` (page 408) Deprecated in Mac OS X v10.4
Extracts slot information from a port reference.

Registering New Ports

`OTCreatePortRef` (page 388) Deprecated in Mac OS X v10.4
Creates a port reference that describes a port's hardware characteristics.

Registering as a Client

`OTRegisterAsClientInContext` (page 442) Deprecated in Mac OS X v10.4
Registers your application as a client of Open Transport and gives Open Transport a notifier function it can use to send you events.

`OTUnregisterAsClientInContext` (page 464) Deprecated in Mac OS X v10.4
Removes your application as a client of Open Transport.

Allocating and Freeing Memory

`OTAllocMemInContext` (page 369) Deprecated in Mac OS X v10.4
Allocates memory using an explicit client context.

`OTFreeMem` (page 401) Deprecated in Mac OS X v10.4
Frees memory allocated with the `OTAllocMem` function.

Memory Manipulation Utility Functions

`OTMemcmp` (page 428) Deprecated in Mac OS X v10.4
Compares the contents of two memory locations.

`OTMemcpy` (page 429) Deprecated in Mac OS X v10.4
Copies data from one memory location to another; the source and destination locations must not overlap.

`OTMemmove` (page 429) Deprecated in Mac OS X v10.4
Copies data from one memory location to another; the source and destination locations may overlap.

`OTMemset` (page 429) Deprecated in Mac OS X v10.4
Sets the specified memory range to a specific value.

`OTMemzero` (page 430) Deprecated in Mac OS X v10.4
Initializes the specified memory range to 0.

Idling and Delaying Processing

`OTDelay` (page 390) Deprecated in Mac OS X v10.4
: Delays processing for a specified number of seconds. This function is only provided for compatibility with the UNIX `sleep` function.

`OTIdle` (page 409) Deprecated in Mac OS X v10.4
: Idles your computer.

String Manipulation Utility Functions

`OTStrCat` (page 460) Deprecated in Mac OS X v10.4
: Concatenates two C strings.

`OTStrCopy` (page 461) Deprecated in Mac OS X v10.4
: Copies a C string.

`OTStrEqual` (page 461) Deprecated in Mac OS X v10.4
: Determines whether two C strings are the same.

`OTStrLength` (page 461) Deprecated in Mac OS X v10.4
: Returns the length of a C string.

Timestamp Utility Functions

`OTElapsedMicroseconds` (page 394) Deprecated in Mac OS X v10.4
: Calculates the time elapsed in microseconds since a specified time.

`OTElapsedMilliseconds` (page 394) Deprecated in Mac OS X v10.4
: Calculates the time elapsed in milliseconds since a specified time.

`OTGetClockTimeInSecs` (page 402) Deprecated in Mac OS X v10.4
: Returns the number of seconds that have elapsed since system boot time.

`OTGetTimeStamp` (page 409) Deprecated in Mac OS X v10.4
: Obtains the current timestamp.

`OTSubtractTimeStamps` (page 462) Deprecated in Mac OS X v10.4
: Subtracts one timestamp value from another.

`OTTimeStampInMicroseconds` (page 463) Deprecated in Mac OS X v10.4
: Calculates the time elapsed in microseconds since since a specified time.

`OTTimeStampInMilliseconds` (page 463) Deprecated in Mac OS X v10.4
: Calculates the time elapsed in milliseconds since since a specified time.

OTLIFO List Utility Functions

`OTLIFODequeue` (page 424) Deprecated in Mac OS X v10.4
: Removes the first link in a LIFO list and returns a pointer to it.

`OTLIFOEnqueue` (page 424) Deprecated in Mac OS X v10.4
: Places a link at the front of a LIFO list.

OTLIFOStealList (page 425) Deprecated in Mac OS X v10.4
Removes all links in a LIFO list and returns a pointer to the first link in the list.

OTReverseList (page 446) Deprecated in Mac OS X v10.4
Reverses the order in which entries are linked in a list.

OTFIFO List Utility Functions

OTAddFirst (page 367) Deprecated in Mac OS X v10.4
Places a link at the front of a FIFO list.

OTAddLast (page 367) Deprecated in Mac OS X v10.4
Adds a link to the end of a FIFO list.

OTFindAndRemoveLink (page 397) Deprecated in Mac OS X v10.4
Finds a link in a FIFO list and removes it.

OTFindLink (page 398) Deprecated in Mac OS X v10.4
Finds a link in a FIFO list and returns a pointer to it.

OTGetFirst (page 404) Deprecated in Mac OS X v10.4
Returns a pointer to the first element in a FIFO list.

OTGetIndexedLink (page 405) Deprecated in Mac OS X v10.4
Returns a pointer to the link at a specified position in a FIFO list.

OTGetLast (page 406) Deprecated in Mac OS X v10.4
Returns the last element in a FIFO list.

OTIsInList (page 422) Deprecated in Mac OS X v10.4
Determines whether the specified link is in the specified list.

OTRemoveFirst (page 444) Deprecated in Mac OS X v10.4
Removes the first link in a FIFO list.

OTRemoveLast (page 444) Deprecated in Mac OS X v10.4
Removes the last link in a FIFO list.

OTRemoveLink (page 445) Deprecated in Mac OS X v10.4
Removes the last link in a FIFO list.

Adding and Removing List Elements

OTDequeue (page 392) Deprecated in Mac OS X v10.4
Removes an element from a list.

OTEnqueue (page 395) Deprecated in Mac OS X v10.4
Adds an element to a list.

Atomic Operations

OTAtomicAdd16 (page 375) Deprecated in Mac OS X v10.4
Atomically adds a 16-bit value to a memory location.

OTAtomicAdd32 (page 375) Deprecated in Mac OS X v10.4
Atomically adds a 32-bit value to a memory location.

`OTAtomicAdd8` (page 376) Deprecated in Mac OS X v10.4
Atomically adds an 8-bit value to a memory location.

`OTAtomicClearBit` (page 376) Deprecated in Mac OS X v10.4
Clears a bit in a byte.

`OTAtomicSetBit` (page 377) Deprecated in Mac OS X v10.4
Sets a specified bit in a byte.

`OTAtomicTestBit` (page 377) Deprecated in Mac OS X v10.4
Tests a bit in a byte and returns its current state.

`OTCompareAndSwap16` (page 383) Deprecated in Mac OS X v10.4
Atomically compares two 16-bit values and changes one of these values if they are the same.

`OTCompareAndSwap32` (page 383) Deprecated in Mac OS X v10.4
Atomically compares two 32-bit values and changes one of these values if they are the same.

`OTCompareAndSwap8` (page 384) Deprecated in Mac OS X v10.4
Atomically compares two 8-bit values and changes one of these values if they are the same.

`OTCompareAndSwapPtr` (page 384) Deprecated in Mac OS X v10.4
Atomically compares the value of a pointer at a memory location and atomically swaps it with a second pointer value if the compare is successful.

Handling No-Copy Receives

`OTBufferDataSize` (page 379) Deprecated in Mac OS X v10.4
Obtains the size of the no-copy receive buffer.

`OTReadBuffer` (page 442) Deprecated in Mac OS X v10.4
Copies data out of a no-copy receive buffer.

`OTReleaseBuffer` (page 443) Deprecated in Mac OS X v10.4
Returns the no-copy receive buffer to the system.

Resolving Internet Addresses

`OTInetAddressToName` (page 409) Deprecated in Mac OS X v10.4
Determines the canonical domain name of the host associated with an internet address.

`OTInetStringToAddress` (page 414) Deprecated in Mac OS X v10.4
Resolves a domain name to its equivalent internet addresses.

Opening a TCP/IP Service Provider

`OTAsyncOpenInternetServicesInContext` (page 370) Deprecated in Mac OS X v10.4
Opens the TCP/IP service provider and returns an Internet services reference.

`OTOpenInternetServicesInContext` (page 433) Deprecated in Mac OS X v10.4
Opens the TCP/IP service provider and returns an internet services reference.

Getting Information About an Internet Host

`OTInetMailExchange` (page 412) Deprecated in Mac OS X v10.4
Returns mail-exchange-host names and preference information for a domain name you specify.

`OTInetSysInfo` (page 415) Deprecated in Mac OS X v10.4
Returns details about a host's processor and operating system.

Retrieving DNS Query Information

`OTInetQuery` (page 413) Deprecated in Mac OS X v10.4
Executes a generic DNS query.

Internet Address Utilities

`OTInetGetInterfaceInfo` (page 410) Deprecated in Mac OS X v10.4
Returns internet address information about the local host.

`OTInetHostToString` (page 411) Deprecated in Mac OS X v10.4
Converts an an address in InetHost format into a character string in dotted-decimal notation.

`OTInetStringToHost` (page 415) Deprecated in Mac OS X v10.4
Converts an IP address string from dotted-decimal notation or hexadecimal notation to an `InetHost` data type.

`OTInitDNSAddress` (page 417) Deprecated in Mac OS X v10.4
Fills in a DNSAddress structure with the data you provide.

`OTInitInetAddress` (page 418) Deprecated in Mac OS X v10.4
Fills in an InetAddress structure with the data you provide.

Single Link Multi-Homing

`OTInetGetSecondaryAddresses` (page 411) Deprecated in Mac OS X v10.4
Returns the active secondary IP addresses.

AppleTalk Utility Functions

`OTCompareDDPAddresses` (page 384) Deprecated in Mac OS X v10.4
Compares two DDP address structures.

`OTExtractNBPName` (page 396) Deprecated in Mac OS X v10.4
Extracts the name part of an NBP name from an NBP entity structure.

`OTExtractNBPType` (page 396) Deprecated in Mac OS X v10.4
Extracts the type part of an NBP name from an NBP entity structure.

`OTExtractNBPZone` (page 397) Deprecated in Mac OS X v10.4
Extracts the zone part of an NBP name from an NBP entity structure.

`OTGetNBPEntityLengthAsAddress` (page 407) Deprecated in Mac OS X v10.4
Obtains the size of an NBP entity structure.

`OTInitDDPAddress` (page 416) Deprecated in Mac OS X v10.4
Initializes a DDP address structure.

`OTInitDDPNBPAddress` (page 417) Deprecated in Mac OS X v10.4
Initializes a combined DDP-NBP address structure.

`OTInitNBPAddress` (page 419) Deprecated in Mac OS X v10.4
Initializes an NBP address structure.

`OTInitNBPEntity` (page 419) Deprecated in Mac OS X v10.4
Initializes an NBP entity structure.

`OTSetAddressFromNBPEntity` (page 448) Deprecated in Mac OS X v10.4
Stores an NBP entity structure as an NBP address string.

`OTSetAddressFromNBPString` (page 449) Deprecated in Mac OS X v10.4
Copies an NBP name string into an NBP address buffer.

`OTSetNBPEntityFromAddress` (page 453) Deprecated in Mac OS X v10.4
Parses and stores an NBP address into an NBP entity.

`OTSetNBPName` (page 453) Deprecated in Mac OS X v10.4
Stores the name part of an NBP name into an NBP entity structure.

`OTSetNBPType` (page 454) Deprecated in Mac OS X v10.4
Stores the type part of an NBP name in an NBP entity structure.

`OTSetNBPZone` (page 455) Deprecated in Mac OS X v10.4
Stores the zone part of an NBP name in an NBP entity structure.

Opening an AppleTalk Service Provider

`OTAsyncOpenAppleTalkServicesInContext` (page 369) Deprecated in Mac OS X v10.4
Opens an asynchronous AppleTalk service provider in context.

`OTOpenAppleTalkServicesInContext` (page 431) Deprecated in Mac OS X v10.4
Opens a synchronous AppleTalk service provider.

Obtaining Information About Zones

`OTATalkGetLocalZones` (page 373) Deprecated in Mac OS X v10.4
Obtains a list of the zones available on your network.

`OTATalkGetMyZone` (page 373) Deprecated in Mac OS X v10.4
Obtains the AppleTalk zone name of the node on which your application is running.

`OTATalkGetZoneList` (page 374) Deprecated in Mac OS X v10.4
Obtains a list of all the zones available on the AppleTalk internet.

Obtaining Information About Your AppleTalk Environment

`OTATalkGetInfo` (page 372) Deprecated in Mac OS X v10.4
Obtains information about the AppleTalk environment for a given node.

Miscellaneous Functions

DisposeOTListSearchUPP (page 361) Deprecated in Mac OS X v10.4
Disposes of a universal procedure pointer (UPP) to a list search callback.

DisposeOTNotifyUPP (page 362) Deprecated in Mac OS X v10.4
Disposes of a universal procedure pointer (UPP) to a notification callback.

DisposeOTProcessUPP (page 362) Deprecated in Mac OS X v10.4
Disposes of a universal procedure pointer (UPP) to a process callback.

InvokeOTListSearchUPP (page 363) Deprecated in Mac OS X v10.4
Calls a list search callback.

InvokeOTNotifyUPP (page 364) Deprecated in Mac OS X v10.4
Calls a notification callback.

InvokeOTProcessUPP (page 364) Deprecated in Mac OS X v10.4
Calls a process callback.

NewOTListSearchUPP (page 364) Deprecated in Mac OS X v10.4
Creates a new universal procedure pointer (UPP) to a list search callback.

NewOTNotifyUPP (page 365) Deprecated in Mac OS X v10.4
Creates a new universal procedure pointer (UPP) to a notification callback.

NewOTProcessUPP (page 365) Deprecated in Mac OS X v10.4
Creates a new universal procedure pointer (UPP) to a process callback.

OTAllocInContext (page 368) Deprecated in Mac OS X v10.4
Allocates a data structure of a specified type.

OTClearBit (page 381) Deprecated in Mac OS X v10.4
Clears a bit atomically.

OTIsBlocking (page 422) Deprecated in Mac OS X v10.4
Returns a boolean indicating whether a provider is blocking.

OTSetBit (page 450) Deprecated in Mac OS X v10.4
Sets a bit atomically.

OTSetBusTypeInPortRef (page 451) Deprecated in Mac OS X v10.4
Sets bus type for a port reference.

OTSetDeviceTypeInPortRef (page 452) Deprecated in Mac OS X v10.4
Sets device type for a port reference.

OTSetFirstClearBit (page 452) Deprecated in Mac OS X v10.4
Atomcially sets the first clear bit in a specified bit map.

OTTestBit (page 462) Deprecated in Mac OS X v10.4
Atomically tests a bit in a specified bit map.

Callbacks by Task

Notifier Callbacks

OTNotifyProcPtr (page 51)

System, Timer, and Deferred Task Callbacks

OTProcessProcPtr (page 51)

Linked List Callbacks

OTListSearchProcPtr (page 50)

Miscellaneous Callbacks

admin_t (page 39)

bufcall_t (page 40)

bufcallp_t (page 40)

closeOld_t (page 41)

closep_t (page 41)

esbbcallProc (page 42)

FreeFuncType (page 42)

old_closep_t (page 42)

old_openp_t (page 43)

openOld_t (page 43)

openp_t (page 44)

Callbacks

admin_t

```
typedef OTInt32 (*admin_t) ();
```

If you name your function `MyAdmin_tCallback`, you would declare it like this:

```
OTInt32 MyAdmin_tCallback ();
```

Parameters

Return Value
See the description of the `OTInt32` data type.

Carbon Porting Notes

Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.

bufcallp_t

```
typedef void (*bufcallp_t) (
    SInt32 size
);
```

If you name your function `MyBufcallp_tCallback`, you would declare it like this:

```
void MyBufcallp_tCallback (
    SInt32 size
);
```

Parameters

size

Carbon Porting Notes

Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.

bufcall_t

```
typedef void (*bufcall_t) (
    SInt32 size
);
```

If you name your function `MyBufcall_tCallback`, you would declare it like this:

```
void MyBufcall_tCallback (
    SInt32 size
);
```

Parameters

size

Carbon Porting Notes

Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.

closeOld_t

```
typedef OTInt32 (*closeOld_t) (
    queue *q
);
```

If you name your function `MyCloseOld_tCallback`, you would declare it like this:

```
OTInt32 MyCloseOld_tCallback (
    queue *q
);
```

Parameters

q

Return Value

See the description of the `OTInt32` data type.

Carbon Porting Notes

Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.

closep_t

```
typedef OTInt32 (*closep_t) (
    queue *q,
    OTInt32 foo,
    cred_t *cred
);
```

If you name your function `MyClosep_tCallback`, you would declare it like this:

```
OTInt32 MyClosep_tCallback (
    queue *q,
    OTInt32 foo,
    cred_t *cred
);
```

Parameters

q

foo

cred

Return Value

See the description of the `OTInt32` data type.

Carbon Porting Notes

Carbon does not support Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.

esbbcallProc

```
typedef void (*esbbcallProc) (
    SInt32 arg
);
```

If you name your function `MyEsbbcallCallback`, you would declare it like this:

```
void MyEsbbcallCallback (
    SInt32 arg
);
```

Parameters

arg

Carbon Porting Notes

This function is not needed in Carbon because the STREAMS subsystem is not available on Mac OS X.

FreeFuncType

```
typedef void (*FreeFuncType) (
    char *arg
);
```

If you name your function `MyFreeFuncTypeCallback`, you would declare it like this:

```
void MyFreeFuncTypeCallback (
    char *arg
);
```

Parameters

arg

Carbon Porting Notes

This function is not needed in Carbon because the STREAMS subsystem is not available on Mac OS X.

old_closep_t

```
typedef OTInt32 (*old_closep_t) (
    queue *q
);
```

If you name your function `MyOld_closep_tCallback`, you would declare it like this:

```
OTInt32 MyOld_closep_tCallback (
    queue *q
);
```

Parameters

q

Return Value
See the description of the `OTInt32` data type.

Carbon Porting Notes

Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.

old_openp_t

```
typedef OTInt32 (*old_openp_t) (
    queue *q,
    dev_t dev,
    OTInt32 foo,
    OTInt32 bar
);
```

If you name your function `MyOld_openp_tCallback`, you would declare it like this:

```
OTInt32 MyOld_openp_tCallback (
    queue *q,
    dev_t dev,
    OTInt32 foo,
    OTInt32 bar
);
```

Parameters

q
dev
foo
bar

Return Value
See the description of the `OTInt32` data type.

Carbon Porting Notes

Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.

openOld_t

```
typedef OTInt32 (*openOld_t) (
    queue *q,
    dev_t dev,
    OTInt32 foo,
    OTInt32 bar
);
```

If you name your function `MyOpenOld_tCallback`, you would declare it like this:

```
OTInt32 MyOpenOld_tCallback (
    queue *q,
    dev_t dev,
    OTInt32 foo,
    OTInt32 bar
);
```

Parameters

q

dev

foo

bar

Return Value

See the description of the `OTInt32` data type.

Carbon Porting Notes

Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.

openp_t

```
typedef OTInt32 (*openp_t) (
    queue *q,
    dev_t *dev,
    OTInt32 foo,
    OTInt32 bar,
    cred_t *cred
);
```

If you name your function `MyOpenp_tCallback`, you would declare it like this:

```
OTInt32 MyOpenp_tCallback (
    queue *q,
    dev_t *dev,
    OTInt32 foo,
    OTInt32 bar,
    cred_t *cred
);
```

Parameters

q

dev

foo

bar

cred

Return Value

See the description of the `OTInt32` data type.

Carbon Porting Notes

Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.

OTAllocMemProcPtr

```
typedef void (*OTAllocMemProcPtr) (
    OTByteCount size
);
```

If you name your function `MyOTAllocMemProc`, you would declare it like this:

```
void MyOTAllocMemProc (
    OTByteCount size
);
```

Parameters

size

Carbon Porting Notes

Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.

OTCanConfigureProcPtr

```
typedef Boolean (*OTCanConfigureProcPtr)
(
    OTConfigurationRef cfig,
    UInt32 pass
);
```

If you name your function `MyOTCanConfigureProc`, you would declare it like this:

```
Boolean MyOTCanConfigureProc (
    OTConfigurationRef cfig,
    UInt32 pass
);
```

Parameters

cfig

pass

Carbon Porting Notes

Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.

OTCFConfigureProcPtr

```
typedef OSStatus (*OTCFConfigureProcPtr)
(
    TOTConfiguratorRef cfigor,
    OTConfigurationRef cfig
);
```

If you name your function `MyOTCFConfigureProc`, you would declare it like this:

```
OSStatus MyOTCFConfigureProc (
    TOTConfiguratorRef cfigor,
    OTConfigurationRef cfig
);
```

Parameters

cfigor

cfig

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Carbon Porting Notes

Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.

OTCFCreateStreamProcPtr

```
typedef OSStatus (*OTCFCreateStreamProcPtr)
(
    TOTConfiguratorRef cfigor,
    OTConfigurationRef cfig,
    OTOpenFlags oFlags,
    OTNotifyUPP proc,
    void *contextPtr
);
```

If you name your function `MyOTCFCreateStreamProc`, you would declare it like this:

```
OSStatus MyOTCFCreateStreamProc (
    TOTConfiguratorRef cfigor,
    OTConfigurationRef cfig,
    OTOpenFlags oFlags,
    OTNotifyUPP proc,
    void *contextPtr
);
```

Parameters

cfigor

cfig

oFlags

proc

contextPtr

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Carbon Porting Notes

Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.

OTCFHandleSystemEventProcPtr

```
typedef void (*OTCFHandleSystemEventProcPtr)
(
    TOTConfiguratorRef cfigor,
    OTEventCode code,
    OTResult result,
    void *cookie
);
```

If you name your function `MyOTCFHandleSystemEventProc`, you would declare it like this:

```
void MyOTCFHandleSystemEventProc (
    TOTConfiguratorRef cfigor,
    OTEventCode code,
    OTResult result,
    void *cookie
);
```

Parameters

cfigor

code

result

cookie

Carbon Porting Notes

Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.

OTCreateConfiguratorProcPtr

```
typedef OSStatus (*OTCreateConfiguratorProcPtr)
(
    TOTConfiguratorRef *cfigor
);
```

If you name your function `MyOTCreateConfiguratorProc`, you would declare it like this:

```
OSStatus MyOTCreateConfiguratorProc (
    TOTConfiguratorRef *cfigor
);
```

Parameters

cfigor

Return Value

A result code. See "Open Transport Result Codes" (page 354).

Carbon Porting Notes

Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.

OTGateProcPtr

```
typedef Boolean (*OTGateProcPtr) (
    OTLink *thisLink
);
```

If you name your function `MyOTGateProc`, you would declare it like this:

```
Boolean MyOTGateProc (
    OTLink *thisLink
);
```

Parameters

thisLink

Carbon Porting Notes

Carbon does not support Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

OTGetPortIconProcPtr

```
typedef Boolean (*OTGetPortIconProcPtr)
(
    OTPortRecord *port,
    OTResourceLocator *iconLocation
);
```

If you name your function `MyOTGetPortIconProc`, you would declare it like this:

```
Boolean MyOTGetPortIconProc (
    OTPortRecord *port,
    OTResourceLocator *iconLocation
);
```

Parameters

port

iconLocation

Carbon Porting Notes

Carbon does not support access to the Open Transport port name or icon because this information is not available on Mac OS X.

OTGetPortNameProcPtr

```
typedef void (*OTGetPortNameProcPtr)
(
    OTPortRecord *port,
    OTBooleanParam includeSlot,
    OTBooleanParam includePort,
    Str255 userVisibleName
);
```

If you name your function `MyOTGetPortNameProc`, you would declare it like this:

```
void MyOTGetPortNameProc (
    OTPortRecord *port,
    OTBooleanParam includeSlot,
    OTBooleanParam includePort,
    Str255 userVisibleName
);
```

Parameters

port

includeSlot

includePort

userVisibleName

Carbon Porting Notes

Carbon does not support access to the Open Transport port name or icon because this information is not available on Mac OS X.

OTHashProcPtr

```
typedef UInt32 (*OTHashProcPtr) (
    OTLink *linkToHash
);
```

If you name your function `MyOTHashProc`, you would declare it like this:

```
UInt32 MyOTHashProc (
    OTLink *linkToHash
);
```

Parameters

linkToHash

Carbon Porting Notes

Carbon does not support Open Transport hash lists because Apple has not identified a developer need for them.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

OTHashSearchProcPtr

```
typedef Boolean (*OTHashSearchProcPtr)
(
    const void *ref,
    OTLink *linkToCheck
);
```

If you name your function `MyOTHashSearchProc`, you would declare it like this:

```
Boolean MyOTHashSearchProc (
    const void *ref,
    OTLink *linkToCheck
);
```

Parameters

ref

linkToCheck

Carbon Porting Notes

Carbon does not support Open Transport hash lists because Apple has not identified a developer need for them.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

OTListSearchProcPtr

```
typedef Boolean (*OTListSearchProcPtr)
(
    const void *ref,
    OTLink *linkToCheck
);
```

If you name your function `MyOTListSearchProc`, you would declare it like this:

```
Boolean MyOTListSearchProc (
    const void *ref,
    OTLink *linkToCheck
);
```

Parameters

ref

linkToCheck

Carbon Porting Notes

This is a function type for a user callback. Use the type `OTListSearchUPP` instead.

Availability

Available in Mac OS X v10.0 and later.

Declared In
OpenTransport.h

OTNotifyProcPtr

```
typedef void (*OTNotifyProcPtr) (
    void *contextPtr,
    OTEventCode code,
    OTResult result,
    void *cookie
);
```

If you name your function `MyOTNotifyProc`, you would declare it like this:

```
void MyOTNotifyProc (
    void *contextPtr,
    OTEventCode code,
    OTResult result,
    void *cookie
);
```

Parameters

contextPtr

code

result

cookie

Carbon Porting Notes

This is a function type for a callback. Use the type `OTNotifyUPP` instead.

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransport.h

OTProcessProcPtr

```
typedef void (*OTProcessProcPtr) (
    void *arg
);
```

If you name your function `MyOTProcessProc`, you would declare it like this:

```
void MyOTProcessProc (
    void *arg
);
```

Parameters

arg

Carbon Porting Notes

Use the `OTProcessUPP` type instead.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

OTSetupConfiguratorProcPtr

```
typedef OSStatus (*OTSetupConfiguratorProcPtr)
(
    OTCanConfigureProcPtr *canConfigure,
    OTCreateConfiguratorProcPtr *createConfigurator,
    UInt8 *configuratorType
);
```

If you name your function `MyOTSetupConfiguratorProc`, you would declare it like this:

```
OSStatus MyOTSetupConfiguratorProc (
    OTCanConfigureProcPtr *canConfigure,
    OTCreateConfiguratorProcPtr *createConfigurator,
    UInt8 *configuratorType
);
```

Parameters

canConfigure

createConfigurator

configuratorType

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Carbon Porting Notes

Carbon does not support Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.

OTSMCompleteProcPtr

```
typedef void (*OTSMCompleteProcPtr) (
    void *contextPtr
);
```

If you name your function `MyOTSMCompleteProc`, you would declare it like this:

```
void MyOTSMCompleteProc (
    void *contextPtr
```

```
);
```

Parameters

contextPtr

Carbon Porting Notes

Carbon does not support Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.

OTStateProcPtr

```
typedef void (*OTStateProcPtr) (
    OTStateMachine *sm
);
```

If you name your function `MyOTStateProc`, you would declare it like this:

```
void MyOTStateProc (
    OTStateMachine *sm
);
```

Parameters

sm

Carbon Porting Notes

Carbon does not support Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.

putp_t

```
typedef OTInt32 (*putp_t) (
    queue *q,
    msgb *mp
);
```

If you name your function `MyPutp_tCallback`, you would declare it like this:

```
OTInt32 MyPutp_tCallback (
    queue *q,
    msgb *mp
);
```

Parameters

q

mp

Return Value

See the description of the `OTInt32` data type.

Carbon Porting Notes

Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.

srvp_t

```
typedef OTInt32 (*srvp_t) (
    queue *q
);
```

If you name your function `MySrvp_tCallback`, you would declare it like this:

```
OTInt32 MySrvp_tCallback (
    queue *q
);
```

Parameters

q

Return Value

See the description of the `OTInt32` data type.

Carbon Porting Notes

Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.

Data Types

AppleTalkInfo

Obtain informations about the current AppleTalk environment.

```
struct AppleTalkInfo {
    DDPAddress fOurAddress;
    DDPAddress fRouterAddress;
    UInt16 fCableRange[2];
    UInt16 fFlags;
};
typedef struct AppleTalkInfo AppleTalkInfo;
```

Fields

`fOurAddress`
: The network number and node ID of your node.

`fRouterAddress`
: The network number and node ID of the closest router on your network.

`fCableRange`
: A two-element array indicating the first and last network numbers for the current extended network to which the machine is connected. For nonextended networks, this returns the name of the zone.

`fFlags`
A set of flag bits that describe the network. See `kATalkInfoIsExtended` (page 244).

Discussion
Use the AppleTalk information structure to obtain information about the current AppleTalk environment for the node on which your application is running.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProviders.h`

ATSvcRef

```
typedef struct OpaqueATSvcRef * ATSvcRef;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProviders.h`

bandinfo

```
struct bandinfo {
    unsigned char bi_pri;
    char pad1;
    SInt32 bi_flag;
};
typedef struct bandinfo bandinfo;
```

Fields
`bi_pri`
`pad1`
`bi_flag`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

boolean_p

```
typedef Boolean boolean_p;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

caddr_t

```
typedef char * caddr_t;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
types.h

CCMiscInfo

```
struct CCMiscInfo {
    UInt32 connectionStatus;
    UInt32 connectionTimeElapsed;
    UInt32 connectionTimeRemaining;
    UInt32 bytesTransmitted;
    UInt32 bytesReceived;
    UInt32 reserved;
};
typedef struct CCMiscInfo CCMiscInfo;
```

Fields
connectionStatus
connectionTimeElapsed
connectionTimeRemaining
bytesTransmitted
bytesReceived
reserved

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProviders.h

CFMLibraryInfo

```
struct CFMLibraryInfo {
    OTLink link;
    char * libName;
    StringPtr intlName;
    FSSpec * fileSpec;
    StringPtr pstring2;
    StringPtr pstring3;
};
typedef struct CFMLibraryInfo CFMLibraryInfo;
```

Fields

`link`

`libName`

`intlName`

`fileSpec`

`pstring2`

`pstring3`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

char_p

```
typedef SInt8 char_p;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

copyreq

```
struct copyreq {
    SInt32 cq_cmd;
    cred * cq_cr;
    UInt32 cq_id;
    caddr_t cq_addr;
    UInt32 cq_size;
    SInt32 cq_flag;
    mblk_t * cq_private;
    long cq_filler[4];
};
typedef struct copyreq copyreq;
```

Fields

`cq_cmd`

`cq_cr`

`cq_id`

`cq_addr`

`cq_size`

`cq_flag`

`cq_private`

`cq_filler`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

copyresp

```
struct copyresp {
    SInt32 cp_cmd;
    cred * cp_cr;
    UInt32 cp_id;
    caddr_t cp_rval;
    UInt32 cp_pad1;
    SInt32 cp_pad2;
    mblk_t * cp_private;
    long cp_filler[4];
};
typedef struct copyresp copyresp;
```

Fields

`cp_cmd`

`cp_cr`

`cp_id`

`cp_rval`

`cp_pad1`

`cp_pad2`

`cp_private`

`cp_filler`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

cred

```
struct cred {
    UInt16 cr_ref;
    UInt16 cr_ngroups;
    uid_t cr_uid;
    gid_t cr_gid;
    uid_t cr_ruid;
    gid_t cr_rgid;
    uid_t cr_suid;
    gid_t cr_sgid;
    gid_t cr_groups[1];
};
typedef struct cred cred;
typedef cred cred_t;
```

Fields

`cr_ref`

`cr_ngroups`

`cr_uid`

`cr_gid`

`cr_ruid`

`cr_rgid`

`cr_suid`

`cr_sgid`

`cr_groups`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

cred_t

```
typedef cred cred_t;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

datab

```
struct datab {
    datab_db_f db_f;
    unsigned char * db_base;
    unsigned char * db_lim;
    unsigned char db_ref;
    unsigned char db_type;
    unsigned char db_iswhat;
    unsigned char db_filler2;
    UInt32 db_size;
    unsigned char * db_msgaddr;
    long db_filler;
};
typedef struct datab datab;
typedef datab dblk_t;
```

Fields

`db_f`

`db_base`

`db_lim`

`db_ref`

`db_type`

`db_iswhat`

`db_filler2`

`db_size`

`db_msgaddr`

`db_filler`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

datab_db_f

```
union datab_db_f {
    datab * freep;
    free_rtn * frtnp;
};
typedef union datab_db_f datab_db_f;
```

Fields

`freep`

`frtnp`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

dblk_t

```
typedef datab dblk_t;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

DDPAddress

```
struct DDPAddress {
    OTAddressType fAddressType;
    UInt16 fNetwork;
    UInt8 fNodeID;
    UInt8 fSocket;
    UInt8 fDDPType;
    UInt8 fPad;
};
typedef struct DDPAddress DDPAddress;
```

Fields
`fAddressType`
`fNetwork`
`fNodeID`
`fSocket`
`fDDPType`
`fPad`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProviders.h`

DDPNBPAddress

```
struct DDPNBPAddress {
    OTAddressType fAddressType;
    UInt16 fNetwork;
    UInt8 fNodeID;
    UInt8 fSocket;
    UInt8 fDDPType;
    UInt8 fPad;
    UInt8 fNBPNameBuffer[105];
};
typedef struct DDPNBPAddress DDPNBPAddress;
```

Fields

`fAddressType`

`fNetwork`

`fNodeID`

`fSocket`

`fDDPType`

`fPad`

`fNBPNameBuffer`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProviders.h`

dev_t

```
typedef UInt32 dev_t;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`types.h`

dl_attach_req_t

```
struct dl_attach_req_t {
    UInt32 dl_primitive;
    UInt32 dl_ppa;
};
typedef struct dl_attach_req_t dl_attach_req_t;
```

Fields

`dl_primitive`

`dl_ppa`

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

dl_bind_ack_t

```
struct dl_bind_ack_t {
    UInt32 dl_primitive;
    UInt32 dl_sap;
    UInt32 dl_addr_length;
    UInt32 dl_addr_offset;
    UInt32 dl_max_conind;
    UInt32 dl_xidtest_flg;
};
typedef struct dl_bind_ack_t dl_bind_ack_t;
```

Fields
dl_primitive
dl_sap
dl_addr_length
dl_addr_offset
dl_max_conind
dl_xidtest_flg

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

dl_bind_req_t

```
struct dl_bind_req_t {
    UInt32 dl_primitive;
    UInt32 dl_sap;
    UInt32 dl_max_conind;
    UInt16 dl_service_mode;
    UInt16 dl_conn_mgmt;
    UInt32 dl_xidtest_flg;
};
typedef struct dl_bind_req_t dl_bind_req_t;
```

Fields
dl_primitive
dl_sap
dl_max_conind
dl_service_mode
dl_conn_mgmt
dl_xidtest_flg

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

dl_connect_con_t

```
struct dl_connect_con_t {
    UInt32 dl_primitive;
    UInt32 dl_resp_addr_length;
    UInt32 dl_resp_addr_offset;
    UInt32 dl_qos_length;
    UInt32 dl_qos_offset;
    UInt32 dl_growth;
};
typedef struct dl_connect_con_t dl_connect_con_t;
```

Fields
dl_primitive
dl_resp_addr_length
dl_resp_addr_offset
dl_qos_length
dl_qos_offset
dl_growth

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

dl_connect_ind_t

```
struct dl_connect_ind_t {
    UInt32 dl_primitive;
    UInt32 dl_correlation;
    UInt32 dl_called_addr_length;
    UInt32 dl_called_addr_offset;
    UInt32 dl_calling_addr_length;
    UInt32 dl_calling_addr_offset;
    UInt32 dl_qos_length;
    UInt32 dl_qos_offset;
    UInt32 dl_growth;
};
typedef struct dl_connect_ind_t dl_connect_ind_t;
```

Fields

`dl_primitive`

`dl_correlation`

`dl_called_addr_length`

`dl_called_addr_offset`

`dl_calling_addr_length`

`dl_calling_addr_offset`

`dl_qos_length`

`dl_qos_offset`

`dl_growth`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

dl_connect_req_t

```
struct dl_connect_req_t {
    UInt32 dl_primitive;
    UInt32 dl_dest_addr_length;
    UInt32 dl_dest_addr_offset;
    UInt32 dl_qos_length;
    UInt32 dl_qos_offset;
    UInt32 dl_growth;
};
typedef struct dl_connect_req_t dl_connect_req_t;
```

Fields

`dl_primitive`

`dl_dest_addr_length`

`dl_dest_addr_offset`

`dl_qos_length`

`dl_qos_offset`

`dl_growth`

Availability

Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

dl_connect_res_t

```
struct dl_connect_res_t {
    UInt32 dl_primitive;
    UInt32 dl_correlation;
    UInt32 dl_resp_token;
    UInt32 dl_qos_length;
    UInt32 dl_qos_offset;
    UInt32 dl_growth;
};
typedef struct dl_connect_res_t dl_connect_res_t;
```

Fields
dl_primitive
dl_correlation
dl_resp_token
dl_qos_length
dl_qos_offset
dl_growth

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

dl_data_ack_ind_t

```
struct dl_data_ack_ind_t {
    UInt32 dl_primitive;
    UInt32 dl_dest_addr_length;
    UInt32 dl_dest_addr_offset;
    UInt32 dl_src_addr_length;
    UInt32 dl_src_addr_offset;
    UInt32 dl_priority;
    UInt32 dl_service_class;
};
typedef struct dl_data_ack_ind_t dl_data_ack_ind_t;
```

Fields
dl_primitive
dl_dest_addr_length
dl_dest_addr_offset
dl_src_addr_length
dl_src_addr_offset
dl_priority
dl_service_class

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

dl_data_ack_req_t

```
struct dl_data_ack_req_t {
    UInt32 dl_primitive;
    UInt32 dl_correlation;
    UInt32 dl_dest_addr_length;
    UInt32 dl_dest_addr_offset;
    UInt32 dl_src_addr_length;
    UInt32 dl_src_addr_offset;
    UInt32 dl_priority;
    UInt32 dl_service_class;
};
typedef struct dl_data_ack_req_t dl_data_ack_req_t;
```

Fields
dl_primitive
dl_correlation
dl_dest_addr_length
dl_dest_addr_offset
dl_src_addr_length
dl_src_addr_offset
dl_priority
dl_service_class

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

dl_data_ack_status_ind_t

```
struct dl_data_ack_status_ind_t {
    UInt32 dl_primitive;
    UInt32 dl_correlation;
    UInt32 dl_status;
};
typedef struct dl_data_ack_status_ind_t dl_data_ack_status_ind_t;
```

Fields
dl_primitive
dl_correlation
dl_status

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

dl_detach_req_t

```
struct dl_detach_req_t {
    UInt32 dl_primitive;
};
typedef struct dl_detach_req_t dl_detach_req_t;
```

Fields
`dl_primitive`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

dl_disabmulti_req_t

```
struct dl_disabmulti_req_t {
    UInt32 dl_primitive;
    UInt32 dl_addr_length;
    UInt32 dl_addr_offset;
};
typedef struct dl_disabmulti_req_t dl_disabmulti_req_t;
```

Fields
`dl_primitive`
`dl_addr_length`
`dl_addr_offset`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

dl_disconnect_ind_t

```
struct dl_disconnect_ind_t {
    UInt32 dl_primitive;
    UInt32 dl_originator;
    UInt32 dl_reason;
    UInt32 dl_correlation;
};
typedef struct dl_disconnect_ind_t dl_disconnect_ind_t;
```

Fields
`dl_primitive`
`dl_originator`
`dl_reason`
`dl_correlation`

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

dl_disconnect_req_t

```
struct dl_disconnect_req_t {
    UInt32 dl_primitive;
    UInt32 dl_reason;
    UInt32 dl_correlation;
};
typedef struct dl_disconnect_req_t dl_disconnect_req_t;
```

Fields
dl_primitive
dl_reason
dl_correlation

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

dl_enabmulti_req_t

```
struct dl_enabmulti_req_t {
    UInt32 dl_primitive;
    UInt32 dl_addr_length;
    UInt32 dl_addr_offset;
};
typedef struct dl_enabmulti_req_t dl_enabmulti_req_t;
```

Fields
dl_primitive
dl_addr_length
dl_addr_offset

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

dl_error_ack_t

```
struct dl_error_ack_t {
    UInt32 dl_primitive;
    UInt32 dl_error_primitive;
    UInt32 dl_errno;
    UInt32 dl_unix_errno;
};
typedef struct dl_error_ack_t dl_error_ack_t;
```

Fields

`dl_primitive`

`dl_error_primitive`

`dl_errno`

`dl_unix_errno`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

dl_get_statistics_ack_t

```
struct dl_get_statistics_ack_t {
    UInt32 dl_primitive;
    UInt32 dl_stat_length;
    UInt32 dl_stat_offset;
};
typedef struct dl_get_statistics_ack_t dl_get_statistics_ack_t;
```

Fields

`dl_primitive`

`dl_stat_length`

`dl_stat_offset`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

dl_get_statistics_req_t

```
struct dl_get_statistics_req_t {
    UInt32 dl_primitive;
};
typedef struct dl_get_statistics_req_t dl_get_statistics_req_t;
```

Fields

`dl_primitive`

Availability

Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

dl_info_ack_t

```
struct dl_info_ack_t {
    UInt32 dl_primitive;
    UInt32 dl_max_sdu;
    UInt32 dl_min_sdu;
    UInt32 dl_addr_length;
    UInt32 dl_mac_type;
    UInt32 dl_reserved;
    UInt32 dl_current_state;
    SInt32 dl_sap_length;
    UInt32 dl_service_mode;
    UInt32 dl_qos_length;
    UInt32 dl_qos_offset;
    UInt32 dl_qos_range_length;
    UInt32 dl_qos_range_offset;
    UInt32 dl_provider_style;
    UInt32 dl_addr_offset;
    UInt32 dl_version;
    UInt32 dl_brdcst_addr_length;
    UInt32 dl_brdcst_addr_offset;
    UInt32 dl_growth;
};
typedef struct dl_info_ack_t dl_info_ack_t;
```

Fields
dl_primitive
dl_max_sdu
dl_min_sdu
dl_addr_length
dl_mac_type
dl_reserved
dl_current_state
dl_sap_length
dl_service_mode
dl_qos_length
dl_qos_offset
dl_qos_range_length
dl_qos_range_offset
dl_provider_style
dl_addr_offset
dl_version
dl_brdcst_addr_length
dl_brdcst_addr_offset
dl_growth

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

dl_info_req_t

```
struct dl_info_req_t {
    UInt32 dl_primitive;
};
typedef struct dl_info_req_t dl_info_req_t;
```

Fields
dl_primitive

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

dl_ok_ack_t

```
struct dl_ok_ack_t {
    UInt32 dl_primitive;
    UInt32 dl_correct_primitive;
};
typedef struct dl_ok_ack_t dl_ok_ack_t;
```

Fields
dl_primitive
dl_correct_primitive

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

dl_phys_addr_ack_t

```
struct dl_phys_addr_ack_t {
    UInt32 dl_primitive;
    UInt32 dl_addr_length;
    UInt32 dl_addr_offset;
};
typedef struct dl_phys_addr_ack_t dl_phys_addr_ack_t;
```

Fields
dl_primitive
dl_addr_length
dl_addr_offset

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

dl_phys_addr_req_t

```
struct dl_phys_addr_req_t {
    UInt32 dl_primitive;
    UInt32 dl_addr_type;
};
typedef struct dl_phys_addr_req_t dl_phys_addr_req_t;
```

Fields
dl_primitive
dl_addr_type

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

DL_primitives

```
union DL_primitives {
    UInt32 dl_primitive;
    dl_info_req_t info_req;
    dl_info_ack_t info_ack;
    dl_attach_req_t attach_req;
    dl_detach_req_t detach_req;
    dl_bind_req_t bind_req;
    dl_bind_ack_t bind_ack;
    dl_unbind_req_t unbind_req;
    dl_subs_bind_req_t subs_bind_req;
    dl_subs_bind_ack_t subs_bind_ack;
    dl_subs_unbind_req_t subs_unbind_req;
    dl_ok_ack_t ok_ack;
    dl_error_ack_t error_ack;
    dl_connect_req_t connect_req;
    dl_connect_ind_t connect_ind;
    dl_connect_res_t connect_res;
    dl_connect_con_t connect_con;
    dl_token_req_t token_req;
    dl_token_ack_t token_ack;
    dl_disconnect_req_t disconnect_req;
    dl_disconnect_ind_t disconnect_ind;
    dl_reset_req_t reset_req;
    dl_reset_ind_t reset_ind;
    dl_reset_res_t reset_res;
    dl_reset_con_t reset_con;
    dl_unitdata_req_t unitdata_req;
    dl_unitdata_ind_t unitdata_ind;
    dl_uderror_ind_t uderror_ind;
    dl_udqos_req_t udqos_req;
    dl_enabmulti_req_t enabmulti_req;
    dl_disabmulti_req_t disabmulti_req;
    dl_promiscon_req_t promiscon_req;
    dl_promiscoff_req_t promiscoff_req;
    dl_phys_addr_req_t physaddr_req;
    dl_phys_addr_ack_t physaddr_ack;
    dl_set_phys_addr_req_t set_physaddr_req;
    dl_get_statistics_req_t get_statistics_req;
    dl_get_statistics_ack_t get_statistics_ack;
    dl_test_req_t test_req;
    dl_test_ind_t test_ind;
    dl_test_res_t test_res;
    dl_test_con_t test_con;
    dl_xid_req_t xid_req;
    dl_xid_ind_t xid_ind;
    dl_xid_res_t xid_res;
    dl_xid_con_t xid_con;
    dl_data_ack_req_t data_ack_req;
    dl_data_ack_ind_t data_ack_ind;
    dl_data_ack_status_ind_t data_ack_status_ind;
    dl_reply_req_t reply_req;
    dl_reply_ind_t reply_ind;
    dl_reply_status_ind_t reply_status_ind;
    dl_reply_update_req_t reply_update_req;
    dl_reply_update_status_ind_t reply_update_status_ind;
};
```

```
typedef union DL_primitives DL_primitives;
```

Fields

```
dl_primitive
info_req
info_ack
attach_req
detach_req
bind_req
bind_ack
unbind_req
subs_bind_req
subs_bind_ack
subs_unbind_req
ok_ack
error_ack
connect_req
connect_ind
connect_res
connect_con
token_req
token_ack
disconnect_req
disconnect_ind
reset_req
reset_ind
reset_res
reset_con
unitdata_req
unitdata_ind
uderror_ind
udqos_req
enabmulti_req
disabmulti_req
promiscon_req
promiscoff_req
physaddr_req
physaddr_ack
set_physaddr_req
get_statistics_req
get_statistics_ack
test_req
test_ind
test_res
test_con
xid_req
xid_ind
xid_res
xid_con
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

dl_priority_t

```
struct dl_priority_t {
    SInt32 dl_min;
    SInt32 dl_max;
};
typedef struct dl_priority_t dl_priority_t;
```

Fields
`dl_min`
`dl_max`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

dl_promiscoff_req_t

```
struct dl_promiscoff_req_t {
    UInt32 dl_primitive;
    UInt32 dl_level;
};
typedef struct dl_promiscoff_req_t dl_promiscoff_req_t;
```

Fields
`dl_primitive`
`dl_level`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

dl_promiscon_req_t

```
struct dl_promiscon_req_t {
    UInt32 dl_primitive;
    UInt32 dl_level;
};
typedef struct dl_promiscon_req_t dl_promiscon_req_t;
```

Fields

`dl_primitive`

`dl_level`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

dl_protect_t

```
struct dl_protect_t {
    SInt32 dl_min;
    SInt32 dl_max;
};
typedef struct dl_protect_t dl_protect_t;
```

Fields

`dl_min`

`dl_max`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

dl_qos_cl_range1_t

```
struct dl_qos_cl_range1_t {
    UInt32 dl_qos_type;
    dl_transdelay_t dl_trans_delay;
    dl_priority_t dl_priority;
    dl_protect_t dl_protection;
    SInt32 dl_residual_error;
};
typedef struct dl_qos_cl_range1_t dl_qos_cl_range1_t;
```

Fields

`dl_qos_type`

`dl_trans_delay`

`dl_priority`

`dl_protection`

`dl_residual_error`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

dl_qos_cl_sel1_t

```
struct dl_qos_cl_sel1_t {
    UInt32 dl_qos_type;
    SInt32 dl_trans_delay;
    SInt32 dl_priority;
    SInt32 dl_protection;
    SInt32 dl_residual_error;
};
typedef struct dl_qos_cl_sel1_t dl_qos_cl_sel1_t;
```

Fields

`dl_qos_type`

`dl_trans_delay`

`dl_priority`

`dl_protection`

`dl_residual_error`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

dl_qos_co_range1_t

```
struct dl_qos_co_range1_t {
    UInt32 dl_qos_type;
    dl_through_t dl_rcv_throughput;
    dl_transdelay_t dl_rcv_trans_delay;
    dl_through_t dl_xmt_throughput;
    dl_transdelay_t dl_xmt_trans_delay;
    dl_priority_t dl_priority;
    dl_protect_t dl_protection;
    SInt32 dl_residual_error;
    dl_resilience_t dl_resilience;
};
typedef struct dl_qos_co_range1_t dl_qos_co_range1_t;
```

Fields

`dl_qos_type`

`dl_rcv_throughput`

`dl_rcv_trans_delay`

`dl_xmt_throughput`

`dl_xmt_trans_delay`

`dl_priority`

`dl_protection`

`dl_residual_error`

`dl_resilience`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

dl_qos_co_sel1_t

```
struct dl_qos_co_sel1_t {
    UInt32 dl_qos_type;
    SInt32 dl_rcv_throughput;
    SInt32 dl_rcv_trans_delay;
    SInt32 dl_xmt_throughput;
    SInt32 dl_xmt_trans_delay;
    SInt32 dl_priority;
    SInt32 dl_protection;
    SInt32 dl_residual_error;
    dl_resilience_t dl_resilience;
};
typedef struct dl_qos_co_sel1_t dl_qos_co_sel1_t;
```

Fields

`dl_qos_type`

`dl_rcv_throughput`

`dl_rcv_trans_delay`

`dl_xmt_throughput`

`dl_xmt_trans_delay`

`dl_priority`

`dl_protection`

`dl_residual_error`

`dl_resilience`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

dl_reply_ind_t

```
struct dl_reply_ind_t {
    UInt32 dl_primitive;
    UInt32 dl_dest_addr_length;
    UInt32 dl_dest_addr_offset;
    UInt32 dl_src_addr_length;
    UInt32 dl_src_addr_offset;
    UInt32 dl_priority;
    UInt32 dl_service_class;
};
typedef struct dl_reply_ind_t dl_reply_ind_t;
```

Fields

`dl_primitive`

`dl_dest_addr_length`

`dl_dest_addr_offset`

`dl_src_addr_length`

`dl_src_addr_offset`

`dl_priority`

`dl_service_class`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

dl_reply_req_t

```
struct dl_reply_req_t {
    UInt32 dl_primitive;
    UInt32 dl_correlation;
    UInt32 dl_dest_addr_length;
    UInt32 dl_dest_addr_offset;
    UInt32 dl_src_addr_length;
    UInt32 dl_src_addr_offset;
    UInt32 dl_priority;
    UInt32 dl_service_class;
};
typedef struct dl_reply_req_t dl_reply_req_t;
```

Fields

`dl_primitive`

`dl_correlation`

`dl_dest_addr_length`

`dl_dest_addr_offset`

`dl_src_addr_length`

`dl_src_addr_offset`

`dl_priority`

`dl_service_class`

Availability

Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

dl_reply_status_ind_t

```
struct dl_reply_status_ind_t {
    UInt32 dl_primitive;
    UInt32 dl_correlation;
    UInt32 dl_status;
};
typedef struct dl_reply_status_ind_t dl_reply_status_ind_t;
```

Fields
dl_primitive
dl_correlation
dl_status

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

dl_reply_update_req_t

```
struct dl_reply_update_req_t {
    UInt32 dl_primitive;
    UInt32 dl_correlation;
    UInt32 dl_src_addr_length;
    UInt32 dl_src_addr_offset;
};
typedef struct dl_reply_update_req_t dl_reply_update_req_t;
```

Fields
dl_primitive
dl_correlation
dl_src_addr_length
dl_src_addr_offset

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

dl_reply_update_status_ind_t

```
struct dl_reply_update_status_ind_t {
    UInt32 dl_primitive;
    UInt32 dl_correlation;
    UInt32 dl_status;
};
typedef struct dl_reply_update_status_ind_t dl_reply_update_status_ind_t;
```

Fields

`dl_primitive`

`dl_correlation`

`dl_status`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

dl_reset_con_t

```
struct dl_reset_con_t {
    UInt32 dl_primitive;
};
typedef struct dl_reset_con_t dl_reset_con_t;
```

Fields

`dl_primitive`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

dl_reset_ind_t

```
struct dl_reset_ind_t {
    UInt32 dl_primitive;
    UInt32 dl_originator;
    UInt32 dl_reason;
};
typedef struct dl_reset_ind_t dl_reset_ind_t;
```

Fields

`dl_primitive`

`dl_originator`

`dl_reason`

Availability

Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

dl_reset_req_t

```
struct dl_reset_req_t {
    UInt32 dl_primitive;
};
typedef struct dl_reset_req_t dl_reset_req_t;
```

Fields
dl_primitive

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

dl_reset_res_t

```
struct dl_reset_res_t {
    UInt32 dl_primitive;
};
typedef struct dl_reset_res_t dl_reset_res_t;
```

Fields
dl_primitive

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

dl_resilience_t

```
struct dl_resilience_t {
    SInt32 dl_disc_prob;
    SInt32 dl_reset_prob;
};
typedef struct dl_resilience_t dl_resilience_t;
```

Fields
dl_disc_prob
dl_reset_prob

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

dl_set_phys_addr_req_t

```
struct dl_set_phys_addr_req_t {
    UInt32 dl_primitive;
    UInt32 dl_addr_length;
    UInt32 dl_addr_offset;
};
typedef struct dl_set_phys_addr_req_t dl_set_phys_addr_req_t;
```

Fields

`dl_primitive`

`dl_addr_length`

`dl_addr_offset`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

dl_subs_bind_ack_t

```
struct dl_subs_bind_ack_t {
    UInt32 dl_primitive;
    UInt32 dl_subs_sap_offset;
    UInt32 dl_subs_sap_length;
};
typedef struct dl_subs_bind_ack_t dl_subs_bind_ack_t;
```

Fields

`dl_primitive`

`dl_subs_sap_offset`

`dl_subs_sap_length`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

dl_subs_bind_req_t

```
struct dl_subs_bind_req_t {
    UInt32 dl_primitive;
    UInt32 dl_subs_sap_offset;
    UInt32 dl_subs_sap_length;
    UInt32 dl_subs_bind_class;
};
typedef struct dl_subs_bind_req_t dl_subs_bind_req_t;
```

Fields

`dl_primitive`

`dl_subs_sap_offset`

`dl_subs_sap_length`

`dl_subs_bind_class`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

dl_subs_unbind_req_t

```
struct dl_subs_unbind_req_t {
    UInt32 dl_primitive;
    UInt32 dl_subs_sap_offset;
    UInt32 dl_subs_sap_length;
};
typedef struct dl_subs_unbind_req_t dl_subs_unbind_req_t;
```

Fields

`dl_primitive`

`dl_subs_sap_offset`

`dl_subs_sap_length`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

dl_test_con_t

```
struct dl_test_con_t {
    UInt32 dl_primitive;
    UInt32 dl_flag;
    UInt32 dl_dest_addr_length;
    UInt32 dl_dest_addr_offset;
    UInt32 dl_src_addr_length;
    UInt32 dl_src_addr_offset;
};
typedef struct dl_test_con_t dl_test_con_t;
```

Fields

`dl_primitive`

`dl_flag`

`dl_dest_addr_length`

`dl_dest_addr_offset`

`dl_src_addr_length`

`dl_src_addr_offset`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

dl_test_ind_t

```
struct dl_test_ind_t {
    UInt32 dl_primitive;
    UInt32 dl_flag;
    UInt32 dl_dest_addr_length;
    UInt32 dl_dest_addr_offset;
    UInt32 dl_src_addr_length;
    UInt32 dl_src_addr_offset;
};
typedef struct dl_test_ind_t dl_test_ind_t;
```

Fields

`dl_primitive`

`dl_flag`

`dl_dest_addr_length`

`dl_dest_addr_offset`

`dl_src_addr_length`

`dl_src_addr_offset`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

dl_test_req_t

```
struct dl_test_req_t {
    UInt32 dl_primitive;
    UInt32 dl_flag;
    UInt32 dl_dest_addr_length;
    UInt32 dl_dest_addr_offset;
};
typedef struct dl_test_req_t dl_test_req_t;
```

Fields

`dl_primitive`

`dl_flag`

`dl_dest_addr_length`

`dl_dest_addr_offset`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

dl_test_res_t

```
struct dl_test_res_t {
    UInt32 dl_primitive;
    UInt32 dl_flag;
    UInt32 dl_dest_addr_length;
    UInt32 dl_dest_addr_offset;
};
typedef struct dl_test_res_t dl_test_res_t;
```

Fields

`dl_primitive`

`dl_flag`

`dl_dest_addr_length`

`dl_dest_addr_offset`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

dl_through_t

```
struct dl_through_t {
    SInt32 dl_target_value;
    SInt32 dl_accept_value;
};
typedef struct dl_through_t dl_through_t;
```

Fields

`dl_target_value`

`dl_accept_value`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

dl_token_ack_t

```
struct dl_token_ack_t {
    UInt32 dl_primitive;
    UInt32 dl_token;
};
typedef struct dl_token_ack_t dl_token_ack_t;
```

Fields

`dl_primitive`

`dl_token`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

dl_token_req_t

```
struct dl_token_req_t {
    UInt32 dl_primitive;
};
typedef struct dl_token_req_t dl_token_req_t;
```

Fields

`dl_primitive`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

dl_transdelay_t

```
struct dl_transdelay_t {
    SInt32 dl_target_value;
    SInt32 dl_accept_value;
};
typedef struct dl_transdelay_t dl_transdelay_t;
```

Fields

`dl_target_value`

`dl_accept_value`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

dl_uderror_ind_t

```
struct dl_uderror_ind_t {
    UInt32 dl_primitive;
    UInt32 dl_dest_addr_length;
    UInt32 dl_dest_addr_offset;
    UInt32 dl_unix_errno;
    UInt32 dl_errno;
};
typedef struct dl_uderror_ind_t dl_uderror_ind_t;
```

Fields

`dl_primitive`

`dl_dest_addr_length`

`dl_dest_addr_offset`

`dl_unix_errno`

`dl_errno`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

dl_udqos_req_t

```
struct dl_udqos_req_t {
    UInt32 dl_primitive;
    UInt32 dl_qos_length;
    UInt32 dl_qos_offset;
};
typedef struct dl_udqos_req_t dl_udqos_req_t;
```

Fields

`dl_primitive`

`dl_qos_length`

`dl_qos_offset`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

dl_unbind_req_t

```
struct dl_unbind_req_t {
    UInt32 dl_primitive;
};
typedef struct dl_unbind_req_t dl_unbind_req_t;
```

Fields

`dl_primitive`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

dl_unitdata_ind_t

```
struct dl_unitdata_ind_t {
    UInt32 dl_primitive;
    UInt32 dl_dest_addr_length;
    UInt32 dl_dest_addr_offset;
    UInt32 dl_src_addr_length;
    UInt32 dl_src_addr_offset;
    UInt32 dl_group_address;
};
typedef struct dl_unitdata_ind_t dl_unitdata_ind_t;
```

Fields

`dl_primitive`

`dl_dest_addr_length`

`dl_dest_addr_offset`

`dl_src_addr_length`

`dl_src_addr_offset`

`dl_group_address`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

dl_unitdata_req_t

```
struct dl_unitdata_req_t {
    UInt32 dl_primitive;
    UInt32 dl_dest_addr_length;
    UInt32 dl_dest_addr_offset;
    dl_priority_t dl_priority;
};
typedef struct dl_unitdata_req_t dl_unitdata_req_t;
```

Fields

`dl_primitive`

`dl_dest_addr_length`

`dl_dest_addr_offset`

`dl_priority`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

dl_xid_con_t

```
struct dl_xid_con_t {
    UInt32 dl_primitive;
    UInt32 dl_flag;
    UInt32 dl_dest_addr_length;
    UInt32 dl_dest_addr_offset;
    UInt32 dl_src_addr_length;
    UInt32 dl_src_addr_offset;
};
typedef struct dl_xid_con_t dl_xid_con_t;
```

Fields

`dl_primitive`

`dl_flag`

`dl_dest_addr_length`

`dl_dest_addr_offset`

`dl_src_addr_length`

`dl_src_addr_offset`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

dl_xid_ind_t

```
struct dl_xid_ind_t {
    UInt32 dl_primitive;
    UInt32 dl_flag;
    UInt32 dl_dest_addr_length;
    UInt32 dl_dest_addr_offset;
    UInt32 dl_src_addr_length;
    UInt32 dl_src_addr_offset;
};
typedef struct dl_xid_ind_t dl_xid_ind_t;
```

Fields

`dl_primitive`

`dl_flag`

`dl_dest_addr_length`

`dl_dest_addr_offset`

`dl_src_addr_length`

`dl_src_addr_offset`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

dl_xid_req_t

```
struct dl_xid_req_t {
    UInt32 dl_primitive;
    UInt32 dl_flag;
    UInt32 dl_dest_addr_length;
    UInt32 dl_dest_addr_offset;
};
typedef struct dl_xid_req_t dl_xid_req_t;
```

Fields

`dl_primitive`

`dl_flag`

`dl_dest_addr_length`

`dl_dest_addr_offset`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

dl_xid_res_t

```
struct dl_xid_res_t {
    UInt32 dl_primitive;
    UInt32 dl_flag;
    UInt32 dl_dest_addr_length;
    UInt32 dl_dest_addr_offset;
};
typedef struct dl_xid_res_t dl_xid_res_t;
```

Fields

`dl_primitive`

`dl_flag`

`dl_dest_addr_length`

`dl_dest_addr_offset`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

DNS Address Structure

Holds host names, partially or fully-qualified domain names, or dotted-decimal format Internet addresses for use with a variety of Open Transport functions.

```
struct DNSAddress {
    OTAddressType fAddressType;
    InetDomainName fName;
};
typedef struct DNSAddress DNSAddress;
```

Fields

`fAddressType`

The address type. For a DNSAddress structure, this should be AF_DNS.

`fName`

The name to be resolved by the DNR.

Discussion

You can use the DNS (domain name system) address structure with the OTConnect function (TCP), with the OTSndUData function (UDP), or with the OTResolveAddress function (either TCP or UDP). If you do so, TCP/IP will resolve the name for you automatically. You can use the OTInitDNSAddress function to fill in a DNS address structure. The DNS address structure is defined by the DNSAddress data type.

The address you specify can be just the host name (“otteam”), a partially qualified domain name (“otteam.ssw”), a fully qualified domain name (“otteam.ssw.apple.com.”), or an internet address in dotted-decimal format (“17.202.99.99”), and can optionally include a port number (“otteam.ssw.apple.com:25” or “17.202.99.99:25”).

Because the port number is not actually part of the domain name, it is possible to have a domain name–port number combination that exceeds 255 bytes. If you wish to specify such a string, you must provide a structure based on the DNS address structure that has sufficient space to contain the full string. In any case, the domain name itself cannot exceed 255 bytes.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProviders.h`

DNS Query Information Structure

Returns answers to DNS queries.

```
struct DNSQueryInfo {
    UInt16 qType;
    UInt16 qClass;
    UInt32 ttl;
    InetDomainName name;
    UInt16 responseType;
    UInt16 resourceLen;
    char resourceData[4];
};
typedef struct DNSQueryInfo DNSQueryInfo;
```

Fields

`qType`

The numerical value of the DNS resource record type, such as MX and PTR, for which you wish to query.

`qClass`

The numerical value of the DNS record class, such as Inet and Hesio, for which you wish to query.

`ttl`
: An integer indicating the DNS resource record's time to live (in seconds).

`name`
: The fully qualified domain name or address for which you made the query.

`responseType`
: The type of response.

`resourceLen`
: The actual length of the resource data returned.

`resourceData`
: The resource data that is returned. This is at least 4 bytes long, and is usually longer.

Discussion
The DNS query information structure is used by the TCP/IP service provider to return answers to DNS queries made using the OTInetQuery function. The DNS query information structure is defined by the DNSQueryInfo data type. For additional information about the constant values for the DNSQueryInfo fields, see the DNS Requests for Comments (RFCs), available over the World Wide Web.

See the Internet Standard for a definitive list of values for the qType, qClass, and respnoseType fields, and for a definition of the format of the resource data.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProviders.h`

EndpointRef

```
typedef struct OpaqueEndpointRef * EndpointRef;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

EnetPacketHeader

```
struct EnetPacketHeader {
    UInt8 fDestAddr[6];
    UInt8 fSourceAddr[6];
    UInt16 fProto;
};
typedef struct EnetPacketHeader EnetPacketHeader;
```

Fields

`fDestAddr`

`fSourceAddr`

`fProto`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProviders.h`

free_rtn

```
struct free_rtn {
    FreeFuncType free_func;
    char * free_arg;
};
typedef struct free_rtn free_rtn;
typedef free_rtn frtn_t;
```

Fields

`free_func`

`free_arg`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

frtn_t

```
typedef free_rtn frtn_t;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

gid_t

```
typedef UInt32 gid_t;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`types.h`

Internet Address Structure

Used for providing a TCP or UDP address to the `OTConnect`, `OTSndURequest`, or `OTBind` functions.

```
struct InetAddress {
    OTAddressType fAddressType;
    InetPort fPort;
    InetHost fHost;
    UInt8 fUnused[8];
};
typedef struct InetAddress InetAddress;
```

Fields

`fAddressType`
: The address type. The field should be AF_INET, which identifies the structure as an InetAddress.

`fPort`
: The port number.

`fHost`
: The 32-bit IP address of the host.

`fUnused`
: Reserved.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProviders.h`

InetDHCPOption

```
struct InetDHCPOption {
    UInt8 fOptionTag;
    UInt8 fOptionLen;
    UInt8 fOptionValue;
};
typedef struct InetDHCPOption InetDHCPOption;
```

Fields

`fOptionTag`

`fOptionLen`

`fOptionValue`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProviders.h`

InetDomainName

```
typedef  InetDomainName[256];
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProviders.h`

InetHost

```
typedef UInt32 InetHost;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProviders.h`

Internet Host Information Sructure

Holds a set of IP addresses associated with an Internet host for use by the `OTInetStringToAddress` function.

```
struct InetHostInfo {
    InetDomainName name;
    InetHost addrs[10];
};
typedef struct InetHostInfo InetHostInfo;
```

Fields

`name`

The canonical name of the host. The canonical name is a fully qualified domain nam, never an alias.

`addrs`

Up to ten IP addresses associated with this host name. Only multihomed hosts have more than one IP address.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProviders.h`

Internet Interface Information Structure

Holds information about an Internet interface for use by the `OTInetGetInterfaceInfo` function.

```
struct InetInterfaceInfo {
    InetHost fAddress;
    InetHost fNetmask;
    InetHost fBroadcastAddr;
    InetHost fDefaultGatewayAddr;
    InetHost fDNSAddr;
    UInt16 fVersion;
    UInt16 fHWAddrLen;
    UInt8 * fHWAddr;
    UInt32 fIfMTU;
    UInt8 * fReservedPtrs[2];
    InetDomainName fDomainName;
    UInt32 fIPSecondaryCount;
    UInt8 fReserved[252];
};
typedef struct InetInterfaceInfo InetInterfaceInfo;
```

Fields

`fAddress`

The IP address of the interface.

`fNetmask`

The subnet mask of the local IP network.

`fBroadcastAddr`

The broadcast address for the interface.

`fDefaultGatewayAddr`

The IP address of the default router. The default is a router that can forward any packet destined outside the locally connected subnet.

`fDNSAddr`

The address of a domain name server. This value can be returned by a server or typed in by the user during configuration of the TCP/IP interface.

`fVersion`
: The version of the OTInetGetInterfaceInfo function; currently equal to kInetInterfaceInfoVersion.

`fHWAddrLen`
: The length (in bytes) of the hardware address. This points into the fReserved field of this structure. It can be nil if the interface has no hardware address or if it won't fit.

`fHWAddr`
: A pointer to the hardware address.

`fIfMTU`
: The maximum transmission unit size in bytes permitted for this interface's hardware.

`fReservedPtrs`
: Reserved.

`fDomainName`
: The default domain name of the host as configured in the TCP/IP control panel. This name doesn't include the host name.

`fIPSecondaryCount`
: The number of IP secondary address available.

`fReserved`
: Reserved.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProviders.h`

Internet Mail Exchange Structure

Holds host names and mail preference values for use with the `OTInetMailExchange` function.

```
struct InetMailExchange {
    UInt16 preference;
    InetDomainName exchange;
};
typedef struct InetMailExchange InetMailExchange;
```

Fields
`preference`
: The mail exchange preference value. The mail exchanger with the lowest preference number is the first one to which mail should be sent.

`exchange`
: The fully qualified domain name of a host that can accept mail for your target host.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProviders.h`

InetPort

```
typedef UInt16 InetPort;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProviders.h`

InetSvcRef

```
typedef struct OpaqueInetSvcRef * InetSvcRef;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProviders.h`

InetSysInfo

Holds information about an Internet host for use by the `OTInetSysInfo` function.

```
struct InetSysInfo {
    char cpuType[32];
    char osType[32];
};
typedef struct InetSysInfo InetSysInfo;
```

Fields

`cpuType`
: The CPU type of the specified host. This is an ASCII string maintained by the domain name server.

`osType`
: The operating system running on the specified host. This is an ASCII string maintained by the domain name server.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProviders.h`

install_info

Fields

int_t

```
typedef  int_t;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

iocblk

```
struct iocblk {
    SInt32 ioc_cmd;
    cred * ioc_cr;
    UInt32 ioc_id;
    UInt32 ioc_count;
    SInt32 ioc_error;
    SInt32 ioc_rval;
    long ioc_filler[4];
};
typedef struct iocblk iocblk;
```

Fields
`ioc_cmd`
`ioc_cr`
`ioc_id`
`ioc_count`
`ioc_error`
`ioc_rval`
`ioc_filler`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

LCPEcho

```
struct LCPEcho {
    UInt32 retryCount;
    UInt32 retryPeriod;
};
typedef struct LCPEcho LCPEcho;
```

Fields

`retryCount`

`retryPeriod`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProviders.h`

linkblk

```
struct linkblk {
    queue_t * l_qtop;
    queue_t * l_qbot;
    SInt32 l_index;
    long l_pad[5];
};
typedef struct linkblk linkblk;
```

Fields

`l_qtop`

`l_qbot`

`l_index`

`l_pad`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

log_ctl

```
struct log_ctl {
    short mid;
    short sid;
    char level;
    char pad1;
    short flags;
    long ltime;
    long ttime;
    SInt32 seq_no;
};
typedef struct log_ctl log_ctl;
```

Fields

`mid`

`sid`

`level`

`pad1`

`flags`

`ltime`

`ttime`

`seq_no`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

major_t

```
typedef UInt16 major_t;
```

MapperRef

```
typedef struct OpaqueMapperRef * MapperRef;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

mblk_t

```
typedef msgb mblk_t;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

minor_t

```
typedef UInt16 minor_t;
```

module_info

```
struct module_info {
    unsigned short mi_idnum;
    char * mi_idname;
    long mi_minpsz;
    long mi_maxpsz;
    unsigned long mi_hiwat;
    unsigned long mi_lowat;
};
typedef struct module_info module_info;
typedef module_info * module_infoPtr;
```

Fields
mi_idnum
mi_idname
mi_minpsz
mi_maxpsz
mi_hiwat
mi_lowat

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

module_stat

```
struct module_stat {
    long ms_pcnt;
    long ms_scnt;
    long ms_ocnt;
    long ms_ccnt;
    long ms_acnt;
    char * ms_xptr;
    short ms_xsize;
};
typedef struct module_stat module_stat;
```

Fields

`ms_pcnt`

`ms_scnt`

`ms_ocnt`

`ms_ccnt`

`ms_acnt`

`ms_xptr`

`ms_xsize`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

MPS_INTR_STATE

```
typedef UInt8 MPS_INTR_STATE;
```

msgb

```
struct msgb {
    struct msgb * b_next;
    struct msgb * b_prev;
    struct msgb * b_cont;
    unsigned char * b_rptr;
    unsigned char * b_wptr;
    datab * b_datap;
    unsigned char b_band;
    unsigned char b_pad1;
    unsigned short b_flag;
};
typedef struct msgb msgb;
typedef msgb mblk_t;
```

Fields

`b_next`

`b_prev`

`b_cont`

`b_rptr`

`b_wptr`

`b_datap`

`b_band`

`b_pad1`

`b_flag`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

NBPAddress

```
struct NBPAddress {
    OTAddressType fAddressType;
    UInt8 fNBPNameBuffer[105];
};
typedef struct NBPAddress NBPAddress;
```

Fields

`fAddressType`

`fNBPNameBuffer`

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProviders.h

NBPEntity

```
struct NBPEntity {
    UInt8 fEntity[99];
};
typedef struct NBPEntity NBPEntity;
```

Fields
fEntity

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProviders.h

netbuf

```
struct netbuf {
     maxlen;
     len;
    char *buf;
};
```

Fields

ot_bind

Fields

ot_optmgmt

Fields

OTAddress

Defines the common structure for all Open Transport addresses.

```
struct OTAddress {
    OTAddressType fAddressType;
    UInt8 fAddress[1];
};
typedef struct OTAddress OTAddress;
```

Fields

`fAddressType`

`fAddress`

Discussion

The `OTAddress` type itself is abstract. You would not declare a structure of this type because it does not contain any address information. However, address formats defined by Open Transport protocols all use the `fAddressType` field to describe the format of the fields to follow, which do contain address information.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransport.h`

OTAddressType

```
typedef UInt16 OTAddressType;
```

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransport.h`

OTAutopushInfo

```
struct OTAutopushInfo {
    UInt32 sap_cmd;
    char sap_device_name[32];
    SInt32 sap_minor;
    SInt32 sap_lastminor;
    SInt32 sap_npush;
    char sap_list[8][32];
};
typedef struct OTAutopushInfo OTAutopushInfo;
```

Fields

`sap_cmd`

`sap_device_name`

`sap_minor`

`sap_lastminor`

`sap_npush`

`sap_list`

Availability

Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

OTBand

```
typedef UInt32 OTBand;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransport.h

OTBooleanParam

```
typedef Boolean OTBooleanParam;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransport.h

No-Copy Receive Buffer Structure

Receives data without copying it.

```
struct OTBuffer {
    void * fLink;
    void * fLink2;
    OTBuffer * fNext;
    UInt8 * fData;
    ByteCount fLen;
    void * fSave;
    UInt8 fBand;
    UInt8 fType;
    UInt8 fPad1;
    UInt8 fFlags;
};
typedef struct OTBuffer OTBuffer;
```

Fields

`fLink`
Reserved.

`fLink2`
Reserved.

`fNext`
A pointer to the next OTBuffer structure in the linked chain. By tracing the chain of fNext pointers, you can access all of the data associated with the message.

`fData`
: A pointer to the data portion of this OTBuffer structure.

`fLen`
: The length of data pointed to by the fData field.

`fSave`
: Reserved.

`fBand`
: The band used for the data transmission. It must be a value between 0 and 255.

`fType`
: The type of the data (normally M_DATA, M_PROTO, or M_PCPROTO).

`fPad1`
: Reserved.

`fFlags`
: The flags associated with the data (MSGMARK, MSGDELIM).

Discussion
You use the no-copy receive buffer structure when you wish to receive data without copying it with the OTRcvUData function, the OTRcvURequest function, the OTRcvUReply function, the OTRcv function, the OTRcvRequest function, and the OTRcvReply function.

If you are familiar with STREAMS mblk_t data structures, you can see that the no-copy receive buffer structure is just a slight modification of the mblk_t structure.

You can only use this buffer for data; you cannot use it for the address or options that may be associated with the incoming data. For example, in the case of an incoming TUnitData structure, you can only no-copy receive the udata portion, not the addr or opt fields.

Special Considerations

Under no circumstance write to this data structure. It is read-only. If you write to it, you can crash the system.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

Buffer Information Structure

A convenience structure for keeping track of where your application left off in an OTBuffer structure.

```
struct OTBufferInfo {
    OTBuffer * fBuffer;
    ByteCount fOffset;
    UInt8 fPad;
};
typedef struct OTBufferInfo OTBufferInfo;
```

Fields
`fBuffer`
: A pointer to the no-copy receive buffer.

`fOffset`
: An offset indicating how far into the buffer you have read.

`fPad`
: Reserved.

Discussion
The buffer information structure is provided for your convenience in keeping track of where you last left off in an OTBuffer structure. Because the no-copy receive buffer structure (OTBuffer) is read-only, you may need to copy the data in sections as you progress through the no-copy receive buffer. The utility function OTReadBuffer is used with this structure to easily copy the data out of an OTBuffer structure.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

OTByteCount

```
typedef ByteCount OTByteCount;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

OTClient

```
typedef struct OpaqueOTClient * OTClient;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

OTClientContextPtr

```
typedef struct OpaqueOTClientContextPtr * OTClientContextPtr;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

OTClientList

Identifies the clients that denied a request to yield a port.

```
struct OTClientList {
    ItemCount fNumClients;
    UInt8 fBuffer[4];
};
typedef struct OTClientList OTClientList;
```

Fields

`fNumClients`

The number of clients in the fBuffer array, normally 1.

`fBuffer`

An array of packed Pascal strings enumerating the name of each client that rejected the request—that is, the names under which the clients registered themselves as an Open Transport clients.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

OTClientName

```
typedef UInt8 * OTClientName;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

OTCommand

```
typedef SInt32 OTCommand;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

OTConfigurationRef

```
typedef struct OTConfiguration * OTConfigurationRef;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

OTData Structure

Specifies the location and size of noncontiguous data.

```
struct OTData {
    void * fNext;
    void * fData;
    ByteCount fLen;
};
typedef struct OTData OTData;
```

Fields

`fNext`

A pointer to the OTData structure that describes the next data fragment. Specify a NULL pointer for the last data fragment.

`fData`

A pointer to the data fragment.

`fLen`

Specifies the size of the fragment in bytes.

Discussion

The OTData structure is an Apple extension used to specify the location and size of noncontiguous data. You use a pointer to this structure in place of a pointer to continguous data normally referenced in TNetbuf.buf field. You can send discontiguous data using the OTSndUData function, the OTSndURequest function, the OTSndUReply function, the OTSnd function, the OTSndRequest function, and the OTSndReply function.

Each OTData structure specifies the location of a data fragment, the size of the fragment, and the location of the OTData structure that specifies the location and size of the next data fragment. The data information structure is defined by the OTData type. For more information, see "Sending Noncontiguous Data."

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransport.h`

OTDataSize

```
typedef SInt32 OTDataSize;
```

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransport.h`

OTDeferredTaskRef

```
typedef long OTDeferredTaskRef;
```

Availability

Available in Mac OS X v10.0 and later.

Declared In
OpenTransport.h

OTEventCode

```
typedef UInt32 OTEventCode;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransport.h

OTError

```
typedef SInt32 OTError;
```

OTGate

```
struct OTGate {
    OTLIFO fLIFO;
    OTList fList;
    OTGateProcPtr fProc;
    SInt32 fNumQueued;
    SInt32 fInside;
};
typedef struct OTGate OTGate;
```

Fields
fLIFO
fList
fProc
fNumQueued
fInside

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

OTHashList

```
struct OTHashList {
    OTHashProcPtr fHashProc;
    ByteCount fHashTableSize;
    OTLink ** fHashBuckets;
};
typedef struct OTHashList OTHashList;
```

Fields

`fHashProc`

`fHashTableSize`

`fHashBuckets`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

OTInt32

```
typedef SInt32 OTInt32;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

OTISDNAddress

```
struct OTISDNAddress {
    OTAddressType fAddressType;
    UInt16 fPhoneLength;
    char fPhoneNumber[37];
};
typedef struct OTISDNAddress OTISDNAddress;
```

Fields

`fAddressType`

`fPhoneLength`

`fPhoneNumber`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProviders.h`

OTItemCount

```
typedef ItemCount OTItemCount;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

LIFO List Structure

Supports last-in, first-out lists in Open Transport.

```
struct OTLIFO {
    OTLink * fHead;
};
typedef struct OTLIFO OTLIFO;
```

Fields
`fHead`
A pointer to the first entry in the linked list. Set this to nil to initialize the structure before using it.

Discussion
Open Transport LIFO (last-in, first-out) lists use the LIFO list structure. You must initialize this structure by setting the structure's fHead field to NULL before using the LIFO list. Most Open Transport LIFO list operations are atomic.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

OTLink

Allows any data structure to be used in an Open Transport list.

```
struct OTLink {
    OTLink * fNext;
};
typedef struct OTLink OTLink;
```

Fields
`fNext`
A pointer to the next entry in the linked list.

Discussion
All of Open Transport's list utilities use the linked list structure, which may be embedded in any data structure that you want to use in an Open Transport list. A linked list structure is defined by the OTLink data type.

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransport.h

FIFO List Structure

Supports first-in, last-out lists in Open Transport

```
struct OTList {
    OTLink * fHead;
};
typedef struct OTList OTList;
```

Fields
fHead
A pointer to the first entry in the linked list. Set this to NULL to initialize the structure before using it.

Discussion
Open Transport FIFO (first-in, first-out) lists use the FIFO list structure. You must initialize this structure by setting the structure's fHead field to NULL before using the LIFO list. The FIFO list structure is defined by the OTList data type.

None of the functions that handle a FIFO list structure are atomic.

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransport.h

OTListSearchUPP

```
typedef OTListSearchProcPtr OTListSearchUPP;
```

Discussion
For more information, see the description of the OTListSearchUPP () callback function.

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransport.h

Lock Data Type

Defines a value used to ensure that Open Transport does not recursively reenter locked areas of code.

```
typedef UInt8 OTLock;
```

Discussion
The lock data type defines a value that is used by the OTClearLock function and the OTAcquireLock function to ensure that Open Transport does not recursively reenter locked areas of code. The lock data type is defined by the OTLock data type

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

OTNameID

```
typedef SInt32 OTNameID;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

OTNotifyUPP

```
typedef OTNotifyProcPtr OTNotifyUPP;
```

Discussion
For more information, see the description of the OTNotifyUPP () callback function.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

OTPCIInfo

```
struct OTPCIInfo {
    RegEntryID fTheID;
    void *fConfigurationInfo;
    ByteCount fConfigurationLength;
};
```

Fields

OTPortCloseStruct

Denies or accepts requests to yield a port.

```
struct OTPortCloseStruct {
    OTPortRef fPortRef;
    ProviderRef fTheProvider;
    OSStatus fDenyReason;
};
typedef struct OTPortCloseStruct OTPortCloseStruct;
```

Fields

`fPortRef`

The port requested to be closed.

`fTheProvider`

The provider that is currently using the port.

`fDenyReason`

A value that you can leave untouched to accept the yield request. To deny the request, change this value to a negative error code corresponding to the reason for your denial (normally you use the kOTUserRequestedErr error).

Discussion

When you are using a port that another client wishes to use, the other client can use the OTYieldPortRequest function (not available in Mac OS X) to ask you to yield the port. If you are registered as a client of Open Transport, you receive a kOTYieldPortRequest event, whose cookie parameter is a pointer to a port close structure. You can use this structure to deny or accept the yield request.

Currently, this callback is only used for serial ports, but it is applicable to any hardware device that cannot share a port with multiple clients. You should check the kOTCanYieldPort bit in the port structure's fInfoFlags field to see whether the port supports yielding.

If the provider is passively listening (that is, bound with a queue length (qlen) greater than 0) and you are willing to yield, you need do nothing. If, however, you are actively connected and you are willing to yield the port, you must issue a synchronous OTSndDisconnect callin order to let the port go.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

The Port Structure

Describes a port's characteristics.

```
struct OTPortRecord {
    OTPortRef fRef;
    UInt32 fPortFlags;
    UInt32 fInfoFlags;
    UInt32 fCapabilities;
    ItemCount fNumChildPorts;
    OTPortRef * fChildPorts;
    char fPortName[36];
    char fModuleName[32];
    char fSlotID[8];
    char fResourceInfo[32];
    char fReserved[164];
};
typedef struct OTPortRecord OTPortRecord;
```

Fields

`fRef`

The port reference; a 32-bit value encoding the port's device type, bus type, slot number, and multiport identifier

`fPortFlags`

Flags describing the port's status. If no bits are set, the port is currently inactive—that is, it is not in use at this time.

`fInfoFlags`

`fCapabilities`

`fNumChildPorts`

`fChildPorts`

An array of the port references for the child ports associated with this port. When you get a Port Record, this pointer typically points into the SReserved field at the end of the record.

`fPortName`

A unique name for this port. The port name is a zero-terminated string that can have a maximum length as indicated by the constant kMaxProviderNameSize.

`fModuleName`

The name of the actual STREAMS module that implements the driver for this port. Open Transport uses this name internally; applications rarely need to use this name.

`fSlotID`

An 8-byte identifier for a port's slot that contains a 7-byte character string plus a zero for termination. This identifier is typically available only for PCI cards.

`fResourceInfo`

A zero-terminated string that describes a shared library that can handle configuration information for the device. This field contains an identifier that allows Open Transport to access auxiliary information about the driver (Open Transport creates shared library IDs from this string to be able to find these extra shared libraries). This string should either be unique to the driver or should be set to a NULL string.

`fReserved`

Reserved.

Discussion

Open Transport uses a port structure to describe a port's characteristics, such as its port name, its child ports, whether it is active or disabled, whether it is private or shareable, and the kind of framing it can use.

Availability

Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

OTPortRef

```
typedef UInt32 OTPortRef;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

OTProcessUPP

```
typedef OTProcessProcPtr OTProcessUPP;
```

Discussion
For more information, see the description of the OTProcessUPP () callback function.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

OTQLen

```
typedef UInt32 OTQLen;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

OTReadInfo

```
struct OTReadInfo {
    UInt32 fType;
    OTCommand fCommand;
    UInt32 fFiller;
    ByteCount fBytes;
    OSStatus fError;
};
typedef struct OTReadInfo OTReadInfo;
```

Fields

`fType`

`fCommand`

`fFiller`

`fBytes`

`fError`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

OTReason

```
typedef SInt32 OTReason;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

OTResourceLocator

```
struct OTResourceLocator {
    FSSpec fFile;
    UInt16 fResID;
};
typedef struct OTResourceLocator OTResourceLocator;
```

Fields

`fFile`

`fResID`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

OTResult

```
typedef SInt32 OTResult;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransport.h

OTScriptInfo

```
struct OTScriptInfo {
    UInt32 fScriptType;
    void * fTheScript;
    UInt32 fScriptLength;
};
typedef struct OTScriptInfo OTScriptInfo;
```

Fields
fScriptType
fTheScript
fScriptLength

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransport.h

OTSequence

```
typedef SInt32 OTSequence;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransport.h

OTSInt16Param

```
typedef SInt16 OTSInt16Param;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransport.h

OTSInt8Param

```
typedef SInt8 OTSInt8Param;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransport.h

OTSlotNumber

```
typedef UInt16 OTSlotNumber;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransport.h

OTStateMachine

```
struct OTStateMachine {
    OTStateMachineDataPad fData;
    void * fCookie;
    OTEventCode fCode;
    OTResult fResult;
};
typedef struct OTStateMachine OTStateMachine;
```

Fields
fData
fCookie
fCode
fResult

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

OTStateMachineDataPad

```
typedef UInt8 OTStateMachineDataPad[12];
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

OTSystemTaskRef

```
typedef  OTSystemTaskRef;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

OTTimeout

```
typedef UInt32 OTTimeout;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

OTTimerTask

```
typedef  OTTimerTask;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

Timestamp Data Type

Contains an Open Transport timestamp.

```
typedef UnsignedWide OTTimeStamp;
```

Discussion
The timestamp data type is a 64-bit value that contains an Open Transport timestamp. The timestamp has unspecified units; you must use one of the timestamp manipulation functions described in "Timestamp Utility Functions" to convert the timestamp to known quantities. The timestamp data type is defined by the OTTimeStamp data type.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

OTUInt16Param

```
typedef UInt16 OTUInt16Param;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransport.h

OTUInt32

```
typedef UInt32 OTUInt32;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransport.h

OTUInt8Param

```
typedef UInt8 OTUInt8Param;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransport.h

OTUnixErr

```
typedef UInt16 OTUnixErr;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransport.h

OTXTILevel

```
typedef UInt32 OTXTILevel;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransport.h

OTXTIName

```
typedef UInt32 OTXTIName;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

pollfd

```
struct pollfd {
    SInt32 fd;
    SInt16 events;
    SInt16 revents;
    SInt32 _ifd;
};
```

Fields

PollRef

```
struct PollRef {
    SInt32 filler;
    SInt16 events;
    SInt16 revents;
    StreamRef ref;
};
typedef struct PollRef PollRef;
```

Fields
`filler`
`events`
`revents`
`ref`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

PPPMRULimits

```
struct PPPMRULimits {
    UInt32 mruSize;
    UInt32 upperMRULimit;
    UInt32 lowerMRULimit;
};
typedef struct PPPMRULimits PPPMRULimits;
```

Fields
`mruSize`
`upperMRULimit`
`lowerMRULimit`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProviders.h`

ProviderRef

```
typedef struct OpaqueProviderRef * ProviderRef;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

q_xtra

```
struct q_xtra {
    UInt32 dummy;
};
typedef struct q_xtra q_xtra;
```

Fields
`dummy`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

qband

```
struct qband {
    qband * qb_next;
    unsigned long qb_count;
    msgb * qb_first;
    msgb * qb_last;
    unsigned long qb_hiwat;
    unsigned long qb_lowat;
    unsigned short qb_flag;
    short qb_pad1;
};
typedef struct qband qband;
typedef qband qband_t;
```

Fields

`qb_next`

`qb_count`

`qb_first`

`qb_last`

`qb_hiwat`

`qb_lowat`

`qb_flag`

`qb_pad1`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

qband_t

```
typedef qband qband_t;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

qfields_t

```
typedef qfields qfields_t;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

qinit

```
struct qinit {
    putp_t qi_putp;
    srvp_t qi_srvp;
    openp_t qi_qopen;
    closep_t qi_qclose;
    admin_t qi_qadmin;
    module_info * qi_minfo;
    module_stat * qi_mstat;
};
typedef struct qinit qinit;
```

Fields

`qi_putp`

`qi_srvp`

`qi_qopen`

`qi_qclose`

`qi_qadmin`

`qi_minfo`

`qi_mstat`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

queue

```
struct queue {
    qinit * q_qinfo;
    msgb * q_first;
    msgb * q_last;
    queue * q_next;
    queue_q_u q_u;
    char * q_ptr;
    unsigned long q_count;
    long q_minpsz;
    long q_maxpsz;
    unsigned long q_hiwat;
    unsigned long q_lowat;
    qband * q_bandp;
    unsigned short q_flag;
    unsigned char q_nband;
    unsigned char q_pad1[1];
    q_xtra * q_osx;
    queue * q_ffcp;
    queue * q_bfcp;
};
typedef struct queue queue;
typedef queue * queuePtr;
```

Fields

`q_qinfo`

`q_first`

`q_last`

`q_next`

`q_u`

`q_ptr`

`q_count`

`q_minpsz`

`q_maxpsz`

`q_hiwat`

`q_lowat`

`q_bandp`

`q_flag`

`q_nband`

`q_pad1`

`q_osx`

`q_ffcp`

`q_bfcp`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

queue_q_u

```
union queue_q_u {
    queue * q_u_link;
    sqh_s * q_u_sqh_parent;
};
typedef union queue_q_u queue_q_u;
```

Fields
`q_u_link`
`q_u_sqh_parent`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

queue_t

```
typedef SInt32 queue_t;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`queue.h`

short_p

```
typedef SInt16 short_p;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

sqh_s

```
struct sqh_s {
    UInt32 dummy;
};
typedef struct sqh_s sqh_s;
```

Fields
`dummy`

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

sth_s

```
struct sth_s {
    UInt32 dummy;
};
typedef struct sth_s sth_s;
```

Fields
dummy

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

str_list

```
struct str_list {
    SInt32 sl_nmods;
    str_mlist * sl_modlist;
};
typedef struct str_list str_list;
```

Fields
sl_nmods
sl_modlist

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

str_mlist

```
struct str_mlist {
    char l_name[32];
};
typedef struct str_mlist str_mlist;
```

Fields
l_name

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

strbuf

```
struct strbuf {
    SInt32 maxlen;
    SInt32 len;
    char * buf;
};
typedef struct strbuf strbuf;
```

Fields

`maxlen`

`len`

`buf`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransport.h`

StreamRef

```
typedef struct OpaqueStreamRef * StreamRef;
```

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

streamtab

```
struct streamtab {
    qinit * st_rdinit;
    qinit * st_wrinit;
    qinit * st_muxrinit;
    qinit * st_muxwinit;
};
typedef struct streamtab streamtab;
```

Fields

`st_rdinit`

`st_wrinit`

`st_muxrinit`

`st_muxwinit`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

strfdinsert

```
struct strfdinsert {
    strbuf ctlbuf;
    strbuf databuf;
    long flags;
    long fildes;
    SInt32 offset;
};
typedef struct strfdinsert strfdinsert;
```

Fields

`ctlbuf`

`databuf`

`flags`

`fildes`

`offset`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

strioctl

```
struct strioctl {
    SInt32 ic_cmd;
    SInt32 ic_timout;
    SInt32 ic_len;
    char * ic_dp;
};
typedef struct strioctl strioctl;
```

Fields

`ic_cmd`

`ic_timout`

`ic_len`

`ic_dp`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

stroptions

```
struct stroptions {
    unsigned long so_flags;
    short so_readopt;
    unsigned short so_wroff;
    long so_minpsz;
    long so_maxpsz;
    unsigned long so_hiwat;
    unsigned long so_lowat;
    unsigned char so_band;
    unsigned char so_filler[3];
    unsigned long so_poll_set;
    unsigned long so_poll_clr;
};
typedef struct stroptions stroptions;
```

Fields

`so_flags`

`so_readopt`

`so_wroff`

`so_minpsz`

`so_maxpsz`

`so_hiwat`

`so_lowat`

`so_band`

`so_filler`

`so_poll_set`

`so_poll_clr`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

strpeek

```
struct strpeek {
    strbuf ctlbuf;
    strbuf databuf;
    long flags;
};
typedef struct strpeek strpeek;
```

Fields

`ctlbuf`

`databuf`

`flags`

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

strpfp

```
struct strpfp {
    unsigned long pass_file_cookie;
    unsigned short pass_uid;
    unsigned short pass_gid;
    sth_s * pass_sth;
};
typedef struct strpfp strpfp;
```

Fields
pass_file_cookie
pass_uid
pass_gid
pass_sth

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

strpmsg

```
struct strpmsg {
    strbuf ctlbuf;
    strbuf databuf;
    SInt32 band;
    long flags;
};
typedef struct strpmsg strpmsg;
```

Fields
ctlbuf
databuf
band
flags

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

strrecvfd

```
struct strrecvfd {
    long fd;
    unsigned short uid;
    unsigned short gid;
    char fill[8];
};
typedef struct strrecvfd strrecvfd;
```

Fields

`fd`

`uid`

`gid`

`fill`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

T_addr_ack

```
struct T_addr_ack {
    long PRIM_type;
    long LOCADDR_length;
    long LOCADDR_offset;
    long REMADDR_length;
    long REMADDR_offset;
};
typedef struct T_addr_ack T_addr_ack;
```

Fields

`PRIM_type`

`LOCADDR_length`

`LOCADDR_offset`

`REMADDR_length`

`REMADDR_offset`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

T_addr_req

```
struct T_addr_req {
    long PRIM_type;
};
typedef struct T_addr_req T_addr_req;
```

Fields
`PRIM_type`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

T_bind_ack

```
struct T_bind_ack {
    long PRIM_type;
    long ADDR_length;
    long ADDR_offset;
    unsigned long CONIND_number;
};
typedef struct T_bind_ack T_bind_ack;
```

Fields
`PRIM_type`
`ADDR_length`
`ADDR_offset`
`CONIND_number`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

T_bind_req

```
struct T_bind_req {
    long PRIM_type;
    long ADDR_length;
    long ADDR_offset;
    unsigned long CONIND_number;
};
typedef struct T_bind_req T_bind_req;
```

Fields

`PRIM_type`

`ADDR_length`

`ADDR_offset`

`CONIND_number`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

t_call

Fields

T_cancelreply_req

```
struct T_cancelreply_req {
    long PRIM_type;
    long SEQ_number;
};
typedef struct T_cancelreply_req T_cancelreply_req;
```

Fields

`PRIM_type`

`SEQ_number`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

T_cancelrequest_req

```
struct T_cancelrequest_req {
    long PRIM_type;
    long SEQ_number;
};
typedef struct T_cancelrequest_req T_cancelrequest_req;
```

Fields

`PRIM_type`

`SEQ_number`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

T_conn_con

```
struct T_conn_con {
    long PRIM_type;
    long RES_length;
    long RES_offset;
    long OPT_length;
    long OPT_offset;
};
typedef struct T_conn_con T_conn_con;
```

Fields

`PRIM_type`

`RES_length`

`RES_offset`

`OPT_length`

`OPT_offset`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

T_conn_ind

```
struct T_conn_ind {
    long PRIM_type;
    long SRC_length;
    long SRC_offset;
    long OPT_length;
    long OPT_offset;
    long SEQ_number;
};
typedef struct T_conn_ind T_conn_ind;
```

Fields

`PRIM_type`

`SRC_length`

`SRC_offset`

`OPT_length`

`OPT_offset`

`SEQ_number`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

T_conn_req

```
struct T_conn_req {
    long PRIM_type;
    long DEST_length;
    long DEST_offset;
    long OPT_length;
    long OPT_offset;
};
typedef struct T_conn_req T_conn_req;
```

Fields

`PRIM_type`

`DEST_length`

`DEST_offset`

`OPT_length`

`OPT_offset`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

T_conn_res

```
struct T_conn_res {
    long PRIM_type;
    queue_t * QUEUE_ptr;
    long OPT_length;
    long OPT_offset;
    long SEQ_number;
};
typedef struct T_conn_res T_conn_res;
```

Fields

`PRIM_type`

`QUEUE_ptr`

`OPT_length`

`OPT_offset`

`SEQ_number`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

T_data_ind

```
struct T_data_ind {
    long PRIM_type;
    long MORE_flag;
};
typedef struct T_data_ind T_data_ind;
```

Fields

`PRIM_type`

`MORE_flag`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

T_data_req

```
struct T_data_req {
    long PRIM_type;
    long MORE_flag;
};
typedef struct T_data_req T_data_req;
```

Fields

`PRIM_type`

`MORE_flag`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

T_delname_req

```
struct T_delname_req {
    long PRIM_type;
    long SEQ_number;
    long NAME_length;
    long NAME_offset;
};
typedef struct T_delname_req T_delname_req;
```

Fields

`PRIM_type`

`SEQ_number`

`NAME_length`

`NAME_offset`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

t_discon

Fields

T_discon_ind

```
struct T_discon_ind {
    long PRIM_type;
    long DISCON_reason;
    long SEQ_number;
};
typedef struct T_discon_ind T_discon_ind;
```

Fields

`PRIM_type`

`DISCON_reason`

`SEQ_number`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

T_discon_req

```
struct T_discon_req {
    long PRIM_type;
    long SEQ_number;
};
typedef struct T_discon_req T_discon_req;
```

Fields

`PRIM_type`

`SEQ_number`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

T_error_ack

```
struct T_error_ack {
    long PRIM_type;
    long ERROR_prim;
    long TLI_error;
    long UNIX_error;
};
typedef struct T_error_ack T_error_ack;
```

Fields

`PRIM_type`

`ERROR_prim`

`TLI_error`

`UNIX_error`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

T_event_ind

```
struct T_event_ind {
    long PRIM_type;
    long EVENT_code;
    long EVENT_cookie;
};
typedef struct T_event_ind T_event_ind;
```

Fields

`PRIM_type`

`EVENT_code`

`EVENT_cookie`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

T_exdata_ind

```
struct T_exdata_ind {
    long PRIM_type;
    long MORE_flag;
};
typedef struct T_exdata_ind T_exdata_ind;
```

Fields

`PRIM_type`

`MORE_flag`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

T_exdata_req

```
struct T_exdata_req {
    long PRIM_type;
    long MORE_flag;
};
typedef struct T_exdata_req T_exdata_req;
```

Fields

`PRIM_type`

`MORE_flag`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

t_info

Fields

T_info_ack

```
struct T_info_ack {
    long PRIM_type;
    long TSDU_size;
    long ETSDU_size;
    long CDATA_size;
    long DDATA_size;
    long ADDR_size;
    long OPT_size;
    long TIDU_size;
    long SERV_type;
    long CURRENT_state;
    long PROVIDER_flag;
};
typedef struct T_info_ack T_info_ack;
```

Fields

PRIM_type

TSDU_size

ETSDU_size

CDATA_size

DDATA_size

ADDR_size

OPT_size

TIDU_size

SERV_type

CURRENT_state

PROVIDER_flag

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

T_info_req

```
struct T_info_req {
    long PRIM_type;
};
typedef struct T_info_req T_info_req;
```

Fields

PRIM_type

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

The Keepalive Structure

Specifies the value of the `OPT_KEEPALIVE` option.

```
struct t_kpalive {
    SInt32 kp_onoff;
    SInt32 kp_timeout;
};
typedef struct t_kpalive t_kpalive;
```

Fields

`kp_onoff`

A constant specifying whether the option is turned on (T_ON) or off (T_OFF).

`kp_timeout`

A positive integer specifying how many minutes Open Transport should maintain a connection in the absence of traffic.

Discussion

The keepalive structure specifies the value of the OPT_KEEPALIVE option, described in "XTI-Level Options and Generic Options" (page 350)

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransport.h`

The Linger Structure

Specifies the value of the `XTI_LINGER` option.

```
struct t_linger {
    SInt32 l_onoff;
    SInt32 l_linger;
};
typedef struct t_linger t_linger;
```

Fields

`l_onoff`

A constant specifying whether the option is turned on (T_ON) or off (T_OFF).

`l_linger`

An integer specifying the linger time, the amount of time in seconds that Open Transport should wait to allow data in an endpoint's internal buffer to be sent before the OTCloseProvider function closes the endpoint.

To request the default value for this option, set the l_linger field to T_UNSPEC.

Discussion

The linger structure specifies the value of the `XTI_LINGER` option, described in "XTI-Level Options and Generic Options" (page 350).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransport.h`

T_lkupname_con

```
struct T_lkupname_con {
    long PRIM_type;
    long SEQ_number;
    long NAME_length;
    long NAME_offset;
    long RSP_count;
    long RSP_cumcount;
};
typedef struct T_lkupname_con T_lkupname_con;
```

Fields

PRIM_type

SEQ_number

NAME_length

NAME_offset

RSP_count

RSP_cumcount

Availability

Available in Mac OS X v10.0 and later.

Declared In

OpenTransportProtocol.h

T_lkupname_req

```
struct T_lkupname_req {
    long PRIM_type;
    long SEQ_number;
    long NAME_length;
    long NAME_offset;
    long ADDR_length;
    long ADDR_offset;
    long MAX_number;
    long MAX_milliseconds;
    long REQ_flags;
};
typedef struct T_lkupname_req T_lkupname_req;
```

Fields

PRIM_type

SEQ_number

NAME_length

NAME_offset

ADDR_length

ADDR_offset

MAX_number

MAX_milliseconds

REQ_flags

Availability

Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

T_MIB_ack

Fields

T_MIB_req

Fields

T_ok_ack

```
struct T_ok_ack {
    long PRIM_type;
    long CORRECT_prim;
};
typedef struct T_ok_ack T_ok_ack;
```

Fields
PRIM_type
CORRECT_prim

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

t_opthdr

Fields

T_optmgmt_ack

```
struct T_optmgmt_ack {
    long PRIM_type;
    long OPT_length;
    long OPT_offset;
    long MGMT_flags;
};
typedef struct T_optmgmt_ack T_optmgmt_ack;
```

Fields

`PRIM_type`

`OPT_length`

`OPT_offset`

`MGMT_flags`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

T_optmgmt_req

```
struct T_optmgmt_req {
    long PRIM_type;
    long OPT_length;
    long OPT_offset;
    long MGMT_flags;
};
typedef struct T_optmgmt_req T_optmgmt_req;
```

Fields

`PRIM_type`

`OPT_length`

`OPT_offset`

`MGMT_flags`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

T_ordrel_ind

```
struct T_ordrel_ind {
    long PRIM_type;
};
typedef struct T_ordrel_ind T_ordrel_ind;
```

Fields
`PRIM_type`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

T_ordrel_req

```
struct T_ordrel_req {
    long PRIM_type;
};
typedef struct T_ordrel_req T_ordrel_req;
```

Fields
`PRIM_type`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

T_primitives

```
union T_primitives {
    long primType;
    T_addr_ack taddrack;
    T_bind_ack tbindack;
    T_bind_req tbindreq;
    T_conn_con tconncon;
    T_conn_ind tconnind;
    T_conn_req tconnreq;
    T_conn_res tconnres;
    T_data_ind tdataind;
    T_data_req tdatareq;
    T_discon_ind tdisconind;
    T_discon_req tdisconreq;
    T_exdata_ind texdataind;
    T_exdata_req texdatareq;
    T_error_ack terrorack;
    T_info_ack tinfoack;
    T_info_req tinforeq;
    T_ok_ack tokack;
    T_optmgmt_ack toptmgmtack;
    T_optmgmt_req toptmgmtreq;
    T_ordrel_ind tordrelind;
    T_ordrel_req tordrelreq;
    T_unbind_req tunbindreq;
    T_uderror_ind tuderrorind;
    T_unitdata_ind tunitdataind;
    T_unitdata_req tunitdatareq;
    T_unitreply_ind tunitreplyind;
    T_unitrequest_ind tunitrequestind;
    T_unitrequest_req tunitrequestreq;
    T_unitreply_req tunitreplyreq;
    T_unitreply_ack tunitreplyack;
    T_reply_ind treplyind;
    T_request_ind trequestind;
    T_request_req trequestreq;
    T_reply_req treplyreq;
    T_reply_ack treplyack;
    T_cancelrequest_req tcancelreqreq;
    T_resolveaddr_req tresolvereq;
    T_resolveaddr_ack tresolveack;
    T_regname_req tregnamereq;
    T_regname_ack tregnameack;
    T_delname_req tdelnamereq;
    T_lkupname_req tlkupnamereq;
    T_lkupname_con tlkupnamecon;
    T_sequence_ack tsequenceack;
    T_event_ind teventind;
};
typedef union T_primitives T_primitives;
```

Fields

`primType`

`taddrack`

`tbindack`

`tbindreq`

```
tconncon
tconnind
tconnreq
tconnres
tdataind
tdatareq
tdisconind
tdisconreq
texdataind
texdatareq
terrorack
tinfoack
tinforeq
tokack
toptmgmtack
toptmgmtreq
tordrelind
tordrelreq
tunbindreq
tuderrorind
tunitdataind
tunitdatareq
tunitreplyind
tunitrequestind
tunitrequestreq
tunitreplyreq
tunitreplyack
treplyind
trequestind
trequestreq
treplyreq
treplyack
tcancelreqreq
tresolvereq
tresolveack
tregnamereq
tregnameack
tdelnamereq
tlkupnamereq
tlkupnamecon
tsequenceack
teventind
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

T_regname_ack

```
struct T_regname_ack {
    long PRIM_type;
    long SEQ_number;
    long REG_id;
    long ADDR_length;
    long ADDR_offset;
};
typedef struct T_regname_ack T_regname_ack;
```

Fields

PRIM_type

SEQ_number

REG_id

ADDR_length

ADDR_offset

Availability

Available in Mac OS X v10.0 and later.

Declared In

OpenTransportProtocol.h

T_regname_req

```
struct T_regname_req {
    long PRIM_type;
    long SEQ_number;
    long NAME_length;
    long NAME_offset;
    long ADDR_length;
    long ADDR_offset;
    long REQ_flags;
};
typedef struct T_regname_req T_regname_req;
```

Fields

PRIM_type

SEQ_number

NAME_length

NAME_offset

ADDR_length

ADDR_offset

REQ_flags

Availability

Available in Mac OS X v10.0 and later.

Declared In

OpenTransportProtocol.h

t_reply

Fields

T_reply_ack

```
struct T_reply_ack {
    long PRIM_type;
    long SEQ_number;
    long TLI_error;
    long UNIX_error;
};
typedef struct T_reply_ack T_reply_ack;
```

Fields

`PRIM_type`

`SEQ_number`

`TLI_error`

`UNIX_error`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

T_reply_ind

```
struct T_reply_ind {
    long PRIM_type;
    long SEQ_number;
    long OPT_length;
    long OPT_offset;
    long REP_flags;
    long TLI_error;
    long UNIX_error;
};
typedef struct T_reply_ind T_reply_ind;
```

Fields

`PRIM_type`

`SEQ_number`

`OPT_length`

`OPT_offset`

`REP_flags`

`TLI_error`

`UNIX_error`

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

T_reply_req

```
struct T_reply_req {
    long PRIM_type;
    long SEQ_number;
    long OPT_length;
    long OPT_offset;
    long REP_flags;
};
typedef struct T_reply_req T_reply_req;
```

Fields
PRIM_type
SEQ_number
OPT_length
OPT_offset
REP_flags

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

t_request

Fields

T_request_ind

```
struct T_request_ind {
    long PRIM_type;
    long SEQ_number;
    long OPT_length;
    long OPT_offset;
    long REQ_flags;
};
typedef struct T_request_ind T_request_ind;
```

Fields

`PRIM_type`

`SEQ_number`

`OPT_length`

`OPT_offset`

`REQ_flags`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

T_request_req

```
struct T_request_req {
    long PRIM_type;
    long SEQ_number;
    long OPT_length;
    long OPT_offset;
    long REQ_flags;
};
typedef struct T_request_req T_request_req;
```

Fields

`PRIM_type`

`SEQ_number`

`OPT_length`

`OPT_offset`

`REQ_flags`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

T_resolveaddr_ack

```
struct T_resolveaddr_ack {
    long PRIM_type;
    long SEQ_number;
    long ADDR_length;
    long ADDR_offset;
    long ORIG_client;
    long ORIG_data;
    long TLI_error;
    long UNIX_error;
};
typedef struct T_resolveaddr_ack T_resolveaddr_ack;
```

Fields

PRIM_type

SEQ_number

ADDR_length

ADDR_offset

ORIG_client

ORIG_data

TLI_error

UNIX_error

Availability

Available in Mac OS X v10.0 and later.

Declared In

OpenTransportProtocol.h

T_resolveaddr_req

```
struct T_resolveaddr_req {
    long PRIM_type;
    long SEQ_number;
    long ADDR_length;
    long ADDR_offset;
    long ORIG_client;
    long ORIG_data;
    long MAX_milliseconds;
};
typedef struct T_resolveaddr_req T_resolveaddr_req;
```

Fields

PRIM_type

SEQ_number

ADDR_length

ADDR_offset

ORIG_client

ORIG_data

MAX_milliseconds

Availability

Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

T_sequence_ack

```
struct T_sequence_ack {
    long PRIM_type;
    long ORIG_prim;
    long SEQ_number;
    long TLI_error;
    long UNIX_error;
};
typedef struct T_sequence_ack T_sequence_ack;
```

Fields
PRIM_type
ORIG_prim
SEQ_number
TLI_error
UNIX_error

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

T_stream_timer

Fields

T_stream_timer_1

Fields

t_uderr

Fields

T_uderror_ind

```
struct T_uderror_ind {
    long PRIM_type;
    long DEST_length;
    long DEST_offset;
    long OPT_length;
    long OPT_offset;
    long ERROR_type;
};
typedef struct T_uderror_ind T_uderror_ind;
```

Fields

`PRIM_type`

`DEST_length`

`DEST_offset`

`OPT_length`

`OPT_offset`

`ERROR_type`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

T_unbind_req

```
struct T_unbind_req {
    long PRIM_type;
};
typedef struct T_unbind_req T_unbind_req;
```

Fields

`PRIM_type`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

t_unitdata

Fields

T_unitdata_ind

```
struct T_unitdata_ind {
    long PRIM_type;
    long SRC_length;
    long SRC_offset;
    long OPT_length;
    long OPT_offset;
};
typedef struct T_unitdata_ind T_unitdata_ind;
```

Fields

`PRIM_type`

`SRC_length`

`SRC_offset`

`OPT_length`

`OPT_offset`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

T_unitdata_req

```
struct T_unitdata_req {
    long PRIM_type;
    long DEST_length;
    long DEST_offset;
    long OPT_length;
    long OPT_offset;
};
typedef struct T_unitdata_req T_unitdata_req;
```

Fields

`PRIM_type`

`DEST_length`

`DEST_offset`

`OPT_length`

`OPT_offset`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

t_unitreply

Fields

T_unitreply_ack

```
struct T_unitreply_ack {
    long PRIM_type;
    long SEQ_number;
    long TLI_error;
    long UNIX_error;
};
typedef struct T_unitreply_ack T_unitreply_ack;
```

Fields

`PRIM_type`

`SEQ_number`

`TLI_error`

`UNIX_error`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

T_unitreply_ind

```
struct T_unitreply_ind {
    long PRIM_type;
    long SEQ_number;
    long OPT_length;
    long OPT_offset;
    long REP_flags;
    long TLI_error;
    long UNIX_error;
};
typedef struct T_unitreply_ind T_unitreply_ind;
```

Fields

`PRIM_type`

`SEQ_number`

`OPT_length`

`OPT_offset`

`REP_flags`

`TLI_error`

`UNIX_error`

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

T_unitreply_req

```
struct T_unitreply_req {
    long PRIM_type;
    long SEQ_number;
    long OPT_length;
    long OPT_offset;
    long REP_flags;
};
typedef struct T_unitreply_req T_unitreply_req;
```

Fields
PRIM_type
SEQ_number
OPT_length
OPT_offset
REP_flags

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

t_unitrequest

Fields

T_unitrequest_ind

```
struct T_unitrequest_ind {
    long PRIM_type;
    long SEQ_number;
    long SRC_length;
    long SRC_offset;
    long OPT_length;
    long OPT_offset;
    long REQ_flags;
};
typedef struct T_unitrequest_ind T_unitrequest_ind;
```

Fields

`PRIM_type`

`SEQ_number`

`SRC_length`

`SRC_offset`

`OPT_length`

`OPT_offset`

`REQ_flags`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

T_unitrequest_req

```
struct T_unitrequest_req {
    long PRIM_type;
    long SEQ_number;
    long DEST_length;
    long DEST_offset;
    long OPT_length;
    long OPT_offset;
    long REQ_flags;
};
typedef struct T_unitrequest_req T_unitrequest_req;
```

Fields

`PRIM_type`

`SEQ_number`

`DEST_length`

`DEST_offset`

`OPT_length`

`OPT_offset`

`REQ_flags`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProtocol.h`

T8022Address

```
struct T8022Address {
    OTAddressType fAddrFamily;
    UInt8 fHWAddr[6];
    UInt16 fSAP;
    UInt8 fSNAP[5];
};
typedef struct T8022Address T8022Address;
```

Fields

`fAddrFamily`

`fHWAddr`

`fSAP`

`fSNAP`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProviders.h`

T8022FullPacketHeader

```
struct T8022FullPacketHeader {
    EnetPacketHeader fEnetPart;
    T8022SNAPHeader f8022Part;
};
typedef struct T8022FullPacketHeader T8022FullPacketHeader;
```

Fields

`fEnetPart`

`f8022Part`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProviders.h`

T8022Header

```
struct T8022Header {
    UInt8 fDSAP;
    UInt8 fSSAP;
    UInt8 fCtrl;
};
typedef struct T8022Header T8022Header;
```

Fields

`fDSAP`

`fSSAP`

`fCtrl`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransportProviders.h`

T8022SNAPHeader

```
struct T8022SNAPHeader {
    UInt8 fDSAP;
    UInt8 fSSAP;
    UInt8 fCtrl;
    UInt8 fSNAP[5];
};
typedef struct T8022SNAPHeader T8022SNAPHeader;
```

Fields

`fDSAP`

`fSSAP`

`fCtrl`

`fSNAP`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProviders.h`

TBind

Describes the protocol address to which an endpoint is currently bound or connected, or specifies the protocol address to which you wish to bind or connect the endpoint.

```
struct TBind {
    TNetbuf addr;
    OTQLen qlen;
};
typedef struct TBind TBind;
```

Fields

`addr`

A `TNetbuf` structure that contains information about an address. The `addr.maxlen` field specifies the maximum size of the address, the `addr.len` field specifies the actual length of the address, and the `addr.buf` field points to the buffer containing the address.

When specifying an address, you must allocate a buffer for the address and initialize it; you must set the `addr.buf` field to point to this buffer; and you must set the `addr.len` field to the size of the address.

When requesting an address, you must allocate a buffer in which the address is to be placed; you must set the `addr.buf` field to point to this buffer; and you must set the `addr.maxlen` field to the maximum size of the address that is being returned. You determine this value by examining the `addr` field of the `TEndpointInfo` (page 174) structure for the endpoint.

`qlen`

For a connection-oriented endpoint, the maximum number of connection requests that can be concurrently outstanding for this endpoint. For more information, see the description of the `OTBind` (page 378) function. For connectionless endpoints, this field has no meaning.

Discussion
The `TBind` structure describes the protocol address to which an endpoint is currently bound or connected, or specifies the protocol address to which you wish to bind or connect the endpoint. For a connection-oriented endpoint, the `TBind` structure also specifies the actual or desired number of connection requests that can be concurrently outstanding for the endpoint.

You pass the `TBind` structure as a parameter to the `OTBind` (page 378) function, the `OTGetProtAddress` (page 407) function, and the `OTResolveAddress` (page 446) function.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

TCall

Specifies the options and data associated with establishing a connection.

```
struct TCall {
    TNetbuf addr;
    TNetbuf opt;
    TNetbuf udata;
    OTSequence sequence;
};
typedef struct TCall TCall;
```

Fields

`addr`
A TNetbuf structure that specifies the location and size of an address buffer

`opt`
A TNetbuf structure that specifies the location and size of an options buffer.

`udata`
A TNetbuf structure that specifies the location and size of a buffer for data associated with a connection or disconnection request.

`sequence`
A 32-bit value used by the OTListen and OTAccept functions to specify the connection ID.

Discussion
You use the `TCall` structure to specify the options and data associated with establishing a connection. You pass a pointer to this structure as a parameter to the OTConnect function, the OTRcvConnect function, the OTListen function, and the OTAccept function.

If you are using the TCall structure to send information, you must allocate a buffer and initialize it to contain the information. Set the .buf field of each TNetbuf to point to the buffer, and then specify the size of the buffer using the .len field. Set this field to 0 if you are not sending data.

If you are using the TCall structure to receive information, you must allocate a buffer into which the function can place the information when it returns. Then set the .buf field of all the TNetbufs to point to this buffer, and set the .maxlen field to the maximum size of the information. Set this field to 0 if you are not interested in receiving information.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

TDiscon

Specifies data sent with the OTSndDisconnect function and retrieved by the OTRcvDisconnect function.

```
struct TDiscon {
    TNetbuf udata;
    OTReason reason;
    OTSequence sequence;
};
typedef struct TDiscon TDiscon;
```

Fields

`udata`
: A TNetbuf structure that references data sent with the OTSndDisconnect function or received by the OTRcvDisconnect function.

`reason`
: A 32-bit value specifying an error code that identifies the reason for the disconnection. These codes are supplied by the protocol. For additional information, consult the documentation provided for the protocol you are using.

`sequence`
: A 32-bit value specifying an outstanding connection request that has been rejected. This field is meaningful only when you have issued several connection requests to the same endpoint and are awaiting the results.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

TEndpointInfo

Describes the initial characteristics of an endpoint that you opened by calling the `OTOpenEndpointInContext` function; returned by calling `OTGetEndpointInfo`.

```
struct TEndpointInfo {
    OTDataSize addr;
    OTDataSize options;
    OTDataSize tsdu;
    OTDataSize etsdu;
    OTDataSize connect;
    OTDataSize discon;
    OTServiceType servtype;
    UInt32 flags;
};
typedef struct TEndpointInfo TEndpointInfo;
```

Fields

`addr`

`options`

A value greater than or equal to 0 indicates the maximum number of bytes needed to store the protocol-specific options that this endpoint supports, if any. A value of `T_INVALID` (-2) indicates that this endpoint has no protocol-specific options that you can set; they are read-only. A value of -3 specifies that the provider does not support any options.

`tsdu`

For a transactionless endpoint, a positive value indicates the maximum number of bytes in a transport service data unit (TSDU) for this endpoint. A value of `T_INFINITE` (-1) indicates that there is no limit to the size of a TSDU. A value of 0 indicates that the provider does not support the concept of a TSDU. This means that you cannot send data with logical boundaries preserved across a connection. A value of `T_INVALID` indicates that this endpoint cannot transfer normal data (as opposed to expedited data).

For a transaction-based endpoint, this field indicates the maximum number of bytes in a response.

`etsdu`

For a transactionless endpoint, a positive value indicates the maximum number of bytes in an expedited transport service data unit (ETSDU) for this endpoint. A value of `T_INFINITE` indicates that there is no limit to the size of a ETSDU. A value of 0 indicates that this endpoint does not support the concept of an ETSDU. This means that you must not send expedited data with logical boundaries preserved across a connection. A value of `T_INVALID` indicates that this endpoint cannot transfer expedited data.

For a transaction-based endpoint, this field indicates the maximum number of bytes in a request.

`connect`

For a connection-oriented endpoint, a value greater than or equal to 0 indicates the maximum amount of data (in bytes) that you can send with the `OTConnect` (page 385) function or the `OTAccept` (page 366) function. A value of `T_INVALID` indicates that this endpoint does not let you send data with these functions. This field is meaningless for other types of endpoints.

`discon`

For a connection-oriented endpoint, a value greater than or equal to 0 indicates the maximum amount of data (in bytes) that you can send using the `OTSndDisconnect` (page 458) function. A value of `T_INVALID` indicates that this endpoint does not let you send data with disconnection requests. This field is meaningless for other types of endpoints.

`servtype`

A constant that indicates what kind of service the endpoint provides. Possible values are given by the "Endpoint Service Types" (page 302) enumeration.

`flags`

A bit FIeld that provides additional information about the endpoint. Possible values are given by the "Endpoint Flags" (page 338) enumeration.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

IP Multicast Address Structure

Supports adding and dropping membership in an IP multicast address.

```
struct TIPAddMulticast {
    InetHost multicastGroupAddress;
    InetHost interfaceAddress;
};
typedef struct TIPAddMulticast TIPAddMulticast;
```

Fields

`multicastGroupAddress`
The IP address of the multicast group for which you want to add or drop membership.

`interfaceAddress`
The IP address of the network interface that you are using for the multicast group.

Discussion
You use the IP multicast address structure with the IP_ADD_MEMBERSHIP and IP_DROP_MEMBERSHIP options when you are adding or dropping membership in an IP multicast address.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProviders.h`

TLookupBuffer

Defines the format of entries in the buffer passed back in the reply parameter of the `OTLookupName` function.

```
struct TLookupBuffer {
    UInt16 fAddressLength;
    UInt16 fNameLength;
    UInt8 fAddressBuffer[1];
};
typedef struct TLookupBuffer TLookupBuffer;
```

Fields

`fAddressLength`
Specifies the size of the address specified by the fAddressBuffer field.

`fNameLength`
Specifies the size of the name that is stored in the buffer following the fAddressBuffer field.

`fAddressBuffer`
The first byte of the address to which the entity whose name follows (in the buffer) is bound.

Discussion
The TLookupBuffer structure defines the format of entries in the buffer passed back in the reply parameter of the OTLookupName function. When you parse the buffer in which the OTLookupName function places the names it has found, you can cast it as a TLookupBuffer structure.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

TLookupReply

Stores information passed back to your application by the OTLookupName function.

```
struct TLookupReply {
    TNetbuf names;
    UInt32 rspcount;
};
typedef struct TLookupReply TLookupReply;
```

Fields

`names`

A TNetbuf structure that specifies the size and location of a buffer into which the OTLookupName function, on return, places the names it has found. You must allocate the buffer into which the replies are stored when the function returns; you must set the names.buf field to point to it; and you must set the names.maxlen field to the size of the buffer.

`rspcount`

A long specifying, on return, the number of names found.

Discussion
You use the TLookupReply structure to store information passed back to you by the OTLookupName function. The information includes both a pointer to a buffer (containing registered entity names matching the criterion specified with the TLookupRequest structure) and the number of names found.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

TLookupRequest

Specifies the entity name to be looked up by the `OTLookupName` function.

```
struct TLookupRequest {
    TNetbuf name;
    TNetbuf addr;
    UInt32 maxcnt;
    OTTimeout timeout;
    OTFlags flags;
};
typedef struct TLookupRequest TLookupRequest;
```

Fields

`name`

A TNetbuf structure specifying the location and size of a buffer that contains the name to be looked up. You must allocate a buffer that contains the name, set the name.buf field to point to it, and set the name.len field to the length of the name.

`addr`

A TNetbuf structure describing the address of the node where you expect the names to be stored. You should normally supply 0 for addr.len. This causes the provider to use internal defaults for the starting point of the search. For a protocol family such as AppleTalk, in which every node has access to name and address information, this parameter is meaningless.

Specifying an address has meaning for those protocols that use a dedicated server or other device to store name information. In such a case, the name specified would override the protocol's default address. To specify an address, you would need to allocate a buffer containing the address, set the addr.buf field to point to it, and set the addr.len field to the length of the address. Consult the documentation supplied with your protocol to determine whether you can or should specify an address.

`maxcnt`

A long specifying the number of names you expect to be returned. Some protocols allow the use of wildcard characters in specifying a name. As a result, the OTLookupName function might find multiple names matching the specified name pattern. If you expect a specific number of replies for a particular name or do not expect to exceed a specific number, you should specify this number to obtain faster execution. Otherwise, set this field to 0xffff ffff; in this case, the timeout value will control the lookup.

`timeout`

A long specifying the amount of time, in milliseconds, that should elapse before Open Transport gives up searching for a name. Specify 0 to leave the timeout value up to the underlying naming system.

`flags`

Discussion

You use the TLookupRequest structure to specify the entity name to be looked up by the OTLookupName function and to set additional values that the mapper provider uses to circumscribe the search.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransport.h`

TNetbuf

Specifies the location and size of a buffer that contains an address, option information, or user data.

```
struct TNetbuf {
    ByteCount maxlen;
    ByteCount len;
    UInt8 * buf;
};
typedef struct TNetbuf TNetbuf;
```

Fields

`maxlen`

The size (in bytes) of the buffer to which the `buf` field points. You must set the `maxlen` field before passing a `TNetbuf` structure to a provider function as an output parameter. Open Transport ignores this field if you pass the `TNetbuf` structure as an input parameter.

`len`

The actual length (in bytes) of the information in the buffer to which the `buf` field points. If you are using the `TNetbuf` structure as an input parameter, you must set this field.

If you pass the `TNetbuf` structure as an output parameter, on return the provider function sets this field to the number of bytes the function has actually placed in the buffer referenced by the `buf` field.

`buf`

A pointer to a buffer. You must make sure that the `buf` field points to a valid buffer and that the buffer is large enough to store the information for which it is intended.

Discussion

You use a TNetbuf structure to specify the location and size of a buffer that contains an address, option information, or user data. Provider functions use TNetbuf structures both as input parameters and output parameters. If you use a TNetbuf structure as an input parameter, you specify the location and size of a buffer containing information you want to send. If you use a TNetbuf structure as an output parameter, you specify the location and the maximum size of the buffer used to hold information when the function returns.

You use a TNetbuf structure to describe the location and size of contiguous data. Open Transport allows you to describe noncontiguous data with the OTData structure.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransport.h`

The TOption Structure

Stores information about a single option in a buffer.

```
struct TOption {
    ByteCount len;
    OTXTILevel level;
    OTXTIName name;
    UInt32 status;
    UInt32 value[1];
};
typedef struct TOption TOption;
```

Fields

`len`

The size (in bytes) of the option information, including the header.

`level`
: The protocol for which the option is defined.

`name`
: The name of the option.

`status`
: A status code specifying whether the negotiation has succeeded or failed. Possible values are given by the "Open Transport Flags and Status Codes" (page 334) enumeration

`value`
: The option value. To have the endpoint select an appropriate value, you can specify the constant T_UNSPEC.

Discussion
The TOption structure stores information about a single option in a buffer. All functions that you use to change or verify option values use a buffer containing TOption structures to store option information. For each option in the buffer, the TOption structure specifies the total length occupied by the option, the protocol level of the option, the option name, the status of a negotiated value, and the value of the option.

You use the TOption structure with the OPT_NEXTHDR macro, the OTCreateOptionString function, the OTNextOption function, and the OTFindOption function.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

The TOptionHeader Structure

Stores information about options in a buffer.

```
struct TOptionHeader {
    ByteCount len;
    OTXTILevel level;
    OTXTIName name;
    UInt32 status;
};
typedef struct TOptionHeader TOptionHeader;
```

Fields
`len`
: The size (in bytes) of the option information, including the header.

`level`
: The protocol affected.

`name`
: The option name.

`status`
: The status value. Possible values are given by the "Open Transport Flags and Status Codes" (page 334).

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransport.h

The Option Management Structure

Manages the `req` and `ret` parameters of the `OTOptionManagement` function

```
struct TOptMgmt {
    TNetbuf opt;
    OTFlags flags;
};
typedef struct TOptMgmt TOptMgmt;
```

Fields

`opt`

A TNetbuf structure describing the buffer containing option information. The opt.maxlen field specifies the maximum size of the buffer. The opt.len field specifies the actual size of the buffer, and the opt.buf field contains the address of the buffer.

On input, as part of the req parameter, the buffer contains TOption structures describing the options to be negotiated or verified, or contains the names of options whose default or current values you are interested in. You must allocate this buffer, place in it the structures describing the options of interest, and set the opt.len field to the size of the buffer.

On output, as part of the ret parameter, the buffer contains the actual values of the options you described in the req parameter. You must allocate a buffer to hold the option information when the function returns and set the opt.maxlen field to the maximum length of this buffer. When the function returns, the opt.len field is set to the actual length of the buffer.

`flags`

For the req parameter, the flags field indicates the action to be taken as defined by the action flags enumeration (page 570). For the ret parameter, the flags field indicates the overall success or failure of the operation performed by the OTOptionManagement function, as defined by the "Open Transport Flags and Status Codes" (page 334) enumeration.

Discussion
The option management structure is used for the req and ret parameters of the OTOptionManagement function. The req parameter is used to verify or negotiate option values. The ret parameter returns information about an endpoint's default, current, or negotiated values.

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransport.h

TOTConfiguratorRef

```
typedef struct OpaqueTOTConfiguratorRef * TOTConfiguratorRef;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransportProtocol.h

TPortRecord

```
struct TPortRecord {
    OTLink fLink;
    char *fPortName;
    char *fModuleName;
    char *fResourceInfo;
    char *fSlotID;
    TPortRecord *fAlias;
    ItemCount fNumChildren;
    OTPortRef *fChildPorts;
    UInt32 fPortFlags;
    UInt32 fInfoFlags;
    UInt32 fCapabilities;
    OTPortRef fRef;
    streamtab *fStreamtab;
    void *fContext;
    void *fExtra;
};
```

Fields

trace_ids

```
struct trace_ids {
    short ti_mid;
    short ti_sid;
    char ti_level;
};
typedef struct trace_ids trace_ids;
```

Fields

`ti_mid`

`ti_sid`

`ti_level`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransportProtocol.h`

TRegisterReply

Stores information returned by the `OTRegisterName` function.

```
struct TRegisterReply {
    TNetbuf addr;
    OTNameID nameid;
};
typedef struct TRegisterReply TRegisterReply;
```

Fields

`addr`

A TNetbuf structure that specifies the location and size of a buffer containing the actual address of the entity whose name you have just registered. This information is passed back to you when the OTRegisterName function returns. You must allocate a buffer, set the addr.buf field to point to it, and set the addr.maxlen field to the size of the buffer.

`nameid`

A unique identifier passed to you when the OTRegisterName function returns. You can use this identifier when you call the OTDeleteNameByID function to delete the name.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

TRegisterRequest

Specifies the entity name you want to register using the OTRegisterName function and, optionally, to specify its address.

```
struct TRegisterRequest {
    TNetbuf name;
    TNetbuf addr;
    OTFlags flags;
};
typedef struct TRegisterRequest TRegisterRequest;
```

Fields

`name`

A TNetbuf structure that specifies the location and size of a buffer containing the entity name you want to register. You must allocate a buffer that contains the name, set the name.buf field to point to that buffer, and set the name.len field to the length of the buffer.

`addr`

A TNetbuf structure that specifies the location and size of a buffer containing the address associated with the entity whose name you want to register. You must allocate a buffer that contains the address, set the addr.buf field to point to that buffer, and set the addr.len field to the length of the buffer. The actual address with which the entity is associated is returned in the addr field of the TRegisterReply structure.

You can set the addr.len field to 0, in which case the underlying protocol finds an appropriate address to associate with the newly registered entity name.

`flags`

A field used to control registration. Normally, this field is set to 0 for default registration behavior. See the documentation for the naming service you are using for more information.

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransport.h

TReply

```
struct TReply {
    TNetbuf data;
    TNetbuf opt;
    OTSequence sequence;
};
typedef struct TReply TReply;
```

Fields
data
opt
sequence

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransport.h

TRequest

```
struct TRequest {
    TNetbuf data;
    TNetbuf opt;
    OTSequence sequence;
};
typedef struct TRequest TRequest;
```

Fields
data
opt
sequence

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransport.h

TUDErr

In the event of failure of the `OTSndUData` function, points to information that explains why it failed.function (page 462) has failed.

```
struct TUDErr {
    TNetbuf addr;
    TNetbuf opt;
    SInt32 error;
};
typedef struct TUDErr TUDErr;
```

Fields

`addr`

A `TNetbuf` (page 178) structure that contains information about the destination address of the data sent using the `OTSndUData` (page 459) function. The `OTRcvUData` (page 440) function fills in the buffer referenced by this structure when the function returns. You must allocate a buffer to contain the address, initialize the `addr.buf` field to point to it, and set the `addr.maxlen` field to specify its maximum size. If you are not interested in address information, set `addr.maxlen` to 0.

`opt`

A `TNetbuf` (page 178) structure that contains information about the options associated with the data sent using the `OTSndUData` (page 459)function. The `OTRcvUDErr` (page 441) function fills in the buffer referenced by this structure when the function returns. If you want to know this information, you must allocate a buffer to contain the option data, initialize the `opt.buf` field to point to it, and initialize the `opt.maxlen` field to specify the maximum size of the buffer. If you are not interested in option information, set the `opt.maxlen` field to 0.

`error`

On return, this specifies a protocol-dependent error code for the `OTSndUData` (page 459) function that failed.

Discussion

In the event of failure of the `OTSndUData` (page 459)function, points to information that explains why it failed.You pass this structure as a parameter to the `OTRcvUData` (page 440) function.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`OpenTransport.h`

TUnitData

Describes the data being sent with the `OTSndUData` function and the data being read with the `OTRcvUData` function.function (page 467)

```
struct TUnitData {
    TNetbuf addr;
    TNetbuf opt;
    TNetbuf udata;
};
typedef struct TUnitData TUnitData;
```

Fields

`addr`

A `TNetbuf` structure for address information.

`opt`

A `TNetbuf` structure for option information.

`udata`
A `TNetbuf` structure for data.

Discussion
You use the `TUnitData` structure to describe the data being sent with the `OTSndUData` (page 459) function and the data being read with the `OTRcvUData` (page 440) function; you pass this structure as a parameter to each of these functions. When sending data you must initialize the `buf` and `len` fields of all the `TNetbuf` structures. When receiving data, you must initialize the `buf` and `maxlen` fields of all the `TNetbuf` structures.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

TUnitReply

```
struct TUnitReply {
    TNetbuf opt;
    TNetbuf udata;
    OTSequence sequence;
};
typedef struct TUnitReply TUnitReply;
```

Fields
`opt`
`udata`
`sequence`

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

TUnitRequest

```
struct TUnitRequest {
    TNetbuf addr;
    TNetbuf opt;
    TNetbuf udata;
    OTSequence sequence;
};
typedef struct TUnitRequest TUnitRequest;
```

Fields

`addr`

`opt`

`udata`

`sequence`

Discussion

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

uchar_p

```
typedef UInt8 uchar_p;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`OpenTransport.h`

uid_t

```
typedef UInt32 uid_t;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`types.h`

uint_t

```
typedef  uint_t;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransport.h

ushort_p

```
typedef UInt16 ushort_p;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
OpenTransport.h

Constants

AF_8022

```
enum {
    AF_8022 = 8200
};
```

Constants
AF_8022
Available in Mac OS X v10.0 and later.
Declared in OpenTransportProviders.h.

AF_ATALK_FAMILY

```
enum {
    AF_ATALK_FAMILY = 0x0100,
    AF_ATALK_DDP = 0x0100,
    AF_ATALK_DDPNBP = AF_ATALK_FAMILY + 1,
    AF_ATALK_NBP = AF_ATALK_FAMILY + 2,
    AF_ATALK_MNODE = AF_ATALK_FAMILY + 3
};
```

Constants
AF_ATALK_FAMILY
Available in Mac OS X v10.0 and later.
Declared in OpenTransportProviders.h.

AF_ATALK_DDP
Available in Mac OS X v10.0 and later.
Declared in OpenTransportProviders.h.

AF_ATALK_DDPNBP
Available in Mac OS X v10.0 and later.
Declared in OpenTransportProviders.h.

`AF_ATALK_NBP`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`AF_ATALK_MNODE`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

AF_DNS

```
enum {
    AF_DNS = 42
};
```

Constants
`AF_DNS`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

AF_INET

```
enum {
    AF_INET = 2
};
```

Constants
`AF_INET`

AF_ISDN

```
enum {
    AF_ISDN = 8192
};
```

Constants
`AF_ISDN`

ANYMARK

```
enum {
    ANYMARK = 0x01,
    LASTMARK = 0x02
};
```

Constants
`ANYMARK`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`LASTMARK`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

ATALK_IOC_FULLSELFSEND

```
enum {
    ATALK_IOC_FULLSELFSEND = ((MIOC_ATALK << 8) | 47),
    ADSP_IOC_FORWARDRESET = ((MIOC_ATALK << 8) | 60)
};
```

Constants

`ATALK_IOC_FULLSELFSEND`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`ADSP_IOC_FORWARDRESET`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

ATK_DDP

```
enum {
    ATK_DDP = 'DDP ',
    ATK_AARP = 'AARP',
    ATK_ATP = 'ATP ',
    ATK_ADSP = 'ADSP',
    ATK_ASP = 'ASP ',
    ATK_PAP = 'PAP ',
    ATK_NBP = 'NBP ',
    ATK_ZIP = 'ZIP '
};
```

Constants

`ATK_DDP`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`ATK_AARP`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`ATK_ATP`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`ATK_ADSP`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`ATK_ASP`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`ATK_PAP`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`ATK_NBP`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`ATK_ZIP`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

BPRI_LO

```
enum {
    BPRI_LO = 1,
    BPRI_MED = 2,
    BPRI_HI = 3
};
```

Constants

`BPRI_LO`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`BPRI_MED`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`BPRI_HI`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

CE_CONT

```
enum {
    CE_CONT = 0,
    CE_NOTE = 0,
    CE_WARN = 1,
    CE_PANIC = 2
};
```

Constants

`CE_CONT`

`CE_NOTE`

`CE_WARN`

`CE_PANIC`

CLONEOPEN

```
enum {
    CLONEOPEN = 0x02,
    MODOPEN = 0x01,
    OPENFAIL = -1
};
```

Constants

`CLONEOPEN`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`MODOPEN`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`OPENFAIL`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

COM_ISDN

```
enum {
    COM_ISDN = 'ISDN'
};
```

Constants

`COM_ISDN`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

COM_PPP

```
enum {
    COM_PPP = 'PPPC'
};
```

Constants

`COM_PPP`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

COM_SERIAL

```
enum {
    COM_SERIAL = 'SERL'
};
```

Constants

`COM_SERIAL`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

DDP_OPT_CHECKSUM

```
enum {
    DDP_OPT_CHECKSUM = 0x0600,
    DDP_OPT_SRCADDR = 0x2101,
    ATP_OPT_REPLYCNT = 0x2110,
    ATP_OPT_DATALEN = 0x2111,
    ATP_OPT_RELTIMER = 0x2112,
    ATP_OPT_TRANID = 0x2113,
    PAP_OPT_OPENRETRY = 0x2120
};
```

Constants

`DDP_OPT_CHECKSUM`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`DDP_OPT_SRCADDR`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`ATP_OPT_REPLYCNT`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`ATP_OPT_DATALEN`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`ATP_OPT_RELTIMER`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`ATP_OPT_TRANID`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`PAP_OPT_OPENRETRY`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

DDP_OPT_HOPCOUNT

```
enum {
    DDP_OPT_HOPCOUNT = 0x2100
};
```

Constants

`DDP_OPT_HOPCOUNT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

DL_ACCESS

```
enum {
    DL_ACCESS = 0x02,
    DL_BADADDR = 0x01,
    DL_BADCORR = 0x05,
    DL_BADDATA = 0x06,
    DL_BADPPA = 0x08,
    DL_BADPRIM = 0x09,
    DL_BADQOSPARAM = 0x0A,
    DL_BADQOSTYPE = 0x0B,
    DL_BADSAP = 0x00,
    DL_BADTOKEN = 0x0C,
    DL_BOUND = 0x0D,
    DL_INITFAILED = 0x0E,
    DL_NOADDR = 0x0F,
    DL_NOTINIT = 0x10,
    DL_OUTSTATE = 0x03,
    DL_SYSERR = 0x04,
    DL_UNSUPPORTED = 0x07,
    DL_UNDELIVERABLE = 0x11,
    DL_NOTSUPPORTED = 0x12,
    DL_TOOMANY = 0x13,
    DL_NOTENAB = 0x14,
    DL_BUSY = 0x15,
    DL_NOAUTO = 0x16,
    DL_NOXIDAUTO = 0x17,
    DL_NOTESTAUTO = 0x18,
    DL_XIDAUTO = 0x19,
    DL_TESTAUTO = 0x1A,
    DL_PENDING = 0x1B
};
```

Constants

`DL_ACCESS`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`DL_BADADDR`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`DL_BADCORR`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`DL_BADDATA`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`DL_BADPPA`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`DL_BADPRIM`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`DL_BADQOSPARAM`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_BADQOSTYPE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_BADSAP`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_BADTOKEN`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_BOUND`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_INITFAILED`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_NOADDR`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_NOTINIT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_OUTSTATE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_SYSERR`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_UNSUPPORTED`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_UNDELIVERABLE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_NOTSUPPORTED`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_TOOMANY`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_NOTENAB`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_BUSY`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_NOAUTO`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_NOXIDAUTO`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_NOTESTAUTO`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_XIDAUTO`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_TESTAUTO`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_PENDING`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

DL_AUTO_XID

```
enum {
    DL_AUTO_XID = 0x01,
    DL_AUTO_TEST = 0x02
};
```

Constants

`DL_AUTO_XID`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_AUTO_TEST`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

DL_CMD_MASK

```
enum {
    DL_CMD_MASK = 0x0F,
    DL_CMD_OK = 0x00,
    DL_CMD_RS = 0x01,
    DL_CMD_UE = 0x05,
    DL_CMD_PE = 0x06,
    DL_CMD_IP = 0x07,
    DL_CMD_UN = 0x09,
    DL_CMD_IT = 0x0F,
    DL_RSP_MASK = 0xF0,
    DL_RSP_OK = 0x00,
    DL_RSP_RS = 0x10,
    DL_RSP_NE = 0x30,
    DL_RSP_NR = 0x40,
    DL_RSP_UE = 0x50,
    DL_RSP_IP = 0x70,
    DL_RSP_UN = 0x90,
    DL_RSP_IT = 0xF0
};
```

Constants

`DL_CMD_MASK`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_CMD_OK`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_CMD_RS`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_CMD_UE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_CMD_PE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_CMD_IP`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_CMD_UN`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_CMD_IT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_RSP_MASK`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_RSP_OK`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_RSP_RS`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_RSP_NE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_RSP_NR`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_RSP_UE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_RSP_IP`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_RSP_UN`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_RSP_IT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

DL_CODLS

```
enum {
    DL_CODLS = 0x01,
    DL_CLDLS = 0x02,
    DL_ACLDLS = 0x04
};
```

Constants

`DL_CODLS`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_CLDLS`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_ACLDLS`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

DL_CONREJ_DEST_UNKNOWN

```
enum {
    DL_CONREJ_DEST_UNKNOWN = 0x0800,
    DL_CONREJ_DEST_UNREACH_PERMANENT = 0x0801,
    DL_CONREJ_DEST_UNREACH_TRANSIENT = 0x0802,
    DL_CONREJ_QOS_UNAVAIL_PERMANENT = 0x0803,
    DL_CONREJ_QOS_UNAVAIL_TRANSIENT = 0x0804,
    DL_CONREJ_PERMANENT_COND = 0x0805,
    DL_CONREJ_TRANSIENT_COND = 0x0806,
    DL_DISC_ABNORMAL_CONDITION = 0x0807,
    DL_DISC_NORMAL_CONDITION = 0x0808,
    DL_DISC_PERMANENT_CONDITION = 0x0809,
    DL_DISC_TRANSIENT_CONDITION = 0x080A,
    DL_DISC_UNSPECIFIED = 0x080B
};
```

Constants

`DL_CONREJ_DEST_UNKNOWN`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`DL_CONREJ_DEST_UNREACH_PERMANENT`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`DL_CONREJ_DEST_UNREACH_TRANSIENT`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`DL_CONREJ_QOS_UNAVAIL_PERMANENT`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`DL_CONREJ_QOS_UNAVAIL_TRANSIENT`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`DL_CONREJ_PERMANENT_COND`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`DL_CONREJ_TRANSIENT_COND`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`DL_DISC_ABNORMAL_CONDITION`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`DL_DISC_NORMAL_CONDITION`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_DISC_PERMANENT_CONDITION`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_DISC_TRANSIENT_CONDITION`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_DISC_UNSPECIFIED`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

DL_CSMACD

```
enum {
    DL_CSMACD = 0x00,
    DL_TPB = 0x01,
    DL_TPR = 0x02,
    DL_METRO = 0x03,
    DL_ETHER = 0x04,
    DL_HDLC = 0x05,
    DL_CHAR = 0x06,
    DL_CTCA = 0x07,
    DL_FDDI = 0x08,
    DL_OTHER = 0x09
};
```

Constants

`DL_CSMACD`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_TPB`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_TPR`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_METRO`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_ETHER`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_HDLC`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_CHAR`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_CTCA`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_FDDI`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_OTHER`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

DL_CURRENT_VERSION

```
enum {
    DL_CURRENT_VERSION = 0x02,
    DL_VERSION_2 = 0x02
};
```

Constants

`DL_CURRENT_VERSION`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_VERSION_2`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

DL_FACT_PHYS_ADDR

```
enum {
    DL_FACT_PHYS_ADDR = 0x01,
    DL_CURR_PHYS_ADDR = 0x02
};
```

Constants

`DL_FACT_PHYS_ADDR`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_CURR_PHYS_ADDR`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

DL_INFO_REQ

```
enum {
    DL_INFO_REQ = 0x00,
    DL_INFO_ACK = 0x03,
    DL_ATTACH_REQ = 0x0B,
    DL_DETACH_REQ = 0x0C,
    DL_BIND_REQ = 0x01,
    DL_BIND_ACK = 0x04,
    DL_UNBIND_REQ = 0x02,
    DL_OK_ACK = 0x06,
    DL_ERROR_ACK = 0x05,
    DL_SUBS_BIND_REQ = 0x1B,
    DL_SUBS_BIND_ACK = 0x1C,
    DL_SUBS_UNBIND_REQ = 0x15,
    DL_ENABMULTI_REQ = 0x1D,
    DL_DISABMULTI_REQ = 0x1E,
    DL_PROMISCON_REQ = 0x1F,
    DL_PROMISCOFF_REQ = 0x20,
    DL_UNITDATA_REQ = 0x07,
    DL_UNITDATA_IND = 0x08,
    DL_UDERROR_IND = 0x09,
    DL_UDQOS_REQ = 0x0A,
    DL_CONNECT_REQ = 0x0D,
    DL_CONNECT_IND = 0x0E,
    DL_CONNECT_RES = 0x0F,
    DL_CONNECT_CON = 0x10,
    DL_TOKEN_REQ = 0x11,
    DL_TOKEN_ACK = 0x12,
    DL_DISCONNECT_REQ = 0x13,
    DL_DISCONNECT_IND = 0x14,
    DL_RESET_REQ = 0x17,
    DL_RESET_IND = 0x18,
    DL_RESET_RES = 0x19,
    DL_RESET_CON = 0x1A,
    DL_DATA_ACK_REQ = 0x21,
    DL_DATA_ACK_IND = 0x22,
    DL_DATA_ACK_STATUS_IND = 0x23,
    DL_REPLY_REQ = 0x24,
    DL_REPLY_IND = 0x25,
    DL_REPLY_STATUS_IND = 0x26,
    DL_REPLY_UPDATE_REQ = 0x27,
    DL_REPLY_UPDATE_STATUS_IND = 0x28,
    DL_XID_REQ = 0x29,
    DL_XID_IND = 0x2A,
    DL_XID_RES = 0x2B,
    DL_XID_CON = 0x2C,
    DL_TEST_REQ = 0x2D,
    DL_TEST_IND = 0x2E,
    DL_TEST_RES = 0x2F,
    DL_TEST_CON = 0x30,
    DL_PHYS_ADDR_REQ = 0x31,
    DL_PHYS_ADDR_ACK = 0x32,
    DL_SET_PHYS_ADDR_REQ = 0x33,
    DL_GET_STATISTICS_REQ = 0x34,
    DL_GET_STATISTICS_ACK = 0x35
};
```

Constants

`DL_INFO_REQ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_INFO_ACK`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_ATTACH_REQ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_DETACH_REQ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_BIND_REQ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_BIND_ACK`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_UNBIND_REQ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_OK_ACK`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_ERROR_ACK`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_SUBS_BIND_REQ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_SUBS_BIND_ACK`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_SUBS_UNBIND_REQ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_ENABMULTI_REQ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_DISABMULTI_REQ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_PROMISCON_REQ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_PROMISCOFF_REQ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_UNITDATA_REQ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_UNITDATA_IND`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_UDERROR_IND`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_UDQOS_REQ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_CONNECT_REQ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_CONNECT_IND`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_CONNECT_RES`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_CONNECT_CON`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_TOKEN_REQ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_TOKEN_ACK`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_DISCONNECT_REQ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_DISCONNECT_IND`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_RESET_REQ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_RESET_IND`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_RESET_RES`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_RESET_CON`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_DATA_ACK_REQ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_DATA_ACK_IND`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_DATA_ACK_STATUS_IND`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_REPLY_REQ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_REPLY_IND`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_REPLY_STATUS_IND`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_REPLY_UPDATE_REQ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_REPLY_UPDATE_STATUS_IND`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_XID_REQ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_XID_IND`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_XID_RES`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_XID_CON`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_TEST_REQ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_TEST_IND`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_TEST_RES`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_TEST_CON`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_PHYS_ADDR_REQ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_PHYS_ADDR_ACK`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_SET_PHYS_ADDR_REQ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_GET_STATISTICS_REQ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_GET_STATISTICS_ACK`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

DL_INFO_REQ_SIZE

```
enum {
    DL_INFO_REQ_SIZE = sizeof(dl_info_req_t),
    DL_INFO_ACK_SIZE = sizeof(dl_info_ack_t),
    DL_ATTACH_REQ_SIZE = sizeof(dl_attach_req_t),
    DL_DETACH_REQ_SIZE = sizeof(dl_detach_req_t),
    DL_BIND_REQ_SIZE = sizeof(dl_bind_req_t),
    DL_BIND_ACK_SIZE = sizeof(dl_bind_ack_t),
    DL_UNBIND_REQ_SIZE = sizeof(dl_unbind_req_t),
    DL_SUBS_BIND_REQ_SIZE = sizeof(dl_subs_bind_req_t),
    DL_SUBS_BIND_ACK_SIZE = sizeof(dl_subs_bind_ack_t),
    DL_SUBS_UNBIND_REQ_SIZE = sizeof(dl_subs_unbind_req_t),
    DL_OK_ACK_SIZE = sizeof(dl_ok_ack_t),
    DL_ERROR_ACK_SIZE = sizeof(dl_error_ack_t),
    DL_CONNECT_REQ_SIZE = sizeof(dl_connect_req_t),
    DL_CONNECT_IND_SIZE = sizeof(dl_connect_ind_t),
    DL_CONNECT_RES_SIZE = sizeof(dl_connect_res_t),
    DL_CONNECT_CON_SIZE = sizeof(dl_connect_con_t),
    DL_TOKEN_REQ_SIZE = sizeof(dl_token_req_t),
    DL_TOKEN_ACK_SIZE = sizeof(dl_token_ack_t),
    DL_DISCONNECT_REQ_SIZE = sizeof(dl_disconnect_req_t),
    DL_DISCONNECT_IND_SIZE = sizeof(dl_disconnect_ind_t),
    DL_RESET_REQ_SIZE = sizeof(dl_reset_req_t),
    DL_RESET_IND_SIZE = sizeof(dl_reset_ind_t),
    DL_RESET_RES_SIZE = sizeof(dl_reset_res_t),
    DL_RESET_CON_SIZE = sizeof(dl_reset_con_t),
    DL_UNITDATA_REQ_SIZE = sizeof(dl_unitdata_req_t),
    DL_UNITDATA_IND_SIZE = sizeof(dl_unitdata_ind_t),
    DL_UDERROR_IND_SIZE = sizeof(dl_uderror_ind_t),
    DL_UDQOS_REQ_SIZE = sizeof(dl_udqos_req_t),
    DL_ENABMULTI_REQ_SIZE = sizeof(dl_enabmulti_req_t),
    DL_DISABMULTI_REQ_SIZE = sizeof(dl_disabmulti_req_t),
    DL_PROMISCON_REQ_SIZE = sizeof(dl_promiscon_req_t),
    DL_PROMISCOFF_REQ_SIZE = sizeof(dl_promiscoff_req_t),
    DL_PHYS_ADDR_REQ_SIZE = sizeof(dl_phys_addr_req_t),
    DL_PHYS_ADDR_ACK_SIZE = sizeof(dl_phys_addr_ack_t),
    DL_SET_PHYS_ADDR_REQ_SIZE = sizeof(dl_set_phys_addr_req_t),
    DL_GET_STATISTICS_REQ_SIZE = sizeof(dl_get_statistics_req_t),
    DL_GET_STATISTICS_ACK_SIZE = sizeof(dl_get_statistics_ack_t),
    DL_XID_REQ_SIZE = sizeof(dl_xid_req_t),
    DL_XID_IND_SIZE = sizeof(dl_xid_ind_t),
    DL_XID_RES_SIZE = sizeof(dl_xid_res_t),
    DL_XID_CON_SIZE = sizeof(dl_xid_con_t),
    DL_TEST_REQ_SIZE = sizeof(dl_test_req_t),
    DL_TEST_IND_SIZE = sizeof(dl_test_ind_t),
    DL_TEST_RES_SIZE = sizeof(dl_test_res_t),
    DL_TEST_CON_SIZE = sizeof(dl_test_con_t),
    DL_DATA_ACK_REQ_SIZE = sizeof(dl_data_ack_req_t),
    DL_DATA_ACK_IND_SIZE = sizeof(dl_data_ack_ind_t),
    DL_DATA_ACK_STATUS_IND_SIZE = sizeof(dl_data_ack_status_ind_t),
    DL_REPLY_REQ_SIZE = sizeof(dl_reply_req_t),
    DL_REPLY_IND_SIZE = sizeof(dl_reply_ind_t),
    DL_REPLY_STATUS_IND_SIZE = sizeof(dl_reply_status_ind_t),
    DL_REPLY_UPDATE_REQ_SIZE = sizeof(dl_reply_update_req_t),
    DL_REPLY_UPDATE_STATUS_IND_SIZE = sizeof(dl_reply_update_status_ind_t)
};
```

Constants

`DL_INFO_REQ_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_INFO_ACK_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_ATTACH_REQ_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_DETACH_REQ_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_BIND_REQ_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_BIND_ACK_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_UNBIND_REQ_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_SUBS_BIND_REQ_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_SUBS_BIND_ACK_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_SUBS_UNBIND_REQ_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_OK_ACK_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_ERROR_ACK_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_CONNECT_REQ_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_CONNECT_IND_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_CONNECT_RES_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_CONNECT_CON_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_TOKEN_REQ_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_TOKEN_ACK_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_DISCONNECT_REQ_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_DISCONNECT_IND_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_RESET_REQ_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_RESET_IND_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_RESET_RES_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_RESET_CON_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_UNITDATA_REQ_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_UNITDATA_IND_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_UDERROR_IND_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_UDQOS_REQ_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_ENABMULTI_REQ_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_DISABMULTI_REQ_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_PROMISCON_REQ_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_PROMISCOFF_REQ_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_PHYS_ADDR_REQ_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_PHYS_ADDR_ACK_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_SET_PHYS_ADDR_REQ_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_GET_STATISTICS_REQ_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_GET_STATISTICS_ACK_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_XID_REQ_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_XID_IND_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_XID_RES_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_XID_CON_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_TEST_REQ_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_TEST_IND_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_TEST_RES_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_TEST_CON_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_DATA_ACK_REQ_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_DATA_ACK_IND_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_DATA_ACK_STATUS_IND_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_REPLY_REQ_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_REPLY_IND_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_REPLY_STATUS_IND_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_REPLY_UPDATE_REQ_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_REPLY_UPDATE_STATUS_IND_SIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

DL_IOC_HDR_INFO

```
enum {
    DL_IOC_HDR_INFO = ((MIOC_DLPI << 8) | 10)
};
```

Constants

`DL_IOC_HDR_INFO`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

DL_NONE

```
enum {
    DL_NONE = 0x0B01,
    DL_MONITOR = 0x0B02,
    DL_MAXIMUM = 0x0B03
};
```

Constants

`DL_NONE`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`DL_MONITOR`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`DL_MAXIMUM`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

DL_PEER_BIND

```
enum {
    DL_PEER_BIND = 0x01,
    DL_HIERARCHICAL_BIND = 0x02
};
```

Constants

`DL_PEER_BIND`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`DL_HIERARCHICAL_BIND`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

DL_POLL_FINAL

```
enum {
    DL_POLL_FINAL = 0x01
};
```

Constants

`DL_POLL_FINAL`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

DL_PROMISC_OFF

```
enum {
    DL_PROMISC_OFF = 0
};
```

Constants

`DL_PROMISC_OFF`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

DL_PROMISC_PHYS

```
enum {
    DL_PROMISC_PHYS = 0x01,
    DL_PROMISC_SAP = 0x02,
    DL_PROMISC_MULTI = 0x03
};
```

Constants

`DL_PROMISC_PHYS`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`DL_PROMISC_SAP`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`DL_PROMISC_MULTI`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

DL_PROVIDER

```
enum {
    DL_PROVIDER = 0x0700,
    DL_USER = 0x0701
};
```

Constants

`DL_PROVIDER`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`DL_USER`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

DL_QOS_CO_RANGE1

```
enum {
    DL_QOS_CO_RANGE1 = 0x0101,
    DL_QOS_CO_SEL1 = 0x0102,
    DL_QOS_CL_RANGE1 = 0x0103,
    DL_QOS_CL_SEL1 = 0x0104
};
```

Constants

`DL_QOS_CO_RANGE1`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_QOS_CO_SEL1`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_QOS_CL_RANGE1`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_QOS_CL_SEL1`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

DL_RESET_FLOW_CONTROL

```
enum {
    DL_RESET_FLOW_CONTROL = 0x0900,
    DL_RESET_LINK_ERROR = 0x0901,
    DL_RESET_RESYNCH = 0x0902
};
```

Constants

`DL_RESET_FLOW_CONTROL`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_RESET_LINK_ERROR`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_RESET_RESYNCH`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

DL_RQST_RSP

```
enum {
    DL_RQST_RSP = 0x01,
    DL_RQST_NORSP = 0x02
};
```

Constants

`DL_RQST_RSP`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`DL_RQST_NORSP`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

DL_STYLE1

```
enum {
    DL_STYLE1 = 0x0500,
    DL_STYLE2 = 0x0501
};
```

Constants

`DL_STYLE1`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`DL_STYLE2`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

DL_UNATTACHED

```
enum {
    DL_UNATTACHED = 0x04,
    DL_ATTACH_PENDING = 0x05,
    DL_DETACH_PENDING = 0x06,
    DL_UNBOUND = 0x00,
    DL_BIND_PENDING = 0x01,
    DL_UNBIND_PENDING = 0x02,
    DL_IDLE = 0x03,
    DL_UDQOS_PENDING = 0x07,
    DL_OUTCON_PENDING = 0x08,
    DL_INCON_PENDING = 0x09,
    DL_CONN_RES_PENDING = 0x0A,
    DL_DATAXFER = 0x0B,
    DL_USER_RESET_PENDING = 0x0C,
    DL_PROV_RESET_PENDING = 0x0D,
    DL_RESET_RES_PENDING = 0x0E,
    DL_DISCON8_PENDING = 0x0F,
    DL_DISCON9_PENDING = 0x10,
    DL_DISCON11_PENDING = 0x11,
    DL_DISCON12_PENDING = 0x12,
    DL_DISCON13_PENDING = 0x13,
    DL_SUBS_BIND_PND = 0x14,
    DL_SUBS_UNBIND_PND = 0x15
};
```

Constants

`DL_UNATTACHED`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`DL_ATTACH_PENDING`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`DL_DETACH_PENDING`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`DL_UNBOUND`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`DL_BIND_PENDING`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`DL_UNBIND_PENDING`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`DL_IDLE`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`DL_UDQOS_PENDING`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_OUTCON_PENDING`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_INCON_PENDING`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_CONN_RES_PENDING`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_DATAXFER`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_USER_RESET_PENDING`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_PROV_RESET_PENDING`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_RESET_RES_PENDING`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_DISCON8_PENDING`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_DISCON9_PENDING`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_DISCON11_PENDING`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_DISCON12_PENDING`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_DISCON13_PENDING`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_SUBS_BIND_PND`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_SUBS_UNBIND_PND`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

DL_UNKNOWN

```
enum {
    DL_UNKNOWN = -1,
    DL_QOS_DONT_CARE = -2
};
```

Constants

`DL_UNKNOWN`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`DL_QOS_DONT_CARE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

DVMRP_INIT

```
enum {
    DVMRP_INIT = 100,
    DVMRP_DONE = 101,
    DVMRP_ADD_VIF = 102,
    DVMRP_DEL_VIF = 103,
    DVMRP_ADD_LGRP = 104,
    DVMRP_DEL_LGRP = 105,
    DVMRP_ADD_MRT = 106,
    DVMRP_DEL_MRT = 107
};
```

Constants

`DVMRP_INIT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`DVMRP_DONE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`DVMRP_ADD_VIF`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`DVMRP_DEL_VIF`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`DVMRP_ADD_LGRP`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`DVMRP_DEL_LGRP`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`DVMRP_ADD_MRT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`DVMRP_DEL_MRT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

EAddrType

```
typedef UInt32 EAddrType;
enum {
    keaStandardAddress = 0,
    keaMulticast = 1,
    keaBroadcast = 2,
    keaBadAddress = 3,
    keaRawPacketBit = 0x80000000,
    keaTimeStampBit = 0x40000000
};
```

Constants

`keaStandardAddress`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`keaMulticast`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`keaBroadcast`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`keaBadAddress`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`keaRawPacketBit`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`keaTimeStampBit`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

EPERM

```
enum {
    EPERM = 1,
    ENOENT = 2,
    ENORSRC = 3,
    EINTR = 4,
    EIO = 5,
    ENXIO = 6,
    EBADF = 9,
    EAGAIN = 11,
    ENOMEM = 12,
    EACCES = 13,
    EFAULT = 14,
    EBUSY = 16,
    EEXIST = 17,
    ENODEV = 19,
    EINVAL = 22,
    ENOTTY = 25,
    EPIPE = 32,
    ERANGE = 34,
    EDEADLK = 35,
    EWOULDBLOCK = 35,
    EALREADY = 37,
    ENOTSOCK = 38,
    EDESTADDRREQ = 39,
    EMSGSIZE = 40,
    EPROTOTYPE = 41,
    ENOPROTOOPT = 42,
    EPROTONOSUPPORT = 43,
    ESOCKTNOSUPPORT = 44,
    EOPNOTSUPP = 45,
    EADDRINUSE = 48,
    EADDRNOTAVAIL = 49,
    ENETDOWN = 50,
    ENETUNREACH = 51,
    ENETRESET = 52,
    ECONNABORTED = 53,
    ECONNRESET = 54,
    ENOBUFS = 55,
    EISCONN = 56,
    ENOTCONN = 57,
    ESHUTDOWN = 58,
    ETOOMANYREFS = 59,
    ETIMEDOUT = 60,
    ECONNREFUSED = 61,
    EHOSTDOWN = 64,
    EHOSTUNREACH = 65,
    EPROTO = 70,
    ETIME = 71,
    ENOSR = 72,
    EBADMSG = 73,
    ECANCEL = 74,
    ENOSTR = 75,
    ENODATA = 76,
    EINPROGRESS = 77,
    ESRCH = 78,
    ENOMSG = 79,
```

```
    ELASTERRNO = 79
};
```

Constants

`EPERM`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`ENOENT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`ENORSRC`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`EINTR`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`EIO`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`ENXIO`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`EBADF`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`EAGAIN`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`ENOMEM`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`EACCES`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`EFAULT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`EBUSY`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`EEXIST`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`ENODEV`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`EINVAL`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`ENOTTY`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`EPIPE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`ERANGE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`EDEADLK`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`EWOULDBLOCK`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`EALREADY`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`ENOTSOCK`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`EDESTADDRREQ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`EMSGSIZE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`EPROTOTYPE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`ENOPROTOOPT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`EPROTONOSUPPORT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

ESOCKTNOSUPPORT
Available in Mac OS X v10.0 and later.
Declared in OpenTransport.h.

EOPNOTSUPP
Available in Mac OS X v10.0 and later.
Declared in OpenTransport.h.

EADDRINUSE
Available in Mac OS X v10.0 and later.
Declared in OpenTransport.h.

EADDRNOTAVAIL
Available in Mac OS X v10.0 and later.
Declared in OpenTransport.h.

ENETDOWN
Available in Mac OS X v10.0 and later.
Declared in OpenTransport.h.

ENETUNREACH
Available in Mac OS X v10.0 and later.
Declared in OpenTransport.h.

ENETRESET
Available in Mac OS X v10.0 and later.
Declared in OpenTransport.h.

ECONNABORTED
Available in Mac OS X v10.0 and later.
Declared in OpenTransport.h.

ECONNRESET
Available in Mac OS X v10.0 and later.
Declared in OpenTransport.h.

ENOBUFS
Available in Mac OS X v10.0 and later.
Declared in OpenTransport.h.

EISCONN
Available in Mac OS X v10.0 and later.
Declared in OpenTransport.h.

ENOTCONN
Available in Mac OS X v10.0 and later.
Declared in OpenTransport.h.

ESHUTDOWN
Available in Mac OS X v10.0 and later.
Declared in OpenTransport.h.

ETOOMANYREFS
Available in Mac OS X v10.0 and later.
Declared in OpenTransport.h.

`ETIMEDOUT`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`ECONNREFUSED`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`EHOSTDOWN`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`EHOSTUNREACH`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`EPROTO`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`ETIME`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`ENOSR`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`EBADMSG`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`ECANCEL`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`ENOSTR`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`ENODATA`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`EINPROGRESS`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`ESRCH`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`ENOMSG`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`ELASTERRNO`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

FLUSHALL

```
enum {
    FLUSHALL = 1,
    FLUSHDATA = 0
};
```

Constants

`FLUSHALL`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`FLUSHDATA`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

FLUSHR

```
enum {
    FLUSHR = 0x01,
    FLUSHW = 0x02,
    FLUSHRW = (FLUSHW | FLUSHR)
};
```

Constants

`FLUSHR`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`FLUSHW`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`FLUSHRW`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

FMNAMESZ

```
enum {
    FMNAMESZ = 31
};
```

Constants

`FMNAMESZ`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

I_NREAD

```
enum {
    I_NREAD = ((MIOC_STREAMIO << 8) | 1),
    I_PUSH = ((MIOC_STREAMIO << 8) | 2),
    I_POP = ((MIOC_STREAMIO << 8) | 3),
    I_LOOK = ((MIOC_STREAMIO << 8) | 4),
    I_FLUSH = ((MIOC_STREAMIO << 8) | 5),
    I_SRDOPT = ((MIOC_STREAMIO << 8) | 6),
    I_GRDOPT = ((MIOC_STREAMIO << 8) | 7),
    I_STR = ((MIOC_STREAMIO << 8) | 8),
    I_SETSIG = ((MIOC_STREAMIO << 8) | 9),
    I_GETSIG = ((MIOC_STREAMIO << 8) | 10),
    I_FIND = ((MIOC_STREAMIO << 8) | 11),
    I_LINK = ((MIOC_STREAMIO << 8) | 12),
    I_UNLINK = ((MIOC_STREAMIO << 8) | 13),
    I_PEEK = ((MIOC_STREAMIO << 8) | 15),
    I_FDINSERT = ((MIOC_STREAMIO << 8) | 16),
    I_SENDFD = ((MIOC_STREAMIO << 8) | 17),
    I_RECVFD = ((MIOC_STREAMIO << 8) | 18),
    I_FLUSHBAND = ((MIOC_STREAMIO << 8) | 19),
    I_SWROPT = ((MIOC_STREAMIO << 8) | 20),
    I_GWROPT = ((MIOC_STREAMIO << 8) | 21),
    I_LIST = ((MIOC_STREAMIO << 8) | 22),
    I_ATMARK = ((MIOC_STREAMIO << 8) | 23),
    I_CKBAND = ((MIOC_STREAMIO << 8) | 24),
    I_GETBAND = ((MIOC_STREAMIO << 8) | 25),
    I_CANPUT = ((MIOC_STREAMIO << 8) | 26),
    I_SETCLTIME = ((MIOC_STREAMIO << 8) | 27),
    I_GETCLTIME = ((MIOC_STREAMIO << 8) | 28),
    I_PLINK = ((MIOC_STREAMIO << 8) | 29),
    I_PUNLINK = ((MIOC_STREAMIO << 8) | 30),
    I_GETMSG = ((MIOC_STREAMIO << 8) | 40),
    I_PUTMSG = ((MIOC_STREAMIO << 8) | 41),
    I_POLL = ((MIOC_STREAMIO << 8) | 42),
    I_SETDELAY = ((MIOC_STREAMIO << 8) | 43),
    I_GETDELAY = ((MIOC_STREAMIO << 8) | 44),
    I_RUN_QUEUES = ((MIOC_STREAMIO << 8) | 45),
    I_GETPMSG = ((MIOC_STREAMIO << 8) | 46),
    I_PUTPMSG = ((MIOC_STREAMIO << 8) | 47),
    I_AUTOPUSH = ((MIOC_STREAMIO << 8) | 48),
    I_PIPE = ((MIOC_STREAMIO << 8) | 49),
    I_HEAP_REPORT = ((MIOC_STREAMIO << 8) | 50),
    I_FIFO = ((MIOC_STREAMIO << 8) | 51)
};
```

Constants

`I_NREAD`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`I_PUSH`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`I_POP`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_LOOK`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_FLUSH`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_SRDOPT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_GRDOPT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_STR`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_SETSIG`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_GETSIG`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_FIND`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_LINK`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_UNLINK`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_PEEK`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_FDINSERT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_SENDFD`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_RECVFD`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_FLUSHBAND`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_SWROPT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_GWROPT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_LIST`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_ATMARK`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_CKBAND`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_GETBAND`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_CANPUT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_SETCLTIME`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_GETCLTIME`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_PLINK`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_PUNLINK`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_GETMSG`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_PUTMSG`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_POLL`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_SETDELAY`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_GETDELAY`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_RUN_QUEUES`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_GETPMSG`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_PUTPMSG`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_AUTOPUSH`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_PIPE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_HEAP_REPORT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`I_FIFO`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

I_OTGetMiscellaneousEvents

```
enum {
    I_OTGetMiscellaneousEvents = ((MIOC_OT << 8) | 1),
    I_OTSetFramingType = ((MIOC_OT << 8) | 2),
    kOTGetFramingValue = 0xFFFFFFFF,
    I_OTSetRawMode = ((MIOC_OT << 8) | 3),
    kOTSetRecvMode = 0x01,
    kOTSendErrorPacket = 0x02,
    I_OTConnect = ((MIOC_OT << 8) | 4),
    I_OTDisconnect = ((MIOC_OT << 8) | 5),
    I_OTScript = ((MIOC_OT << 8) | 6)
};
```

Constants

`I_OTGetMiscellaneousEvents`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`I_OTSetFramingType`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTGetFramingValue`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`I_OTSetRawMode`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTSetRecvMode`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTSendErrorPacket`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`I_OTConnect`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`I_OTDisconnect`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`I_OTScript`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

I_OTISDNAlerting

```
enum {
    I_OTISDNAlerting = ((MIOC_ISDN << 8) | 100),
    I_OTISDNSuspend = ((MIOC_ISDN << 8) | 101),
    I_OTISDNSuspendAcknowledge = ((MIOC_ISDN << 8) | 102),
    I_OTISDNSuspendReject = ((MIOC_ISDN << 8) | 103),
    I_OTISDNResume = ((MIOC_ISDN << 8) | 104),
    I_OTISDNResumeAcknowledge = ((MIOC_ISDN << 8) | 105),
    I_OTISDNResumeReject = ((MIOC_ISDN << 8) | 106),
    I_OTISDNFaciltity = ((MIOC_ISDN << 8) | 107)
};
```

Constants

`I_OTISDNAlerting`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`I_OTISDNSuspend`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`I_OTISDNSuspendAcknowledge`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`I_OTISDNSuspendReject`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`I_OTISDNResume`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`I_OTISDNResumeAcknowledge`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`I_OTISDNResumeReject`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`I_OTISDNFaciltity`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

I_SAD_SAP

```
enum {
    I_SAD_SAP = ((MIOC_SAD << 8) | 1),
    I_SAD_GAP = ((MIOC_SAD << 8) | 2),
    I_SAD_VML = ((MIOC_SAD << 8) | 3)
};
```

Constants

`I_SAD_SAP`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`I_SAD_GAP`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`I_SAD_VML`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

I_SetSerialDTR

```
enum {
    I_SetSerialDTR = ((MIOC_SRL << 8) | 0),
    kOTSerialSetDTROff = 0,
    kOTSerialSetDTROn = 1,
    I_SetSerialBreak = ((MIOC_SRL << 8) | 1),
    kOTSerialSetBreakOn = 0xFFFFFFFF,
    kOTSerialSetBreakOff = 0,
    I_SetSerialXOffState = ((MIOC_SRL << 8) | 2),
    kOTSerialForceXOffTrue = 1,
    kOTSerialForceXOffFalse = 0,
    I_SetSerialXOn = ((MIOC_SRL << 8) | 3),
    kOTSerialSendXOnAlways = 1,
    kOTSerialSendXOnIfXOffTrue = 0,
    I_SetSerialXOff = ((MIOC_SRL << 8) | 4),
    kOTSerialSendXOffAlways = 1,
    kOTSerialSendXOffIfXOnTrue = 0
};
```

Constants

`I_SetSerialDTR`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`kOTSerialSetDTROff`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`kOTSerialSetDTROn`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`I_SetSerialBreak`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialSetBreakOn`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialSetBreakOff`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`I_SetSerialXOffState`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialForceXOffTrue`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialForceXOffFalse`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`I_SetSerialXOn`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialSendXOnAlways`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialSendXOnIfXOffTrue`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`I_SetSerialXOff`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialSendXOffAlways`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialSendXOffIfXOnTrue`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

I_TRCLOG

```
enum {
    I_TRCLOG = ((MIOC_STRLOG << 8) | 1),
    I_ERRLOG = ((MIOC_STRLOG << 8) | 2)
};
```

Constants

`I_TRCLOG`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`I_ERRLOG`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

INET_IP

```
enum {
    INET_IP = 0x00,
    INET_TCP = 0x06,
    INET_UDP = 0x11
};
```

Constants

`INET_IP`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`INET_TCP`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`INET_UDP`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

INFPSZ

```
enum {
    INFPSZ = -1
};
```

Constants

`INFPSZ`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

INFTIM

```
enum {
    INFTIM = 0xFFFFFFFF
};
```

Constants

`INFTIM`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

IP_OPTIONS

```
enum {
    IP_OPTIONS = 1,
    IP_TOS = 2,
    IP_TTL = 3,
    IP_REUSEADDR = 4,
    IP_DONTROUTE = 16,
    IP_BROADCAST = 32,
    IP_REUSEPORT = 512,
    IP_HDRINCL = 4098,
    IP_RCVOPTS = 4101,
    IP_RCVDSTADDR = 4103,
    IP_MULTICAST_IF = 4112,
    IP_MULTICAST_TTL = 4113,
    IP_MULTICAST_LOOP = 4114,
    IP_ADD_MEMBERSHIP = 4115,
    IP_DROP_MEMBERSHIP = 4116,
    IP_BROADCAST_IFNAME = 4117,
    IP_RCVIFADDR = 4118
};
```

Constants

```
IP_OPTIONS
IP_TOS
IP_TTL
IP_REUSEADDR
IP_DONTROUTE
IP_BROADCAST
IP_REUSEPORT
IP_HDRINCL
IP_RCVOPTS
IP_RCVDSTADDR
IP_MULTICAST_IF
IP_MULTICAST_TTL
IP_MULTICAST_LOOP
IP_ADD_MEMBERSHIP
IP_DROP_MEMBERSHIP
IP_BROADCAST_IFNAME
IP_RCVIFADDR
```

IPCP_OPT_GETREMOTEPROTOADDR

```
enum {
    IPCP_OPT_GETREMOTEPROTOADDR = 0x00007000,
    IPCP_OPT_GETLOCALPROTOADDR = 0x00007001,
    IPCP_OPT_TCPHDRCOMPRESSION = 0x00007002,
    LCP_OPT_PPPCOMPRESSION = 0x00007003,
    LCP_OPT_MRU = 0x00007004,
    LCP_OPT_RCACCMAP = 0x00007005,
    LCP_OPT_TXACCMAP = 0x00007006,
    SEC_OPT_OUTAUTHENTICATION = 0x00007007,
    SEC_OPT_ID = 0x00007008,
    SEC_OPT_PASSWORD = 0x00007009,
```

```
    CC_OPT_REMINDERTIMER = 0x00007010,
    CC_OPT_IPIDLETIMER = 0x00007011,
    CC_OPT_DTEADDRESSTYPE = 0x00007012,
    CC_OPT_DTEADDRESS = 0x00007013,
    CC_OPT_CALLINFO = 0x00007014,
    CC_OPT_GETMISCINFO = 0x00007015,
    PPP_OPT_GETCURRENTSTATE = 0x00007016,
    LCP_OPT_ECHO = 0x00007017,
    CC_OPT_SERIALPORTNAME = 0x00007200
};
```

Constants

`IPCP_OPT_GETREMOTEPROTOADDR`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`IPCP_OPT_GETLOCALPROTOADDR`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`IPCP_OPT_TCPHDRCOMPRESSION`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`LCP_OPT_PPPCOMPRESSION`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`LCP_OPT_MRU`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`LCP_OPT_RCACCMAP`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`LCP_OPT_TXACCMAP`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`SEC_OPT_OUTAUTHENTICATION`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`SEC_OPT_ID`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`SEC_OPT_PASSWORD`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`CC_OPT_REMINDERTIMER`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`CC_OPT_IPIDLETIMER`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`CC_OPT_DTEADDRESSTYPE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`CC_OPT_DTEADDRESS`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`CC_OPT_CALLINFO`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`CC_OPT_GETMISCINFO`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`PPP_OPT_GETCURRENTSTATE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`LCP_OPT_ECHO`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`CC_OPT_SERIALPORTNAME`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

ISDN_OPT_COMMTYPE

```
enum {
    ISDN_OPT_COMMTYPE = 0x0200,
    ISDN_OPT_FRAMINGTYPE = 0x0201,
    ISDN_OPT_56KADAPTATION = 0x0202
};
```

Constants

`ISDN_OPT_COMMTYPE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`ISDN_OPT_FRAMINGTYPE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`ISDN_OPT_56KADAPTATION`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

k8022BasicAddressLength

```
enum {
    k8022BasicAddressLength = sizeof(OTAddressType) + k48BitAddrLength
+ sizeof(UInt16),
    k8022SNAPAddressLength = sizeof(OTAddressType) + k48BitAddrLength
+ sizeof(UInt16) + k8022SNAPLength
};
```

Constants

`k8022BasicAddressLength`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`k8022SNAPAddressLength`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kAF_ISDN

```
enum {
    kAF_ISDN = 0x2000
};
```

Constants

`kAF_ISDN`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kAllATalkRoutersDown

```
enum {
    kAllATalkRoutersDown = 0,
    kLocalATalkRoutersDown = -1L,
    kARARouterDisconnected = -2L
};
```

Constants

`kAllATalkRoutersDown`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kLocalATalkRoutersDown`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kARARouterDisconnected`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kAllDHCPOptions

```
enum {
    kAllDHCPOptions = -1,
    kDHCPLongOption = 126,
    kDHCPLongOptionReq = 127
};
```

Constants

`kAllDHCPOptions`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kDHCPLongOption`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kDHCPLongOptionReq`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kAppleTalkEvent

```
enum {
    kAppleTalkEvent = kPROTOCOLEVENT | 0x00010000,
    T_GETMYZONECOMPLETE = kAppleTalkEvent + 1,
    T_GETLOCALZONESCOMPLETE = kAppleTalkEvent + 2,
    T_GETZONELISTCOMPLETE = kAppleTalkEvent + 3,
    T_GETATALKINFOCOMPLETE = kAppleTalkEvent + 4,
    T_ATALKROUTERDOWNEVENT = kAppleTalkEvent + 51,
    T_ATALKROUTERUPEVENT = kAppleTalkEvent + 52,
    T_ATALKZONENAMECHANGEDEVENT = kAppleTalkEvent + 53,
    T_ATALKCONNECTIVITYCHANGEDEVENT = kAppleTalkEvent + 54,
    T_ATALKINTERNETAVAILABLEEVENT = kAppleTalkEvent + 55,
    T_ATALKCABLERANGECHANGEDEVENT = kAppleTalkEvent + 56
};
```

Constants

`kAppleTalkEvent`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`T_GETMYZONECOMPLETE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`T_GETLOCALZONESCOMPLETE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`T_GETZONELISTCOMPLETE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`T_GETATALKINFOCOMPLETE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`T_ATALKROUTERDOWNEVENT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`T_ATALKROUTERUPEVENT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`T_ATALKZONENAMECHANGEDEVENT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`T_ATALKCONNECTIVITYCHANGEDEVENT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`T_ATALKINTERNETAVAILABLEEVENT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`T_ATALKCABLERANGECHANGEDEVENT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kARARouterOnline

```
enum {
    kARARouterOnline = -1L,
    kATalkRouterOnline = 0,
    kLocalATalkRouterOnline = -2L
};
```

Constants

`kARARouterOnline`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kATalkRouterOnline`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kLocalATalkRouterOnline`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kATalkInfoIsExtended

```
enum {
    kATalkInfoIsExtended = 0x0001,
    kATalkInfoHasRouter = 0x0002,
    kATalkInfoOneZone = 0x0004
};
```

Constants

`kATalkInfoIsExtended`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kATalkInfoHasRouter`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kATalkInfoOneZone`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kCCReminderTimerDisabled

```
enum {
    kCCReminderTimerDisabled = 0,
    kCCIPIdleTimerDisabled = 0
};
```

Constants

`kCCReminderTimerDisabled`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kCCIPIdleTimerDisabled`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kDDPAddressLength

```
enum {
    kDDPAddressLength = 8,
    kNBPAddressLength = kNBPEntityBufferSize,
    kAppleTalkAddressLength = kDDPAddressLength + kNBPEntityBufferSize
};
```

Constants

`kDDPAddressLength`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kNBPAddressLength`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kAppleTalkAddressLength`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kDefaultAppleTalkServicesPath

```
enum {
    kDefaultAppleTalkServicesPath = -3
};
```

Constants
`kDefaultAppleTalkServicesPath`

kDefaultInetInterface

```
enum {
    kDefaultInetInterface = -1
};
```

Constants
`kDefaultInetInterface`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kDefaultInternetServicesPath

```
enum {
    kDefaultInternetServicesPath = -3
};
```

Constants
`kDefaultInternetServicesPath`

kE164Address

```
enum {
    kE164Address = 1,
    kPhoneAddress = 1,
    kCompoundPhoneAddress = 2,
    kX121Address = 3
};
```

Constants
`kE164Address`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kPhoneAddress`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kCompoundPhoneAddress`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kX121Address`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kECHO_TSDU

```
enum {
    kECHO_TSDU = 585
};
```

Constants
`kECHO_TSDU`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kEnetPacketHeaderLength

```
enum {
    kEnetPacketHeaderLength = (2 * k48BitAddrLength) + k8022DLSAPLength,
    kEnetTSDU = 1514,
    kTokenRingTSDU = 4458,
    kFDDITSDU = 4458,
    k8022SAPLength = 1,
    k8022BasicHeaderLength = 3,
    k8022SNAPHeaderLength = k8022SNAPLength + k8022BasicHeaderLength
};
```

Constants

`kEnetPacketHeaderLength`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kEnetTSDU`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kTokenRingTSDU`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kFDDITSDU`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`k8022SAPLength`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`k8022BasicHeaderLength`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`k8022SNAPHeaderLength`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kFirstMinorNumber

```
enum {
    kFirstMinorNumber = 10
};
```

Constants
`kFirstMinorNumber`

kInetInterfaceInfoVersion

```
enum {
    kInetInterfaceInfoVersion = 3
};
```

Constants
`kInetInterfaceInfoVersion`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kIP_OPTIONS

```
enum {
    kIP_OPTIONS = 0x01,
    kIP_TOS = 0x02,
    kIP_TTL = 0x03,
    kIP_REUSEADDR = 0x04,
    kIP_DONTROUTE = 0x10,
    kIP_BROADCAST = 0x20,
    kIP_REUSEPORT = 0x0200,
    kIP_HDRINCL = 0x1002,
    kIP_RCVOPTS = 0x1005,
    kIP_RCVDSTADDR = 0x1007,
    kIP_MULTICAST_IF = 0x1010,
    kIP_MULTICAST_TTL = 0x1011,
    kIP_MULTICAST_LOOP = 0x1012,
    kIP_ADD_MEMBERSHIP = 0x1013,
    kIP_DROP_MEMBERSHIP = 0x1014,
    kIP_BROADCAST_IFNAME = 0x1015,
    kIP_RCVIFADDR = 0x1016
};
```

Constants
`kIP_OPTIONS`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kIP_TOS`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kIP_TTL`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kIP_REUSEADDR`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kIP_DONTROUTE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kIP_BROADCAST`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kIP_REUSEPORT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kIP_HDRINCL`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kIP_RCVOPTS`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kIP_RCVDSTADDR`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kIP_MULTICAST_IF`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kIP_MULTICAST_TTL`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kIP_MULTICAST_LOOP`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kIP_ADD_MEMBERSHIP`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kIP_DROP_MEMBERSHIP`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kIP_BROADCAST_IFNAME`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kIP_RCVIFADDR`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

kIPCPTCPHdrCompressionDisabled

```
enum {
    kIPCPTCPHdrCompressionDisabled = 0,
    kIPCPTCPHdrCompressionEnabled = 1
};
```

Constants

`kIPCPTCPHdrCompressionDisabled`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`kIPCPTCPHdrCompressionEnabled`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

kISDNModuleID

```
enum {
    kISDNModuleID = 7300
};
```

Constants

`kISDNModuleID`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

kMaxHostAddrs

```
enum {
    kMaxHostAddrs = 10,
    kMaxSysStringLen = 32,
    kMaxHostNameLen = 255
};
```

Constants

`kMaxHostAddrs`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`kMaxSysStringLen`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`kMaxHostNameLen`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

Port-Related Constants

Provide length and size values for modules, provider names, and slot IDs.

```
enum {
    kMaxModuleNameLength = 31,
    kMaxModuleNameSize = kMaxModuleNameLength + 1,
    kMaxProviderNameLength = kMaxModuleNameLength + 4,
    kMaxProviderNameSize = kMaxProviderNameLength + 1,
    kMaxSlotIDLength = 7,
    kMaxSlotIDSize = kMaxSlotIDLength + 1,
    kMaxResourceInfoLength = 31,
    kMaxResourceInfoSize = 32,
    kMaxPortNameLength = kMaxModuleNameLength + 4,
    kMaxPortNameSize = kMaxPortNameLength + 1
};
```

Constants

`kMaxModuleNameLength`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kMaxModuleNameSize`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kMaxProviderNameLength`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kMaxProviderNameSize`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kMaxSlotIDLength`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kMaxSlotIDSize`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kMaxResourceInfoLength`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kMaxResourceInfoSize`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kMaxPortNameLength`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kMaxPortNameSize`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

Discussion
These constants provide length and size values for modules, provider names, and slot IDs. These fields all end with a byte for the terminating zero. The constant kMaxProviderNameSize permits a length of 36 bytes: 31 bytes for the name, up to 4 bytes of extra characters (called minor numbers in STREAMS specifications, and currently not used), and a byte for the zero that terminates the string.

kMaxServices

```
enum {
    kMaxServices = 20
};
```

Constants
`kMaxServices`

kMulticastLength

```
enum {
    kMulticastLength = 6,
    k48BitAddrLength = 6,
    k8022DLSAPLength = 2,
    k8022SNAPLength = 5,
    kEnetAddressLength = k48BitAddrLength + k8022DLSAPLength,
    kSNAPSAP = 0x00AA,
    kIPXSAP = 0x00FF,
    kMax8022SAP = 0x00FE,
    k8022GlobalSAP = 0x00FF,
    kMinDIXSAP = 1501,
    kMaxDIXSAP = 0xFFFF
};
```

Constants
`kMulticastLength`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`k48BitAddrLength`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`k8022DLSAPLength`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`k8022SNAPLength`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kEnetAddressLength`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kSNAPSAP`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kIPXSAP`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kMax8022SAP`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`k8022GlobalSAP`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kMinDIXSAP`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kMaxDIXSAP`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kNBPMaxNameLength

```
enum {
    kNBPMaxNameLength = 32,
    kNBPMaxTypeLength = 32,
    kNBPMaxZoneLength = 32,
    kNBPSlushLength = 9,
    kNBPMaxEntityLength = (kNBPMaxNameLength + kNBPMaxTypeLength
+ kNBPMaxZoneLength + 3),
    kNBPEntityBufferSize = (kNBPMaxNameLength + kNBPMaxTypeLength
+ kNBPMaxZoneLength + kNBPSlushLength),
    kNBPWildCard = 0x3D,
    kNBPImbeddedWildCard = 0xC5,
    kNBPDefaultZone = 0x2A
};
```

Constants

`kNBPMaxNameLength`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kNBPMaxTypeLength`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kNBPMaxZoneLength`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kNBPSlushLength`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kNBPMaxEntityLength`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kNBPEntityBufferSize`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kNBPWildCard`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kNBPImbeddedWildCard`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kNBPDefaultZone`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kNetbufDataIsOTData

```
enum {
    kNetbufDataIsOTData = 0xFFFFFFFE
};
```

Constants

`kNetbufDataIsOTData`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

kO_ASYNC

Constants

kOTAnyInetAddress

```
enum {
    kOTAnyInetAddress = 0
};
```

Constants

`kOTAnyInetAddress`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

kOTAutopushMax

```
enum {
    kOTAutopushMax = 8
};
```

Constants

`kOTAutopushMax`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

kOTCFMClass

```
enum {
    kOTCFMClass = 'otan'
};
```

Constants

`kOTCFMClass`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

kOTDefaultConfigurator

```
enum {
    kOTDefaultConfigurator = 0,
    kOTProtocolFamilyConfigurator = 1,
    kOTLinkDriverConfigurator = 2
};
```

Constants

`kOTDefaultConfigurator`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`kOTProtocolFamilyConfigurator`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`kOTLinkDriverConfigurator`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

kOTFLUSHBAND

```
enum {
    kOTFLUSHBAND = 0x40
};
```

Constants
`kOTFLUSHBAND`

Port Framing Capabilities

Describe a port's framing capabilities.

```
enum {
    kOTFramingEthernet = 0x01,
    kOTFramingEthernetIPX = 0x02,
    kOTFraming8023 = 0x04,
    kOTFraming8022 = 0x08
};
```

Constants
`kOTFramingEthernet`
The port can use standard Ethernet framing.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`kOTFramingEthernetIPX`
The port can use IPX Ethernet framing.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`kOTFraming8023`
The port can use 802.3 Ethernet framing.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`kOTFraming8022`
The port can use 802.2 Ethernet framing.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

Discussion
This enumeration contains flags indicating the type of framing capability that a port has. If the port can handle only one type of framing, this field is 0. This field is dependent on the ports device type.

kOTGenericName

```
enum {
    kOTGenericName = 0
};
```

Constants

`kOTGenericName`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

kOTGetDataSymbol

```
enum {
    kOTGetDataSymbol = 0,
    kOTGetCodeSymbol = 1,
    kOTLoadNewCopy = 2,
    kOTLoadACopy = 4,
    kOTFindACopy = 8,
    kOTLibMask = kOTLoadNewCopy | kOTLoadACopy | kOTFindACopy,
    kOTLoadLibResident = 0x20
};
```

Constants

`kOTGetDataSymbol`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`kOTGetCodeSymbol`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`kOTLoadNewCopy`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`kOTLoadACopy`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`kOTFindACopy`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`kOTLibMask`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`kOTLoadLibResident`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

kOTInitialScan

```
enum {
    kOTInitialScan = 0,
    kOTScanAfterSleep = 1
};
```

Constants

`kOTInitialScan`

`kOTScanAfterSleep`

kOTInvalidPortRef

```
enum {
    kOTInvalidPortRef = 0
};
```

Constants

`kOTInvalidPortRef`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

kOTInvalidRef

```
enum {
    kOTInvalidRef = 0,
    kOTInvalidProviderRef = 0,
    kOTInvalidEndpointRef = 0,
    kOTInvalidMapperRef = 0
};
```

Constants
`kOTInvalidRef`
`kOTInvalidProviderRef`
`kOTInvalidEndpointRef`
`kOTInvalidMapperRef`

kOTInvalidStreamRef

```
enum {
    kOTInvalidStreamRef = 0
};
```

Constants
`kOTInvalidStreamRef`

kOTISDNDefaultCommType

```
enum {
    kOTISDNDefaultCommType = kOTISDNDigital64k,
    kOTISDNDefaultFramingType = kOTISDNFramingHDLC,
    kOTISDNDefault56KAdaptation = kOTISDNNot56KAdaptation
};
```

Constants

`kOTISDNDefaultCommType`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNDefaultFramingType`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNDefault56KAdaptation`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kOTISDNFramingTransparent

```
enum {
    kOTISDNFramingTransparent = 0x0010,
    kOTISDNFramingHDLC = 0x0020,
    kOTISDNFramingV110 = 0x0040,
    kOTISDNFramingV14E = 0x0080
};
```

Constants

`kOTISDNFramingTransparent`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNFramingHDLC`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNFramingV110`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNFramingV14E`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kOTISDNFramingTransparentSupported

```
enum {
    kOTISDNFramingTransparentSupported = 0x0010,
    kOTISDNFramingHDLCSupported = 0x0020,
    kOTISDNFramingV110Supported = 0x0040,
    kOTISDNFramingV14ESupported = 0x0080
};
```

Constants

`kOTISDNFramingTransparentSupported`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNFramingHDLCSupported`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNFramingV110Supported`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNFramingV14ESupported`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kOTISDNMaxPhoneSize

```
enum {
    kOTISDNMaxPhoneSize = 32,
    kOTISDNMaxSubSize = 4
};
```

Constants

`kOTISDNMaxPhoneSize`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`kOTISDNMaxSubSize`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

kOTISDNMaxUserDataSize

```
enum {
    kOTISDNMaxUserDataSize = 32
};
```

Constants

`kOTISDNMaxUserDataSize`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

kOTISDNNot56KAdaptation

```
enum {
    kOTISDNNot56KAdaptation = false,
    kOTISDN56KAdaptation = true
};
```

Constants

`kOTISDNNot56KAdaptation`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`kOTISDN56KAdaptation`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

kOTISDNTelephoneALaw

```
enum {
    kOTISDNTelephoneALaw = 1,
    kOTISDNTelephoneMuLaw = 26,
    kOTISDNDigital64k = 13,
    kOTISDNDigital56k = 37,
    kOTISDNVideo64k = 41,
    kOTISDNVideo56k = 42
};
```

Constants

`kOTISDNTelephoneALaw`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNTelephoneMuLaw`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNDigital64k`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNDigital56k`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNVideo64k`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNVideo56k`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kOTISDNUnallocatedNumber

```
enum {
    kOTISDNUnallocatedNumber = 1,
    kOTISDNNoRouteToSpecifiedTransitNetwork = 2,
    kOTISDNNoRouteToDestination = 3,
    kOTISDNChannelUnacceptable = 6,
    kOTISDNNormal = 16,
    kOTISDNUserBusy = 17,
    kOTISDNNoUserResponding = 18,
    kOTISDNNoAnswerFromUser = 19,
    kOTISDNCallRejected = 21,
    kOTISDNNumberChanged = 22,
    kOTISDNNonSelectedUserClearing = 26,
    kOTISDNDestinationOutOfOrder = 27,
    kOTISDNInvalidNumberFormat = 28,
    kOTISDNFacilityRejected = 29,
    kOTISDNNormalUnspecified = 31,
    kOTISDNNoCircuitChannelAvailable = 34,
    kOTISDNNetworkOutOfOrder = 41,
    kOTISDNSwitchingEquipmentCongestion = 42,
    kOTISDNAccessInformationDiscarded = 43,
    kOTISDNRequestedCircuitChannelNotAvailable = 44,
    kOTISDNResourceUnavailableUnspecified = 45,
    kOTISDNQualityOfServiceUnvailable = 49,
    kOTISDNRequestedFacilityNotSubscribed = 50,
    kOTISDNBearerCapabilityNotAuthorized = 57,
    kOTISDNBearerCapabilityNotPresentlyAvailable = 58,
    kOTISDNCallRestricted = 59,
    kOTISDNServiceOrOptionNotAvilableUnspecified = 63,
    kOTISDNBearerCapabilityNotImplemented = 65,
    kOTISDNRequestedFacilityNotImplemented = 69,
    kOTISDNOnlyRestrictedDigitalBearer = 70,
    kOTISDNServiceOrOptionNotImplementedUnspecified = 79,
    kOTISDNCallIdentityNotUsed = 83,
    kOTISDNCallIdentityInUse = 84,
    kOTISDNNoCallSuspended = 85,
    kOTISDNCallIdentityCleared = 86,
    kOTISDNIncompatibleDestination = 88,
    kOTISDNInvalidTransitNetworkSelection = 91,
    kOTISDNInvalidMessageUnspecified = 95,
    kOTISDNMandatoryInformationElementIsMissing = 96,
    kOTISDNMessageTypeNonExistentOrNotImplemented = 97,
    kOTISDNInterworkingUnspecified = 127
};
```

Constants

`kOTISDNUnallocatedNumber`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`kOTISDNNoRouteToSpecifiedTransitNetwork`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`kOTISDNNoRouteToDestination`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNChannelUnacceptable`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNNormal`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNUserBusy`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNNoUserResponding`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNNoAnswerFromUser`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNCallRejected`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNNumberChanged`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNNonSelectedUserClearing`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNDestinationOutOfOrder`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNInvalidNumberFormat`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNFacilityRejected`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNNormalUnspecified`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNNoCircuitChannelAvailable`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNNetworkOutOfOrder`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNSwitchingEquipmentCongestion`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNAccessInformationDiscarded`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNRequestedCircuitChannelNotAvailable`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNResourceUnavailableUnspecified`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNQualityOfServiceUnvailable`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNRequestedFacilityNotSubscribed`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNBearerCapabilityNotAuthorized`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNBearerCapabilityNotPresentlyAvailable`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNCallRestricted`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNServiceOrOptionNotAvilableUnspecified`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNBearerCapabilityNotImplemented`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNRequestedFacilityNotImplemented`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNOnlyRestrictedDigitalBearer`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNServiceOrOptionNotImplementedUnspecified`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNCallIdentityNotUsed`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNCallIdentityInUse`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNNoCallSuspended`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNCallIdentityCleared`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNIncompatibleDestination`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNInvalidTransitNetworkSelection`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNInvalidMessageUnspecified`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNMandatoryInformationElementIsMissing`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNMessageTypeNonExistentOrNotImplemented`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTISDNInterworkingUnspecified`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kOTLastSlotNumber

```
enum {
    kOTLastSlotNumber = 255,
    kOTLastOtherNumber = 255
};
```

Constants

`kOTLastSlotNumber`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTLastOtherNumber`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

kOTLvlFatal

```
enum {
    kOTLvlFatal = 0,
    kOTLvlNonfatal = 1,
    kOTLvlExtFatal = 2,
    kOTLvlExtNonfatal = 3,
    kOTLvlUserErr = 4,
    kOTLvlInfoErr = 5,
    kOTLvlInfoOnly = 6
};
```

Constants
kOTLvlFatal
kOTLvlNonfatal
kOTLvlExtFatal
kOTLvlExtNonfatal
kOTLvlUserErr
kOTLvlInfoErr
kOTLvlInfoOnly

kOTMinimumTimerValue

```
enum {
    kOTMinimumTimerValue = 8
};
```

Constants
kOTMinimumTimerValue

kOTModIsDriver

```
enum {
    kOTModIsDriver = 1,
    kOTModIsModule = 2,
    kOTModNoWriter = 16,
    kOTModUpperIsTPI = 4096,
    kOTModUpperIsDLPI = 8192,
    kOTModLowerIsTPI = 16384,
    kOTModLowerIsDLPI = 32768,
    kOTModGlobalContext = 8388608,
    kOTModUsesInterrupts = 134217728,
    kOTModIsComplexDriver = 536870912,
    kOTModIsFilter = 1073741824
};
```

Constants
kOTModIsDriver
kOTModIsModule
kOTModNoWriter
kOTModUpperIsTPI
kOTModUpperIsDLPI
kOTModLowerIsTPI

```
kOTModLowerIsDLPI
kOTModGlobalContext
kOTModUsesInterrupts
kOTModIsComplexDriver
kOTModIsFilter
```

kOTNetbufDataIsOTBufferStar

```
enum {
    kOTNetbufDataIsOTBufferStar = 0xFFFFFFFD
};
```

Constants

`kOTNetbufDataIsOTBufferStar`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

kOTNetbufIsRawMode

```
enum {
    kOTNetbufIsRawMode = 0xFFFFFFFF
};
```

Constants

`kOTNetbufIsRawMode`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

kOTNoMemoryConfigurationPtr

```
enum {
    kOTNoMemoryConfigurationPtr = 0,
    kOTInvalidConfigurationPtr = -1
};
```

Constants

`kOTNoMemoryConfigurationPtr`

`kOTInvalidConfigurationPtr`

kOTNoMessagesAvailable

```
enum {
    kOTNoMessagesAvailable = 0xFFFFFFFF,
    kOTAnyMsgType = 0xFFFFFFFE,
    kOTDataMsgTypes = 0xFFFFFFFC,
    kOTMProtoMsgTypes = 0xFFFFFFFB,
    kOTOnlyMProtoMsgTypes = 0xFFFFFFFA
};
```

Constants

`kOTNoMessagesAvailable`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`kOTAnyMsgType`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`kOTDataMsgTypes`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`kOTMProtoMsgTypes`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`kOTOnlyMProtoMsgTypes`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

kOTOptionHeaderSize

```
enum {
    kOTOptionHeaderSize = sizeof(TOptionHeader),
    kOTBooleanOptionDataSize = sizeof(UInt32),
    kOTBooleanOptionSize = kOTOptionHeaderSize + kOTBooleanOptionDataSize,
    kOTOneByteOptionSize = kOTOptionHeaderSize + 1,
    kOTTwoByteOptionSize = kOTOptionHeaderSize + 2,
    kOTFourByteOptionSize = kOTOptionHeaderSize + sizeof(UInt32)
};
```

Constants

`kOTOptionHeaderSize`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTBooleanOptionDataSize`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTBooleanOptionSize`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTOneByteOptionSize`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTTwoByteOptionSize`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTFourByteOptionSize`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

kOTPCINoErrorStayLoaded

```
enum {
    kOTPCINoErrorStayLoaded = 1
};
```

Constants

`kOTPCINoErrorStayLoaded`

Port Flags

Specify a port's status.

```
enum {
    kOTPortIsActive = 0x00000001,
    kOTPortIsDisabled = 0x00000002,
    kOTPortIsUnavailable = 0x00000004,
    kOTPortIsOffline = 0x00000008
};
```

Constants

`kOTPortIsActive`
The port is in use.
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTPortIsDisabled`
The port may or may not be in use, but no other client can use it.
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTPortIsUnavailable`
The port is not available for use.
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTPortIsOffline`
The port is off-line. This bit is typically only set when the port is active, the port autoconnects, and it is currently not connected.
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

Port Additional Flags

Specify additional information about a port.

```
enum {
    kOTPortIsDLPI = 0x00000001,
    kOTPortIsTPI = 0x00000002,
    kOTPortCanYield = 0x00000004,
    kOTPortCanArbitrate = 0x00000008,
    kOTPortIsTransitory = 0x00000010,
    kOTPortAutoConnects = 0x00000020,
    kOTPortIsSystemRegistered = 0x00004000,
    kOTPortIsPrivate = 0x00008000,
    kOTPortIsAlias = 0x80000000
};
```

Constants

`kOTPortIsDLPI`
The port driver is a DLPI STREAMS module.
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTPortIsTPI`
The port driver is a TPI STREAMS module.
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTPortCanYield`
The port can yield when requested.
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTPortCanArbitrate`
Reserved.
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTPortIsTransitory`
The port has off-line/on-line status.
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTPortAutoConnects`
The port auto connects. The port goes on-line and off-line on demand. ISDN is a typical example.
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTPortIsSystemRegistered`
The port was registered by the system from the Name Registry
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTPortIsPrivate`
The port is private.
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTPortIsAlias`
The port is an alias for another port.
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

kOTPrintOnly

```
enum {
    kOTPrintOnly = 0,
    kOTPrintThenStop = 1
};
```

Constants

`kOTPrintOnly`

`kOTPrintThenStop`

kOTRawRcvOn

```
enum {
    kOTRawRcvOn = 0,
    kOTRawRcvOff = 1,
    kOTRawRcvOnWithTimeStamp = 2
};
```

Constants

`kOTRawRcvOn`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`kOTRawRcvOff`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`kOTRawRcvOnWithTimeStamp`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

kOTSerialDefaultBaudRate

```
enum {
    kOTSerialDefaultBaudRate = 19200,
    kOTSerialDefaultDataBits = 8,
    kOTSerialDefaultStopBits = 10,
    kOTSerialDefaultParity = kOTSerialNoParity,
    kOTSerialDefaultHandshake = 0,
    kOTSerialDefaultOnChar = ('Q' & 0xFFFFFFBF),
    kOTSerialDefaultOffChar = ('S' & 0xFFFFFFBF),
    kOTSerialDefaultSndBufSize = 1024,
    kOTSerialDefaultRcvBufSize = 1024,
    kOTSerialDefaultSndLoWat = 96,
    kOTSerialDefaultRcvLoWat = 1,
    kOTSerialDefaultRcvTimeout = 10
};
```

Constants

`kOTSerialDefaultBaudRate`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialDefaultDataBits`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialDefaultStopBits`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialDefaultParity`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialDefaultHandshake`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialDefaultOnChar`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialDefaultOffChar`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialDefaultSndBufSize`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialDefaultRcvBufSize`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialDefaultSndLoWat`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialDefaultRcvLoWat`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialDefaultRcvTimeout`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kOTSerialFramingAsync

```
enum {
    kOTSerialFramingAsync = 0x01,
    kOTSerialFramingHDLC = 0x02,
    kOTSerialFramingSDLC = 0x04,
    kOTSerialFramingAsyncPackets = 0x08,
    kOTSerialFramingPPP = 0x10
};
```

Constants

`kOTSerialFramingAsync`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialFramingHDLC`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialFramingSDLC`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialFramingAsyncPackets`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialFramingPPP`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kOTSerialSwOverRunErr

```
enum {
    kOTSerialSwOverRunErr = 0x01,
    kOTSerialBreakOn = 0x08,
    kOTSerialParityErr = 0x10,
    kOTSerialOverrunErr = 0x20,
    kOTSerialFramingErr = 0x40,
    kOTSerialXOffSent = 0x00010000,
    kOTSerialDTRNegated = 0x00020000,
    kOTSerialCTLHold = 0x00040000,
    kOTSerialXOffHold = 0x00080000,
    kOTSerialOutputBreakOn = 0x01000000
};
```

Constants

`kOTSerialSwOverRunErr`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialBreakOn`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialParityErr`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialOverrunErr`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialFramingErr`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialXOffSent`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialDTRNegated`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialCTLHold`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialXOffHold`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialOutputBreakOn`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kOTSerialXOnOffInputHandshake

```
enum {
    kOTSerialXOnOffInputHandshake = 1,
    kOTSerialXOnOffOutputHandshake = 2,
    kOTSerialCTSInputHandshake = 4,
    kOTSerialDTROutputHandshake = 8
};
```

Constants

`kOTSerialXOnOffInputHandshake`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialXOnOffOutputHandshake`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialCTSInputHandshake`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kOTSerialDTROutputHandshake`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kOTSpecificConfigPass

```
enum {
    kOTSpecificConfigPass = 0,
    kOTGenericConfigPass = 1
};
```

Constants

`kOTSpecificConfigPass`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`kOTGenericConfigPass`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

kOTT_BIND_REQ

```
enum {
    kOTT_BIND_REQ = 101,
    kOTT_CONN_REQ = 102,
    kOTT_CONN_RES = 103,
    kOTT_DATA_REQ = 104,
    kOTT_DISCON_REQ = 105,
    kOTT_EXDATA_REQ = 106,
    kOTT_INFO_REQ = 107,
    kOTT_OPTMGMT_REQ = 108,
    kOTT_ORDREL_REQ = 109,
    kOTT_UNBIND_REQ = 110,
    kOTT_UNITDATA_REQ = 111,
    kOTT_ADDR_REQ = 112,
    kOTT_UREQUEST_REQ = 113,
    kOTT_REQUEST_REQ = 114,
    kOTT_UREPLY_REQ = 115,
    kOTT_REPLY_REQ = 116,
    kOTT_CANCELREQUEST_REQ = 117,
    kOTT_CANCELREPLY_REQ = 118,
    kOTT_REGNAME_REQ = 119,
    kOTT_DELNAME_REQ = 120,
    kOTT_LKUPNAME_REQ = 121,
    kOTT_BIND_ACK = 122,
    kOTT_CONN_CON = 123,
    kOTT_CONN_IND = 124,
    kOTT_DATA_IND = 125,
    kOTT_DISCON_IND = 126,
    kOTT_ERROR_ACK = 127,
    kOTT_EXDATA_IND = 128,
    kOTT_INFO_ACK = 129,
    kOTT_OK_ACK = 130,
    kOTT_OPTMGMT_ACK = 131,
    kOTT_ORDREL_IND = 132,
    kOTT_UNITDATA_IND = 133,
    kOTT_UDERROR_IND = 134,
    kOTT_ADDR_ACK = 135,
    kOTT_UREQUEST_IND = 136,
    kOTT_REQUEST_IND = 137,
    kOTT_UREPLY_IND = 138,
    kOTT_REPLY_IND = 139,
    kOTT_UREPLY_ACK = 140,
    kOTT_REPLY_ACK = 141,
    kOTT_RESOLVEADDR_REQ = 142,
    kOTT_RESOLVEADDR_ACK = 143,
    kOTT_LKUPNAME_CON = 146,
    kOTT_LKUPNAME_RES = 147,
    kOTT_REGNAME_ACK = 148,
    kOTT_SEQUENCED_ACK = 149,
    kOTT_EVENT_IND = 160
};
```

Constants

kOTT_BIND_REQ
kOTT_CONN_REQ
kOTT_CONN_RES
kOTT_DATA_REQ

```
kOTT_DISCON_REQ
kOTT_EXDATA_REQ
kOTT_INFO_REQ
kOTT_OPTMGMT_REQ
kOTT_ORDREL_REQ
kOTT_UNBIND_REQ
kOTT_UNITDATA_REQ
kOTT_ADDR_REQ
kOTT_UREQUEST_REQ
kOTT_REQUEST_REQ
kOTT_UREPLY_REQ
kOTT_REPLY_REQ
kOTT_CANCELREQUEST_REQ
kOTT_CANCELREPLY_REQ
kOTT_REGNAME_REQ
kOTT_DELNAME_REQ
kOTT_LKUPNAME_REQ
kOTT_BIND_ACK
kOTT_CONN_CON
kOTT_CONN_IND
kOTT_DATA_IND
kOTT_DISCON_IND
kOTT_ERROR_ACK
kOTT_EXDATA_IND
kOTT_INFO_ACK
kOTT_OK_ACK
kOTT_OPTMGMT_ACK
kOTT_ORDREL_IND
kOTT_UNITDATA_IND
kOTT_UDERROR_IND
kOTT_ADDR_ACK
kOTT_UREQUEST_IND
kOTT_REQUEST_IND
kOTT_UREPLY_IND
kOTT_REPLY_IND
kOTT_UREPLY_ACK
kOTT_REPLY_ACK
kOTT_RESOLVEADDR_REQ
kOTT_RESOLVEADDR_ACK
kOTT_LKUPNAME_CON
kOTT_LKUPNAME_RES
kOTT_REGNAME_ACK
kOTT_SEQUENCED_ACK
kOTT_EVENT_IND
```

kOTT_TIMER_REQ

```
enum {
    kOTT_TIMER_REQ = 80,
    kOTT_MIB_REQ = 81,
    kOTT_MIB_ACK = 82,
    kOTT_PRIVATE_REQ = 90
```

```
};
```

Constants
kOTT_TIMER_REQ
kOTT_MIB_REQ
kOTT_MIB_ACK
kOTT_PRIVATE_REQ

kOTTRANSPARENT

```
enum {
    kOTTRANSPARENT = 0xFFFFFFFF
};
```

Constants
kOTTRANSPARENT

kPPPAsyncMapCharsNone

```
enum {
    kPPPAsyncMapCharsNone = 0x00000000,
    kPPPAsyncMapCharsXOnXOff = 0x000A0000,
    kPPPAsyncMapCharsAll = 0xFFFFFFFF
};
```

Constants
kPPPAsyncMapCharsNone
Available in Mac OS X v10.0 and later.
Declared in OpenTransportProviders.h.

kPPPAsyncMapCharsXOnXOff
Available in Mac OS X v10.0 and later.
Declared in OpenTransportProviders.h.

kPPPAsyncMapCharsAll
Available in Mac OS X v10.0 and later.
Declared in OpenTransportProviders.h.

kPPPCompressionDisabled

```
enum {
    kPPPCompressionDisabled = 0x00000000,
    kPPPProtoCompression = 0x00000001,
    kPPPAddrCompression = 0x00000002
};
```

Constants
kPPPCompressionDisabled
Available in Mac OS X v10.0 and later.
Declared in OpenTransportProviders.h.

`kPPPProtoCompression`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`kPPPAddrCompression`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

kPPPConnectionStatusDialogsFlag

```
enum {
    kPPPConnectionStatusDialogsFlag = 0x00000001,
    kPPPConnectionRemindersFlag = 0x00000002,
    kPPPConnectionFlashingIconFlag = 0x00000004,
    kPPPOutPasswordDialogsFlag = 0x00000008,
    kPPPAllAlertsDisabledFlag = 0x00000000,
    kPPPAllAlertsEnabledFlag = 0x0000000F
};
```

Constants

`kPPPConnectionStatusDialogsFlag`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`kPPPConnectionRemindersFlag`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`kPPPConnectionFlashingIconFlag`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`kPPPOutPasswordDialogsFlag`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`kPPPAllAlertsDisabledFlag`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`kPPPAllAlertsEnabledFlag`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

kPPPConnectionStatusIdle

```
enum {
    kPPPConnectionStatusIdle = 1,
    kPPPConnectionStatusConnecting = 2,
    kPPPConnectionStatusConnected = 3,
    kPPPConnectionStatusDisconnecting = 4
};
```

Constants

`kPPPConnectionStatusIdle`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kPPPConnectionStatusConnecting`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kPPPConnectionStatusConnected`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kPPPConnectionStatusDisconnecting`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kPPPEvent

```
enum {
    kPPPEvent = kPROTOCOLEVENT | 0x000F0000,
    kPPPConnectCompleteEvent = kPPPEvent + 1,
    kPPPSetScriptCompleteEvent = kPPPEvent + 2,
    kPPPDisconnectCompleteEvent = kPPPEvent + 3,
    kPPPDisconnectEvent = kPPPEvent + 4,
    kPPPIPCPUpEvent = kPPPEvent + 5,
    kPPPIPCPDownEvent = kPPPEvent + 6,
    kPPPLCPUpEvent = kPPPEvent + 7,
    kPPPLCPDownEvent = kPPPEvent + 8,
    kPPPLowerLayerUpEvent = kPPPEvent + 9,
    kPPPLowerLayerDownEvent = kPPPEvent + 10,
    kPPPAuthenticationStartedEvent = kPPPEvent + 11,
    kPPPAuthenticationFinishedEvent = kPPPEvent + 12,
    kPPPDCEInitStartedEvent = kPPPEvent + 13,
    kPPPDCEInitFinishedEvent = kPPPEvent + 14,
    kPPPDCECallStartedEvent = kPPPEvent + 15,
    kPPPDCECallFinishedEvent = kPPPEvent + 16
};
```

Constants

`kPPPEvent`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kPPPConnectCompleteEvent`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kPPPSetScriptCompleteEvent`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kPPPDisconnectCompleteEvent`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kPPPDisconnectEvent`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kPPPIPCPUpEvent`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kPPPIPCPDownEvent`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kPPPLCPUpEvent`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kPPPLCPDownEvent`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kPPPLowerLayerUpEvent`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kPPPLowerLayerDownEvent`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kPPPAuthenticationStartedEvent`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kPPPAuthenticationFinishedEvent`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kPPPDCEInitStartedEvent`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kPPPDCEInitFinishedEvent`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kPPPDCECallStartedEvent`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kPPPDCECallFinishedEvent`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kPPPMaxIDLength

```
enum {
    kPPPMaxIDLength = 255,
    kPPPMaxPasswordLength = 255,
    kPPPMaxDTEAddressLength = 127,
    kPPPMaxCallInfoLength = 255
};
```

Constants

`kPPPMaxIDLength`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kPPPMaxPasswordLength`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kPPPMaxDTEAddressLength`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kPPPMaxCallInfoLength`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kPPPMinMRU

```
enum {
    kPPPMinMRU = 0,
    kPPPMaxMRU = 4500
};
```

Constants

`kPPPMinMRU`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kPPPMaxMRU`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kPPPNoOutAuthentication

```
enum {
    kPPPNoOutAuthentication = 0,
    kPPPCHAPOrPAPOutAuthentication = 1
};
```

Constants

`kPPPNoOutAuthentication`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kPPPCHAPOrPAPOutAuthentication`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kPPPScriptTypeModem

```
enum {
    kPPPScriptTypeModem = 1,
    kPPPScriptTypeConnect = 2,
    kPPPMaxScriptSize = 32000
};
```

Constants

`kPPPScriptTypeModem`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kPPPScriptTypeConnect`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kPPPMaxScriptSize`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kPPPStateInitial

```
enum {
    kPPPStateInitial = 1,
    kPPPStateClosed = 2,
    kPPPStateClosing = 3,
    kPPPStateOpening = 4,
    kPPPStateOpened = 5
};
```

Constants

`kPPPStateInitial`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kPPPStateClosed`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kPPPStateClosing`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kPPPStateOpening`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kPPPStateOpened`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kRAProductClientOnly

```
enum {
    kRAProductClientOnly = 2,
    kRAProductOnePortServer = 3,
    kRAProductManyPortServer = 4
};
```

Constants

`kRAProductClientOnly`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kRAProductOnePortServer`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kRAProductManyPortServer`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kSAP_ONE

```
enum {
    kSAP_ONE = 1,
    kSAP_RANGE = 2,
    kSAP_ALL = 3,
    kSAP_CLEAR = 4
};
```

Constants

`kSAP_ONE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`kSAP_RANGE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`kSAP_ALL`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`kSAP_CLEAR`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

kSerialABModuleID

```
enum {
    kSerialABModuleID = 7200
};
```

Constants

`kSerialABModuleID`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kSIGHUP

```
enum {
    kSIGHUP = 1,
    kSIGURG = 16,
    kSIGPOLL = 30
};
```

Constants

`kSIGHUP`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kSIGURG`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kSIGPOLL`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

kT_UNSPEC

```
enum {
    kT_UNSPEC = 0xFFFFFFFD,
    T_ALLOPT = 0
};
```

Constants

`kT_UNSPEC`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_ALLOPT`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

kT8022HeaderLength

```
enum {
    kT8022HeaderLength = 3,
    kT8022SNAPHeaderLength = 3 + k8022SNAPLength,
   kT8022FullPacketHeaderLength = kEnetPacketHeaderLength + kT8022SNAPHeaderLength
};
```

Constants

`kT8022HeaderLength`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`kT8022SNAPHeaderLength`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`kT8022FullPacketHeaderLength`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

kT8022ModuleID

```
enum {
    kT8022ModuleID = 7100,
    kEnetModuleID = 7101,
    kTokenRingModuleID = 7102,
    kFDDIModuleID = 7103
};
```

Constants

`kT8022ModuleID`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`kEnetModuleID`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kTokenRingModuleID`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kFDDIModuleID`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

kZIPMaxZoneLength

```
enum {
    kZIPMaxZoneLength = kNBPMaxZoneLength
};
```

Constants

`kZIPMaxZoneLength`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

LNK_ENET

```
enum {
    LNK_ENET = 'ENET',
    LNK_TOKN = 'TOKN',
    LNK_FDDI = 'FDDI',
    LNK_TPI = 'LTPI'
};
```

Constants

`LNK_ENET`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`LNK_TOKN`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`LNK_FDDI`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`LNK_TPI`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

LOGMSGSZ

```
enum {
    LOGMSGSZ = 128
};
```

Constants

`LOGMSGSZ`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

M_MI

```
enum {
    M_MI = 0x40,
    M_MI_READ_RESET = 1,
    M_MI_READ_SEEK = 2,
    M_MI_READ_END = 4
};
```

Constants

`M_MI`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`M_MI_READ_RESET`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`M_MI_READ_SEEK`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`M_MI_READ_END`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

MIOC_ISDN

```
enum {
    MIOC_ISDN = 85
};
```

Constants

`MIOC_ISDN`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

MIOC_STREAMIO

```
enum {
    MIOC_STREAMIO = 65,
    MIOC_TMOD = 'a',
    MIOC_STRLOG = 'b',
    MIOC_ND = 'c',
    MIOC_ECHO = 'd',
    MIOC_TLI = 'e',
    MIOC_RESERVEDf = 'f',
    MIOC_SAD = 'g',
    MIOC_ARP = 'h',
    MIOC_HAVOC = 72,
    MIOC_RESERVEDi = 'i',
    MIOC_SIOC = 'j',
    MIOC_TCP = 'k',
    MIOC_DLPI = 'l',
    MIOC_SOCKETS = 'm',
    MIOC_IPX = 'o',
    MIOC_OT = 79,
    MIOC_ATALK = 84,
    MIOC_SRL = 85,
    MIOC_RESERVEDp = 'p',
    MIOC_RESERVEDr = 'r',
    MIOC_RESERVEDs = 's',
    MIOC_CFIG = 'z'
};
```

Constants

`MIOC_STREAMIO`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`MIOC_TMOD`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`MIOC_STRLOG`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`MIOC_ND`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`MIOC_ECHO`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`MIOC_TLI`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`MIOC_RESERVEDf`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`MIOC_SAD`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`MIOC_ARP`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`MIOC_HAVOC`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`MIOC_RESERVEDi`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`MIOC_SIOC`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`MIOC_TCP`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`MIOC_DLPI`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`MIOC_SOCKETS`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`MIOC_IPX`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`MIOC_OT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`MIOC_ATALK`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`MIOC_SRL`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`MIOC_RESERVEDp`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`MIOC_RESERVEDr`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`MIOC_RESERVEDs`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`MIOC_CFIG`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

MORECTL

```
enum {
    MORECTL = 0x01,
    MOREDATA = 0x02
};
```

Constants

`MORECTL`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`MOREDATA`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

MSG_HIPRI

```
enum {
    MSG_HIPRI = 0x01,
    MSG_BAND = 0x02,
    MSG_ANY = 0x04
};
```

Constants

`MSG_HIPRI`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`MSG_BAND`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`MSG_ANY`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

MSGMARK

```
enum {
    MSGMARK = 0x01,
    MSGNOLOOP = 0x02,
    MSGDELIM = 0x04,
    MSGNOGET = 0x08
};
```

Constants

`MSGMARK`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`MSGNOLOOP`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`MSGDELIM`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`MSGNOGET`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

MUXID_ALL

```
enum {
    MUXID_ALL = -1
};
```

Constants

`MUXID_ALL`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

NOERROR

```
enum {
    NOERROR = -1
};
```

Constants

`NOERROR`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

O_ASYNC

```
enum {
    O_ASYNC = kO_ASYNC,
    O_NDELAY = kO_NDELAY,
    O_NONBLOCK = kO_NONBLOCK
};
```

Constants

`O_ASYNC`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`O_NDELAY`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`O_NONBLOCK`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

OPT_ADDMCAST

```
enum {
    OPT_ADDMCAST = 0x1000,
    OPT_DELMCAST = 0x1001,
    OPT_RCVPACKETTYPE = 0x1002,
    OPT_RCVDESTADDR = 0x1003,
    OPT_SETRAWMODE = 0x1004,
    OPT_SETPROMISCUOUS = 0x1005
};
```

Constants

`OPT_ADDMCAST`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`OPT_DELMCAST`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`OPT_RCVPACKETTYPE`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`OPT_RCVDESTADDR`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`OPT_SETRAWMODE`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`OPT_SETPROMISCUOUS`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

Bus Type Constants

Specify the bus type of a port.

```
typedef UInt8 OTBusType;
enum {
    kOTUnknownBusPort = 0,
    kOTMotherboardBus = 1,
    kOTNuBus = 2,
    kOTPCIBus = 3,
    kOTGeoPort = 4,
    kOTPCCardBus = 5,
    kOTFireWireBus = 6,
    kOTLastBusIndex = 15
};
```

Constants

`kOTUnknownBusPort`
The port's bus type is not a known type.
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTMotherboardBus`
The port is on the motherboard.
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTNuBus`
The port is on a NuBus.
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTPCIBus`
The port is on a PCI bus.
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTGeoPort`
The port is a GeoPort device.
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTPCCardBus`
The port is on a PCCard bus.
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTFireWireBus`
The port is on a Firewire bus.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTLastBusIndex`
The maximum bus type that the port can support.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

Hardware Device Types

Lists hardware device types for Open Transport ports.

```
typedef UInt16 OTDeviceType;
enum {
    kOTNoDeviceType = 0,
    kOTADEVDevice = 1,
    kOTMDEVDevice = 2,
    kOTLocalTalkDevice = 3,
    kOTIRTalkDevice = 4,
    kOTTokenRingDevice = 5,
    kOTISDNDevice = 6,
    kOTATMDevice = 7,
    kOTSMDSDevice = 8,
    kOTSerialDevice = 9,
    kOTEthernetDevice = 10,
    kOTSLIPDevice = 11,
    kOTPPPDevice = 12,
    kOTModemDevice = 13,
    kOTFastEthernetDevice = 14,
    kOTFDDIDevice = 15,
    kOTIrDADevice = 16,
    kOTATMSNAPDevice = 17,
    kOTFibreChannelDevice = 18,
    kOTFireWireDevice = 19,
    kOTPseudoDevice = 1023,
    kOTLastDeviceIndex = 1022
};
```

Constants

`kOTNoDeviceType`
The port's device type is not specified. This value is illegal.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTADEVDevice`
The port is specified as an 'adev' device, which is a pseudodevice used by AppleTalk.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTMDEVDevice`
The port is specified as an 'mdev' device, which is a pseudodevice used by TCP.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTLocalTalkDevice`
The port is specified as a LocalTalk device.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTIRTalkDevice`
The port is specified as an IRTalk device.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTTokenRingDevice`
The port is specified as a token ring device.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTISDNDevice`
The port is specified as an ISDN device.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTATMDevice`
The port is specified as an ATM device.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTSMDSDevice`
The port is specified as a SMDS device.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTSerialDevice`
The port is specified as a serial device.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTEthernetDevice`
The port is specified as an Ethernet device.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTSLIPDevice`
The port is specified as a SLIP pseudodevice.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTPPPDevice`
The port is specified as a SLIP pseudodevice.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTModemDevice`
The port is specified as a modem pseudodevice.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTFastEthernetDevice`
The port is specified as an 100 MB Ethernet device.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTFDDIDevice`
The port is specified as a FDDI device.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTIrDADevice`
The port is specified as an IrDA Infrared device.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTATMSNAPDevice`
The port is specified as an ATM pseudodevice simulating a SNAP device.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTFibreChannelDevice`
The port is specified as a Fibre Channel device.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTFireWireDevice`
The port is specified as a Firewire device.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTPseudoDevice`
The port is designated as a pseudodevice.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTLastDeviceIndex`
The maximum device types that a port can use.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

Special Considerations

Do not arbitrarily add new device types. Please contact Developer Support at Apple Computer, Inc. to obtain a new, unique device type.

OTInitializationFlags

```
typedef UInt32 OTInitializationFlags;
enum {
    kInitOTForApplicationMask = 1,
    kInitOTForExtensionMask = 2
};
```

Constants

`kInitOTForApplicationMask`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kInitOTForExtensionMask`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

OTOpenFlags

```
typedef UInt32 OTOpenFlags;
enum {
    kO_ASYNC = 0x01,
    kO_NDELAY = 0x04,
    kO_NONBLOCK = 0x04
};
```

Constants

`kO_ASYNC`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kO_NDELAY`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kO_NONBLOCK`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

OTPacketType

```
typedef UInt32 OTPacketType;
enum {
    kETypeStandard = 0,
    kETypeMulticast = 1,
    kETypeBroadcast = 2,
    kETRawPacketBit = 0x80000000,
    kETTimeStampBit = 0x40000000
};
```

Constants

`kETypeStandard`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kETypeMulticast`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kETypeBroadcast`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kETRawPacketBit`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`kETTimeStampBit`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

Endpoint Service Types

Contains values that Open Transport can return in the `servtype` field of the `TEndpointInfo` structure.

```
typedef UInt32 OTServiceType;
enum {
    T_COTS = 1,
    T_COTS_ORD = 2,
    T_CLTS = 3,
    T_TRANS = 5,
    T_TRANS_ORD = 6,
    T_TRANS_CLTS = 7
};
```

Constants

`T_COTS`
Connection-oriented transactionless service without orderly release.
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`T_COTS_ORD`
: Connection-oriented transactionless service with optional orderly release.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_CLTS`
: Connectionless transactionless service.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_TRANS`
: Connection-oriented transaction-based service without orderly release.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_TRANS_ORD`
: Connection-oriented transaction-based service with optional orderly release.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_TRANS_CLTS`
: Connectionless transaction-based service.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

Endpoint States

Define enpoint states for the `OTGetEndpointState` function.

```
typedef UInt32 OTXTIStates;
enum {
    T_UNINIT = 0,
    T_UNBND = 1,
    T_IDLE = 2,
    T_OUTCON = 3,
    T_INCON = 4,
    T_DATAXFER = 5,
    T_OUTREL = 6,
    T_INREL = 7
};
```

Constants

`T_UNINIT`
: This endpoint has been closed and destroyed.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_UNBND`
: This endpoint is initialized but has not yet been bound to an address.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_IDLE`
This endpoint has been bound to an address and is ready for use: connectionless endpoints can send or receive data; connection-oriented endpoints can initiate or listen for a connection.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_OUTCON`
This endpoint has initiated a connection and is waiting for the peer endpoint to accept the connection.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_INCON`
This endpoint has received a connection request but has not yet accepted or rejected the request.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_DATAXFER`
This connection-oriented endpoint can now transfer data because the connection has been established.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_OUTREL`
This endpoint has issued an orderly disconnect that the peer has not acknowledged. The endpoint can continue to read data, but must not send any more data.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_INREL`
This endpoint has received a request for an orderly disconnect, which it has not yet acknowledged. The endpoint can continue to send data until it acknowledges the disconnection request, but it must not read data.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

ParityOptionValues

```
typedef UInt32 ParityOptionValues;
enum {
    kOTSerialNoParity = 0,
    kOTSerialOddParity = 1,
    kOTSerialEvenParity = 2
};
```

Constants

`kOTSerialNoParity`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`kOTSerialOddParity`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`kOTSerialEvenParity`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

QB_FULL

```
enum {
    QB_FULL = 0x01,
    QB_WANTW = 0x02,
    QB_BACK = 0x04
};
```

Constants

`QB_FULL`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`QB_WANTW`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`QB_BACK`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

qfields

```
typedef SInt32 qfields;
enum {
    QHIWAT = 0,
    QLOWAT = 1,
    QMAXPSZ = 2,
    QMINPSZ = 3,
    QCOUNT = 4,
    QFIRST = 5,
    QLAST = 6,
    QFLAG = 7,
    QBAD = 8
};
```

Constants

`QHIWAT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`QLOWAT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`QMAXPSZ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`QMINPSZ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`QCOUNT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`QFIRST`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`QLAST`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`QFLAG`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`QBAD`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

QNORM

```
enum {
    QNORM = 0,
    M_DATA = 0,
    M_PROTO = 1,
    M_BREAK = 0x08,
    M_PASSFP = 0x09,
    M_SIG = 0x0B,
    M_DELAY = 0x0C,
    M_CTL = 0x0D,
    M_IOCTL = 0x0E,
    M_SETOPTS = 0x10,
    M_RSE = 0x11
};
```

Constants

`QNORM`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`M_DATA`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`M_PROTO`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`M_BREAK`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`M_PASSFP`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`M_SIG`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`M_DELAY`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`M_CTL`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`M_IOCTL`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`M_SETOPTS`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`M_RSE`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

QPCTL

```
enum {
    QPCTL = 0x80,
    M_IOCACK = 0x81,
    M_IOCNAK = 0x82,
    M_PCPROTO = 0x83,
    M_PCSIG = 0x84,
    M_FLUSH = 0x86,
    M_STOP = 0x87,
    M_START = 0x88,
    M_HANGUP = 0x89,
    M_ERROR = 0x8A,
    M_READ = 0x8B,
    M_COPYIN = 0x8C,
    M_COPYOUT = 0x8D,
    M_IOCDATA = 0x8E,
    M_PCRSE = 0x90,
    M_STOPI = 0x91,
    M_STARTI = 0x92,
    M_HPDATA = 0x93
};
```

Constants

`QPCTL`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`M_IOCACK`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`M_IOCNAK`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`M_PCPROTO`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`M_PCSIG`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`M_FLUSH`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`M_STOP`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`M_START`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`M_HANGUP`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`M_ERROR`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`M_READ`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`M_COPYIN`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`M_COPYOUT`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`M_IOCDATA`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`M_PCRSE`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`M_STOPI`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`M_STARTI`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`M_HPDATA`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

QREADR

```
enum {
    QREADR = 0x01,
    QNOENB = 0x02,
    QFULL = 0x04,
    QWANTR = 0x08,
    QWANTW = 0x10,
    QUSE = 0x20,
    QENAB = 0x40,
    QBACK = 0x80,
    QOLD = 0x0100,
    QHLIST = 0x0200,
    QWELDED = 0x0400,
    QUNWELDING = 0x0800,
    QPROTECTED = 0x1000,
    QEXCOPENCLOSE = 0x2000
};
```

Constants

`QREADR`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`QNOENB`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`QFULL`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`QWANTR`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`QWANTW`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`QUSE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`QENAB`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`QBACK`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`QOLD`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`QHLIST`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`QWELDED`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`QUNWELDING`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`QPROTECTED`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`QEXCOPENCLOSE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

RNORM

```
enum {
    RNORM = 0x01,
    RMSGD = 0x02,
    RMSGN = 0x04,
    RFILL = 0x08
};
```

Constants

`RNORM`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`RMSGD`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`RMSGN`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`RFILL`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

RPROTNORM

```
enum {
    RPROTNORM = 0x10,
    RPROTDIS = 0x20,
    RPROTDAT = 0x40
};
```

Constants

`RPROTNORM`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`RPROTDIS`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`RPROTDAT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

RS_EXDATA

```
enum {
    RS_EXDATA = 0x20,
    RS_ALLOWAGAIN = 0x40,
    RS_DELIMITMSG = 0x80
};
```

Constants

`RS_EXDATA`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`RS_ALLOWAGAIN`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`RS_DELIMITMSG`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

RS_HIPRI

```
enum {
    RS_HIPRI = 0x01
};
```

Constants

`RS_HIPRI`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

S_INPUT

```
enum {
    S_INPUT = 0x01,
    S_HIPRI = 0x02,
    S_OUTPUT = 0x04,
    S_MSG = 0x08,
    S_RDNORM = 0x10,
    S_RDBAND = 0x20,
    S_WRNORM = 0x40,
    S_WRBAND = 0x80,
    S_ERROR = 0x0100,
    S_HANGUP = 0x0200,
    S_BANDURG = 0x0400
};
```

Constants

`S_INPUT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`S_HIPRI`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`S_OUTPUT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`S_MSG`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`S_RDNORM`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`S_RDBAND`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`S_WRNORM`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`S_WRBAND`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`S_ERROR`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`S_HANGUP`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

S_BANDURG
Available in Mac OS X v10.0 and later.
Declared in OpenTransportProtocol.h.

SENDZERO

```
enum {
    SENDZERO = 0x0001,
    XPG4_1 = 0x0002
};
```

Constants

SENDZERO
Available in Mac OS X v10.0 and later.
Declared in OpenTransportProtocol.h.

XPG4_1
Available in Mac OS X v10.0 and later.
Declared in OpenTransportProtocol.h.

SERIAL_OPT_BAUDRATE

```
enum {
    SERIAL_OPT_BAUDRATE = 0x0100,
    SERIAL_OPT_DATABITS = 0x0101,
    SERIAL_OPT_STOPBITS = 0x0102,
    SERIAL_OPT_PARITY = 0x0103,
    SERIAL_OPT_STATUS = 0x0104,
    SERIAL_OPT_HANDSHAKE = 0x0105,
    SERIAL_OPT_RCVTIMEOUT = 0x0106,
    SERIAL_OPT_ERRORCHARACTER = 0x0107,
    SERIAL_OPT_EXTCLOCK = 0x0108,
    SERIAL_OPT_BURSTMODE = 0x0109,
    SERIAL_OPT_DUMMY = 0x010A
};
```

Constants

SERIAL_OPT_BAUDRATE
Available in Mac OS X v10.0 and later.
Declared in OpenTransportProviders.h.

SERIAL_OPT_DATABITS
Available in Mac OS X v10.0 and later.
Declared in OpenTransportProviders.h.

SERIAL_OPT_STOPBITS
Available in Mac OS X v10.0 and later.
Declared in OpenTransportProviders.h.

SERIAL_OPT_PARITY
Available in Mac OS X v10.0 and later.
Declared in OpenTransportProviders.h.

`SERIAL_OPT_STATUS`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`SERIAL_OPT_HANDSHAKE`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`SERIAL_OPT_RCVTIMEOUT`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`SERIAL_OPT_ERRORCHARACTER`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`SERIAL_OPT_EXTCLOCK`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`SERIAL_OPT_BURSTMODE`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`SERIAL_OPT_DUMMY`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

SIGHUP

```
enum {
    SIGHUP = 1,
    SIGURG = 16,
    SIGPOLL = 30
};
```

Constants

`SIGHUP`

`SIGURG`

`SIGPOLL`

SL_FATAL

```
enum {
    SL_FATAL = 0x01,
    SL_NOTIFY = 0x02,
    SL_ERROR = 0x04,
    SL_TRACE = 0x08,
    SL_CONSOLE = 0x00,
    SL_WARN = 0x20,
    SL_NOTE = 0x40
};
```

Constants

`SL_FATAL`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`SL_NOTIFY`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`SL_ERROR`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`SL_TRACE`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`SL_CONSOLE`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`SL_WARN`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`SL_NOTE`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

SNDZERO

```
enum {
    SNDZERO = 0x01
};
```

Constants

SNDZERO

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

SO_ALL

```
enum {
    SO_ALL = 0x7FFF,
    SO_READOPT = 0x0001,
    SO_WROFF = 0x0002,
    SO_MINPSZ = 0x0004,
    SO_MAXPSZ = 0x0008,
    SO_HIWAT = 0x0010,
    SO_LOWAT = 0x0020,
    SO_MREADON = 0x0040,
    SO_MREADOFF = 0x0080,
    SO_NDELON = 0x0100,
    SO_NDELOFF = 0x0200,
    SO_ISTTY = 0x0400,
    SO_ISNTTY = 0x0800,
    SO_TOSTOP = 0x1000,
    SO_TONSTOP = 0x2000,
    SO_BAND = 0x4000,
    SO_POLL_SET = 0x8000,
    SO_POLL_CLR = 0x00010000
};
```

Constants

SO_ALL

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

SO_READOPT

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

SO_WROFF

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

SO_MINPSZ

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

SO_MAXPSZ

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`SO_HIWAT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`SO_LOWAT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`SO_MREADON`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`SO_MREADOFF`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`SO_NDELON`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`SO_NDELOFF`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`SO_ISTTY`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`SO_ISNTTY`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`SO_TOSTOP`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`SO_TONSTOP`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`SO_BAND`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`SO_POLL_SET`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`SO_POLL_CLR`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

SQLVL_QUEUE

```
enum {
    SQLVL_QUEUE = 1,
    SQLVL_QUEUEPAIR = 2,
    SQLVL_MODULE = 3,
    SQLVL_GLOBAL = 4,
    SQLVL_DEFAULT = 3
};
```

Constants

`SQLVL_QUEUE`

`SQLVL_QUEUEPAIR`

`SQLVL_MODULE`

`SQLVL_GLOBAL`

`SQLVL_DEFAULT`

STRCANON

```
enum {
    STRCANON = 0x01,
    RECOPY = 0x02
};
```

Constants

`STRCANON`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`RECOPY`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

STRCTLSZ

```
enum {
    STRCTLSZ = 256,
    STRMSGSZ = 8192
};
```

Constants

`STRCTLSZ`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`STRMSGSZ`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

T_ADDR

```
enum {
    T_ADDR = 0x01,
    T_OPT = 0x02,
    T_UDATA = 0x04,
    T_ALL = 0xFFFF
};
typedef UInt32 OTFieldsType;
```

Constants

`T_ADDR`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_OPT`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_UDATA`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_ALL`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

T_ATALKBADROUTEREVENT

```
enum {
    T_ATALKBADROUTEREVENT = kAppleTalkEvent + 70,
    T_ALLNODESTAKENEVENT = kAppleTalkEvent + 71,
    T_FIXEDNODETAKENEVENT = kAppleTalkEvent + 72,
    T_MPPCOMPATCFIGEVENT = kAppleTalkEvent + 73,
    T_FIXEDNODEBADEVENT = kAppleTalkEvent + 74
};
```

Constants

`T_ATALKBADROUTEREVENT`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`T_ALLNODESTAKENEVENT`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`T_FIXEDNODETAKENEVENT`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`T_MPPCOMPATCFIGEVENT`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`T_FIXEDNODEBADEVENT`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

Structure Types

Specify the various Open Transport structure types that can be allocated by the `OTAllocInContext` function.

```
enum {
    T_BIND = 1,
    T_OPTMGMT = 2,
    T_CALL = 3,
    T_DIS = 4,
    T_UNITDATA = 5,
    T_UDERROR = 6,
    T_INFO = 7,
    T_REPLYDATA = 8,
    T_REQUESTDATA = 9,
    T_UNITREQUEST = 10,
    T_UNITREPLY = 11
};
typedef UInt32 OTStructType;
```

Constants

`T_BIND`
Specifies the `TBind` (page 172) structure.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_OPTMGMT`
Specifies the `The Option Management Structure` (page 181) structure.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_CALL`
Specifies the `TCall` (page 173) structure

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_DIS`
Specifies the `TDiscon` (page 174) structure.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_UNITDATA`
Specifies the `TUnitData` (page 185) structure.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_UDERROR`
Specifies the `TUDErr` (page 184) structure.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_INFO`
Speciifies the `TEndpointInfo` (page 174) structure.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_REPLYDATA`
Specifies the `TReply` (page 184) structure.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_REQUESTDATA`
Specifies the `TRequest` (page 184) structure.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_UNITREQUEST`
Specifies the `TUnitRequest` (page 187) structure.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_UNITREPLY`
Specifies the `TUnitReply` (page 186) structure.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

T_DNRSTRINGTOADDRCOMPLETE

```
enum {
    T_DNRSTRINGTOADDRCOMPLETE = kPRIVATEEVENT + 1,
    T_DNRADDRTONAMECOMPLETE = kPRIVATEEVENT + 2,
    T_DNRSYSINFOCOMPLETE = kPRIVATEEVENT + 3,
    T_DNRMAILEXCHANGECOMPLETE = kPRIVATEEVENT + 4,
    T_DNRQUERYCOMPLETE = kPRIVATEEVENT + 5
};
```

Constants

`T_DNRSTRINGTOADDRCOMPLETE`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`T_DNRADDRTONAMECOMPLETE`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`T_DNRSYSINFOCOMPLETE`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`T_DNRMAILEXCHANGECOMPLETE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`T_DNRQUERYCOMPLETE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

T_GARBAGE

```
enum {
    T_GARBAGE = 2
};
```

Constants
`T_GARBAGE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

T_INFINITE

```
enum {
    T_INFINITE = -1,
    T_INVALID = -2
};
```

Constants
`T_INFINITE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`T_INVALID`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

Event Codes

Define the constant names that provider functions can use for event codes, or define port-related events Open Transport can send to an client application.

```
typedef UInt32 OTEventCode;
enum {
    T_LISTEN = 0x0001,
    T_CONNECT = 0x0002,
    T_DATA = 0x0004,
    T_EXDATA = 0x0008,
    T_DISCONNECT = 0x0010,
    T_ERROR = 0x0020,
    T_UDERR = 0x0040,
    T_ORDREL = 0x0080,
    T_GODATA = 0x0100,
    T_GOEXDATA = 0x0200,
    T_REQUEST = 0x0400,
    T_REPLY = 0x0800,
    T_PASSCON = 0x1000,
    T_RESET = 0x2000,
    kPRIVATEEVENT = 0x10000000,
    kCOMPLETEEVENT = 0x20000000,
    T_BINDCOMPLETE = 0x20000001,
    T_UNBINDCOMPLETE = 0x20000002,
    T_ACCEPTCOMPLETE = 0x20000003,
    T_REPLYCOMPLETE = 0x20000004,
    T_DISCONNECTCOMPLETE = 0x20000005,
    T_OPTMGMTCOMPLETE = 0x20000006,
    T_OPENCOMPLETE = 0x20000007,
    T_GETPROTADDRCOMPLETE = 0x20000008,
    T_RESOLVEADDRCOMPLETE = 0x20000009,
    T_GETINFOCOMPLETE = 0x2000000A,
    T_SYNCCOMPLETE = 0x2000000B,
    T_MEMORYRELEASED = 0x2000000C,
    T_REGNAMECOMPLETE = 0x2000000D,
    T_DELNAMECOMPLETE = 0x2000000E,
    T_LKUPNAMECOMPLETE = 0x2000000F,
    T_LKUPNAMERESULT = 0x20000010,
    kOTSyncIdleEvent = 0x20000011,
    kSTREAMEVENT = 0x21000000,
    kOTReservedEvent1 = 0x21000001,
    kGetmsgEvent = 0x21000002,
    kStreamReadEvent = 0x21000003,
    kStreamWriteEvent = 0x21000004,
    kStreamIoctlEvent = 0x21000005,
    kOTReservedEvent2 = 0x21000006,
    kStreamOpenEvent = 0x21000007,
    kPollEvent = 0x21000008,
    kOTReservedEvent3 = 0x21000009,
    kOTReservedEvent4 = 0x2100000A,
    kOTReservedEvent5 = 0x2100000B,
    kOTReservedEvent6 = 0x2100000C,
    kOTReservedEvent7 = 0x2100000D,
    kOTReservedEvent8 = 0x2100000E,
    kSIGNALEVENT = 0x22000000,
    kPROTOCOLEVENT = 0x23000000,
    kOTProviderIsDisconnected = 0x23000001,
    kOTProviderIsReconnected = 0x23000002,
    kOTProviderWillClose = 0x24000001,
    kOTProviderIsClosed = 0x24000002,
    kOTPortDisabled = 0x25000001,
    kOTPortEnabled = 0x25000002,
```

```
    kOTPortOffline = 0x25000003,
    kOTPortOnline = 0x25000004,
    kOTClosePortRequest = 0x25000005,
    kOTYieldPortRequest = 0x25000005,
    kOTNewPortRegistered = 0x25000006,
    kOTPortNetworkChange = 0x25000007,
    kOTConfigurationChanged = 0x26000001,
    kOTSystemSleep = 0x26000002,
    kOTSystemShutdown = 0x26000003,
    kOTSystemAwaken = 0x26000004,
    kOTSystemIdle = 0x26000005,
    kOTSystemSleepPrep = 0x26000006,
    kOTSystemShutdownPrep = 0x26000007,
    kOTSystemAwakenPrep = 0x26000008,
    kOTStackIsLoading = 0x27000001,
    kOTStackWasLoaded = 0x27000002,
    kOTStackIsUnloading = 0x27000003
};

/*   The following event codes are used internally by Open Transport.*/
enum {
    kOTDisablePortEvent = 0x21000001,
    kStreamCloseEvent = 0x21000006,
    kBackgroundStreamEvent = 0x21000009,
    kIoctlRecvFdEvent = 0x2100000A,
    kOTTryShutdownEvent = 0x2100000B,
    kOTScheduleTerminationEvent = 0x2100000C,
    kOTEnablePortEvent = 0x2100000D,
    kOTNewPortRegisteredEvent = 0x2100000E,
    kOTPortOfflineEvent = 0x2100000F,
    kOTPortOnlineEvent = 0x21000010,
    kOTPortNetworkChangeEvent = 0x21000011
};
```

Constants

`T_LISTEN`

A connection request has arrived. Call the `OTListen` function to read the request.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_CONNECT`

The passive peer has accepted a connection that you requested using the `OTConnect` function. Call the `OTRcvConnect` function to retrieve any data or option information that the passive peer has specified when accepting the connection or to retrieve the address to which you are actually connected. The `cookie` parameter to the notifier function is the `sndCall` parameter that you specified when calling the `OTConnect` function.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_DATA`

Normal data has arrived. Depending on the type of service of the endpoint you are using, you can call the `OTRcvUData` function or the `OTRcv` function to read it. Continue reading data until the function returns with the `kOTNoDataErr` result; you will not get another indication that data has arrived until you have read all the data and got this error.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_EXDATA`

Expedited data has arrived. Use the `OTRcv` function to read it. Continue reading data by calling the `OTRcv` function until the function returns with the `kOTNoDataErr` result; you will not get another indication that data has arrived until you have read all the data and got this error.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_DISCONNECT`

A connection has been torn down or rejected. Use the `OTRcvDisconnect` function to clear the event.

If the event is used to signify that a connection has been terminated, the cookie parameter to the notifier is `NULL`.

If the event indicates a rejected connection request, the `cookie` parameter to the notification routine is the same as the `sndCall` parameter that you passed to the `OTConnect` function.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_ERROR`

Obsolete.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_UDERR`

The provider was not able to send the data you specified using the `OTSndUData` function even though the function returned successfully. You must call the `OTRcvUDErr` function to clear this event and determine why the function failed.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_ORDREL`

The remote client has called the `OTSndOrderlyDisconnect` function to initiate an orderly disconnect. You must call the `OTRcvOrderlyDisconnect` function to acknowledge receiving the event.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_GODATA`

Flow-control restrictions have been lifted. You can now send normal data.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_GOEXDATA`

Flow-control restrictions have been lifted. You can now send expedited data.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_REQUEST`

A request has arrived. Depending on the type of service for the endpoint you are using, you can call the `OTRcvRequest` function or the `OTRcvURequest` function to receive it. You must continue to call the function until it returns with the `kOTNoDataErr` result.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_REPLY`

A response to a request has arrived. Depending on the type of service of the endpoint you are using, you can call the `OTRcvReply` function or `OTRcvUReply` function to receive it. You must continue to call the function until it returns with the `kOTNoDataErr` result.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_PASSCON`

When the `OTAccept` function completes, the endpoint provider passes this event to the endpoint receiving the connection (whether that endpoint is the same as or different from the endpoint that calls the `OTAccept` function). The `cookie` parameter contains the `resRef` parameter to the `OTAccept` function.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_RESET`

A connection-oriented endpoint has received a reset from the remote end and has flushed all unread and unsent data. This only occurs for some types of endpoints, and it generally leaves the endpoint in an unknown state.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kPRIVATEEVENT`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kCOMPLETEEVENT`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_BINDCOMPLETE`

The `OTBind` function has completed. The `cookie` parameter contains the `retAddr` parameter of the bind call.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_UNBINDCOMPLETE`

The `OTUnbind` function has completed. The `cookie` parameter is meaningless.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_ACCEPTCOMPLETE`

The `OTAccept` function has completed. The `cookie` parameter contains the `resRef` parameter to the `OTAccept` function.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_REPLYCOMPLETE`
The `OTSndUReply` or `OTSndReply` functions have completed. The `cookie` parameter contains the sequence number used in the `OTSndUReply` or `OTSndReply` call.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_DISCONNECTCOMPLETE`
The `OTSndDisconnect` function has completed. The `cookie` parameter contains the call parameter of the `OTSndDisconnect` function.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_OPTMGMTCOMPLETE`
The `OTOptionManagement` function has completed. The `cookie` parameter contains the `ret` parameter that you passed to the function.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_OPENCOMPLETE`
An asynchronous call to open a provider has completed. The `cookie` parameter contains the provider reference.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_GETPROTADDRCOMPLETE`
The `OTGetProtAddress` function has completed. The `cookie` parameter contains the `peerAddr` parameter that you passed to the `OTGetProtocolAddress` function. If you passed `NULL` for that parameter, the `cookie` parameter contains the address passed in the `boundAddr` parameter.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_RESOLVEADDRCOMPLETE`
The `OTResolveAddress` function has completed. The `cookie` parameter contains the `retAddr` parameter of the `OTResolveAddress` function.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_GETINFOCOMPLETE`
The `OTGetEndpointInfo` function has completed. The `cookie` parameter contains the `info` parameter of the `OTGetEndpointInfo` function.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_SYNCCOMPLETE`
The `OTSync` function has completed. The `cookie` parameter is meaningless.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_MEMORYRELEASED`

You are using an asynchronous endpoint that acknowledges sends and Open Transport is done using the buffers containing the data you are sending. If you called the `OTSnd` function, the `cookie` parameter contains the `buf` parameter. If you called the `OTSndUData` function, the `cookie` parameter contains the `udata` parameter. The `result` parameter contains the number of bytes that were sent. This might be less than the number you meant to send due to flow-control or memory restrictions.

You should not wait for the `T_MEMORYRELEASED` event from a previous send operation to trigger more sends. The exact time this event occurs depends on how the underlying provider is implemented. It might hold on to memory until the next send occurs, or behave in some other way which causes it to delay releasing memory.

Note that `T_MEMORYRELEASED` events can reenter your notifier. See `OTAckSends` (page 366)for more information.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_REGNAMECOMPLETE`

The `OTRegisterName` function has completed. The `cookie` parameter is the reply parameter, unless it was `NULL`. In this case, it is the `req` parameter.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_DELNAMECOMPLETE`

The `OTDeleteName` function or the `OTDeleteNameByID` function has completed. The `cookie` parameter contains the `name` parameter or the `nameId` parameter of the function, respectively.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_LKUPNAMECOMPLETE`

The `OTLookupName` function has completed. The `cookie` parameter contains the `reply` parameter of the `OTLookUpName` function.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_LKUPNAMERESULT`

An `OTLookupName` function has found a name and is returning it, but the lookup is not yet complete. The `cookie` parameter contains the `reply` parameter passed to the `OTLookupName` function.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTSyncIdleEvent`

A synchronous call is waiting to complete.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kSTREAMEVENT`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTReservedEvent1`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kGetmsgEvent`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kStreamReadEvent`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kStreamWriteEvent`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kStreamIoctlEvent`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTReservedEvent2`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kStreamOpenEvent`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kPollEvent`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTReservedEvent3`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTReservedEvent4`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTReservedEvent5`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTReservedEvent6`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTReservedEvent7`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTReservedEvent8`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kSIGNALEVENT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kPROTOCOLEVENT`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTProviderIsDisconnected`

Your provider was bound with a `qlen` parameter value greater than 0 and it has been disconnected (is no longer listening). You receive this event after a port has accepted a request to temporarily yield ownership of a port to another provider, which causes this provider to be disconnected from the port in question. This currently only happens with serial ports, but could also happen with other connection-oriented drivers that have characteristics similar to serial ports. You get a `kOTProviderIsReconnected` message when the port reverts back to this provider's ownership.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTProviderIsReconnected`

Your provider has been reconnected, that is, the cause for its disconnection has been relieved.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTProviderWillClose`

When you return from the notifier function, Open Transport will close the provider whose reference is contained in the `cookie` parameter. The `result` parameter contains a result code specifying the reason why the provider had to close. For example, the user may have decided to switch links using the TCP/IP or AppleTalk control panel. The result codes that can be returned are in the range –3280 through –3285; these are documented in "Open Transport Result Codes" (page 354).

You can only get this event at system task time. Consequently, you are allowed to set the endpoint to synchronous mode (from within the notifier function) and call functions synchronously before you return from the notifier, at which point the provider is closed. After this, any calls other than `OTCloseProvider` will fail with a `kOTOutStateErr` error.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTProviderIsClosed`

The provider has closed. The reason for being closed can be found in the `OTResult` value passed to your notifier. The reasons typically are `kOTPortHasDiedErr`, `kOTPortWasEjectedErr`, or `kOTPortLostConnectionErr`. At this point, any calls other than `OTCloseProvider` will fail with a `kOTOutStateErr` error.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTPortDisabled`

A port has gone off line, as when the user removes a PCMCIA card while the computer is running. The OTResult parameter specifies the reason, if known, and the cookie parameter provides the port reference of the port that went off line. A port going off line often results in providers getting kOTProviderIsClosed events. There is no guarantee in Open Transport as to which of these events will be received first.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTPortEnabled`

A port that had previously been disabled is now reenabled, as when the user reinserts a previously removed PCMCIA card while the computer is running. The cookie parameter is the port reference of the port that is now enabled.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTPortOffline`

The port is now offline.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTPortOnline`

A request has been made to close or yield this port.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTClosePortRequest`

You currently are using a provider that is using a port that some other application wants to use. The OTResult parameter is the reason for the request (normally kOTNoError or kOTUserRequestedErr), and the cookie parameter is a pointer to an OTPortCloseStruct structure.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTYieldPortRequest`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTNewPortRegistered`

A new port has been registered with Open Transport, as when the user inserts a new PCMCIA card. The cookie parameter is the port reference of the new port. Your provider receives this event the first time a new port is enabled. Subsequently, if a port is reenabled after being disabled, you receive the kOTPortEnabled event instead.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTPortNetworkChange`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTConfigurationChanged`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTSystemSleep`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTSystemShutdown`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTSystemAwaken`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTSystemIdle`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTSystemSleepPrep`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTSystemShutdownPrep`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTSystemAwakenPrep`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTStackIsLoading`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTStackWasLoaded`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTStackIsUnloading`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTDisablePortEvent`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kStreamCloseEvent`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kBackgroundStreamEvent`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kIoctlRecvFdEvent`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTTryShutdownEvent`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTScheduleTerminationEvent`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`kOTEnablePortEvent`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTNewPortRegisteredEvent`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTPortOfflineEvent`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTPortOnlineEvent`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kOTPortNetworkChangeEvent`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

Open Transport Flags and Status Codes

Specify information about data transmitted with the `OTSnd` or `OTRcv` functions, or specify options for the `OTOptionManagement` function, or indicate the result status of an option negotiation.

```
typedef UInt32 OTFlags;
/*   These flags are used when sending and receiving data.The constants
defined are masks.*/
enum {
    T_MORE = 0x0001,
    T_EXPEDITED = 0x0002,
    T_ACKNOWLEDGED = 0x0004,
    T_PARTIALDATA = 0x0008,
    T_NORECEIPT = 0x0010,
    T_TIMEDOUT = 0x0020
};

/* These flags are used in the TOptMgmt structure to request services.*/
enum {
    T_NEGOTIATE = 0x0004,
    T_CHECK = 0x0008,
    T_DEFAULT = 0x0010,
    T_CURRENT = 0x0080
};
```

```
/*   These flags are used in the TOptMgmt and TOption structures
to   return results.*/
enum {
    T_SUCCESS = 0x0020,
    T_FAILURE = 0x0040,
    T_PARTSUCCESS = 0x0100,
    T_READONLY = 0x0200,
    T_NOTSUPPORT = 0x0400
};
```

Constants

`T_MORE`

There is more data for the current TSDU or ETSDU. The next send or receive operation will handle additional data for this TSDU or ETSDU.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_EXPEDITED`

On sends, the data is sent as expedited data if the endpoint supports expedited data. On receives, the flag indicates that expedited data was sent.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_ACKNOWLEDGED`

The transaction must be acknowledged before the send or receive function can complete.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_PARTIALDATA`

There is more data for the current TSDU or ETSDU. Unlike `T_MORE`, `T_PARTIALDATA` does not guarantee that the next send or receive operation will handle additional data for this TSDU or ETSDU.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_NORECEIPT`

There is no need to send a `T_REPLY_COMPLETE` event to complete the transaction. If you don't need to know when the transaction is actually done, you can set this flag to improve performance.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_TIMEDOUT`

The reply timed out. If a protocol such as ATP loses the acknowledgment for a transaction that needs to be acknowledged, the transaction will eventually time out. Since the reply didn't really fail (it just timed out), Open Transport can send a `T_REPLY_COMPLETE` event to complete the transaction and set this flag to explain what happened.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_NEGOTIATE`

Negotiate the option values specified in the opt.buf field of the req parameter.

The overall result of the negotiation is specified by the flags field of the ret parameter. The opt.buf field of the ret parameter points to a buffer where negotiated values for each option are placed.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_CHECK`

Verify whether the endpoint supports the options referenced by the opt.buf field of the req parameter. The overall result of the verification is specified by the flags field of the ret parameter. Specific verification results are returned in the opt.buf field of the ret parameter.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_DEFAULT`

Retrieve the default value for those options in the buffer referenced by the req->opt.buf field. To retrieve default values for all the options supported by an endpoint, include just the option T_ALLOPT in the options buffer.Option values are returned in the opt.buf field of the ret parameter.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_CURRENT`

Retrieve the current value for those options that the endpoint supports and that are specified in the buffer referenced by the req->opt.buf field. To retrieve current values for all the options that an endpoint supports, include just the option T_ALLOPT in the options buffer.Option values are returned in the opt.buf field of the ret parameter.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_SUCCESS`

The requested value was negotiated.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_FAILURE`

The negotiation failed.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_PARTSUCCESS`

A lower requested value was negotiated.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_READONLY`

The option was read-only.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_NOTSUPPORT`
The endpoint does not support the requested value.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

T_NOTOS

```
enum {
    T_NOTOS = 0x00,
    T_LDELAY = (1 << 4),
    T_HITHRPT = (1 << 3),
    T_HIREL = (1 << 2)
};
```

Constants

`T_NOTOS`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`T_LDELAY`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`T_HITHRPT`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`T_HIREL`
Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

T_NULL

```
enum {
    T_NULL = NULL,
    T_UNSPEC = -3
};
```

Constants

`T_NULL`

`T_UNSPEC`
The option does not have a fully specified value at this time. An endpoint provider might return this status code if it cannot currently access the option value. This might happen if the endpoint is in the state T_UNBND in systems where the protocol stack resides on a separate host.

T_ROUTINE

```
enum {
    T_ROUTINE = 0,
    T_PRIORITY = 1,
    T_IMMEDIATE = 2,
    T_FLASH = 3,
    T_OVERRIDEFLASH = 4,
    T_CRITIC_ECP = 5,
    T_INETCONTROL = 6,
    T_NETCONTROL = 7
};
```

Constants

`T_ROUTINE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`T_PRIORITY`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`T_IMMEDIATE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`T_FLASH`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`T_OVERRIDEFLASH`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`T_CRITIC_ECP`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`T_INETCONTROL`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

`T_NETCONTROL`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProviders.h`.

Endpoint Flags

Specifies information about an endpoint.

```
enum {
    T_SENDZERO = 0x0001,
    T_XPG4_1 = 0x0002,
    T_CAN_SUPPORT_MDATA = 0x10000000,
    T_CAN_RESOLVE_ADDR = 0x40000000,
    T_CAN_SUPPLY_MIB = 0x20000000
};
```

Constants

`T_SENDZERO`

This endpoint lets you send and receive zero-length TSDUs.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_XPG4_1`

This endpoint supports the `OTGetProtAddress` function (conforms to XTI in XPG4).

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_CAN_SUPPORT_MDATA`

This endpoint supports `M_DATA`, that is, it permits receiving and returning raw packets.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_CAN_RESOLVE_ADDR`

This endpoint supports the `OTResolveAddress` function.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_CAN_SUPPLY_MIB`

This endpoint can supply the Management Information Base (MIB) data used by the Simple Network Management Protocol (SNMP). At this time you cannot access this data.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

T_UNSPEC

Constants

T_YES

```
enum {
    T_YES = 1,
    T_NO = 0,
    T_UNUSED = -1,
    kT_NULL = 0,
    T_ABSREQ = 0x8000
};
```

Constants

`T_YES`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_NO`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_UNUSED`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`kT_NULL`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`T_ABSREQ`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

TCP_NODELAY

```
enum {
    TCP_NODELAY = 0x01,
    TCP_MAXSEG = 0x02,
    TCP_NOTIFY_THRESHOLD = 0x10,
    TCP_ABORT_THRESHOLD = 0x11,
    TCP_CONN_NOTIFY_THRESHOLD = 0x12,
    TCP_CONN_ABORT_THRESHOLD = 0x13,
    TCP_OOBINLINE = 0x14,
    TCP_URGENT_PTR_TYPE = 0x15,
    TCP_KEEPALIVE = 0x0008
};
```

Constants

`TCP_NODELAY`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`TCP_MAXSEG`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`TCP_NOTIFY_THRESHOLD`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`TCP_ABORT_THRESHOLD`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`TCP_CONN_NOTIFY_THRESHOLD`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`TCP_CONN_ABORT_THRESHOLD`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`TCP_OOBINLINE`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`TCP_URGENT_PTR_TYPE`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`TCP_KEEPALIVE`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

TE_OPENED

```
enum {
    TE_OPENED = 1,
    TE_BIND = 2,
    TE_OPTMGMT = 3,
    TE_UNBIND = 4,
    TE_CLOSED = 5,
    TE_CONNECT1 = 6,
    TE_CONNECT2 = 7,
    TE_ACCEPT1 = 8,
    TE_ACCEPT2 = 9,
    TE_ACCEPT3 = 10,
    TE_SND = 11,
    TE_SNDDIS1 = 12,
    TE_SNDDIS2 = 13,
    TE_SNDREL = 14,
    TE_SNDUDATA = 15,
    TE_LISTEN = 16,
    TE_RCVCONNECT = 17,
    TE_RCV = 18,
    TE_RCVDIS1 = 19,
    TE_RCVDIS2 = 20,
    TE_RCVDIS3 = 21,
    TE_RCVREL = 22,
    TE_RCVUDATA = 23,
    TE_RCVUDERR = 24,
    TE_PASS_CONN = 25,
    TE_BAD_EVENT = 26
};
```

Constants

`TE_OPENED`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`TE_BIND`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`TE_OPTMGMT`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`TE_UNBIND`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`TE_CLOSED`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`TE_CONNECT1`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProtocol.h`.

`TE_CONNECT2`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TE_ACCEPT1`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TE_ACCEPT2`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TE_ACCEPT3`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TE_SND`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TE_SNDDIS1`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TE_SNDDIS2`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TE_SNDREL`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TE_SNDUDATA`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TE_LISTEN`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TE_RCVCONNECT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TE_RCV`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TE_RCVDIS1`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TE_RCVDIS2`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TE_RCVDIS3`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TE_RCVREL`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TE_RCVUDATA`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TE_RCVUDERR`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TE_PASS_CONN`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TE_BAD_EVENT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

TS_UNBND

```
enum {
    TS_UNBND = 1,
    TS_WACK_BREQ = 2,
    TS_WACK_UREQ = 3,
    TS_IDLE = 4,
    TS_WACK_OPTREQ = 5,
    TS_WACK_CREQ = 6,
    TS_WCON_CREQ = 7,
    TS_WRES_CIND = 8,
    TS_WACK_CRES = 9,
    TS_DATA_XFER = 10,
    TS_WIND_ORDREL = 11,
    TS_WREQ_ORDREL = 12,
    TS_WACK_DREQ6 = 13,
    TS_WACK_DREQ7 = 14,
    TS_WACK_DREQ9 = 15,
    TS_WACK_DREQ10 = 16,
    TS_WACK_DREQ11 = 17,
    TS_WACK_ORDREL = 18,
    TS_NOSTATES = 19,
    TS_BAD_STATE = 19
};
```

Constants

`TS_UNBND`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TS_WACK_BREQ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TS_WACK_UREQ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TS_IDLE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TS_WACK_OPTREQ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TS_WACK_CREQ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TS_WCON_CREQ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TS_WRES_CIND`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TS_WACK_CRES`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TS_DATA_XFER`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TS_WIND_ORDREL`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TS_WREQ_ORDREL`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TS_WACK_DREQ6`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TS_WACK_DREQ7`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TS_WACK_DREQ9`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TS_WACK_DREQ10`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TS_WACK_DREQ11`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TS_WACK_ORDREL`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TS_NOSTATES`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

`TS_BAD_STATE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransportProtocol.h`.

TSUCCESS

```
typedef UInt16 OTXTIErr;
enum {
    TSUCCESS = 0,
    TBADADDR = 1,
    TBADOPT = 2,
    TACCES = 3,
    TBADF = 4,
    TNOADDR = 5,
    TOUTSTATE = 6,
    TBADSEQ = 7,
    TSYSERR = 8,
    TLOOK = 9,
    TBADDATA = 10,
    TBUFOVFLW = 11,
    TFLOW = 12,
    TNODATA = 13,
    TNODIS = 14,
    TNOUDERR = 15,
    TBADFLAG = 16,
    TNOREL = 17,
    TNOTSUPPORT = 18,
    TSTATECHNG = 19,
    TNOSTRUCTYPE = 20,
    TBADNAME = 21,
    TBADQLEN = 22,
    TADDRBUSY = 23,
    TINDOUT = 24,
    TPROVMISMATCH = 25,
    TRESQLEN = 26,
    TRESADDR = 27,
    TQFULL = 28,
    TPROTO = 29,
    TBADSYNC = 30,
    TCANCELED = 31,
    TLASTXTIERROR = 31
};
```

Constants

`TSUCCESS`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`TBADADDR`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`TBADOPT`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`TACCES`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`TBADF`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`TNOADDR`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`TOUTSTATE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`TBADSEQ`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`TSYSERR`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`TLOOK`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`TBADDATA`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`TBUFOVFLW`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`TFLOW`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`TNODATA`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`TNODIS`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`TNOUDERR`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`TBADFLAG`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`TNOREL`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`TNOTSUPPORT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`TSTATECHNG`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`TNOSTRUCTYPE`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`TBADNAME`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`TBADQLEN`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`TADDRBUSY`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`TINDOUT`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`TPROVMISMATCH`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`TRESQLEN`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`TRESADDR`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`TQFULL`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`TPROTO`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`TBADSYNC`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`TCANCELED`
Available in Mac OS X v10.0 and later.
Declared in `OpenTransport.h`.

`TLASTXTIERROR`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

UDP_CHECKSUM

```
enum {
    UDP_CHECKSUM = 0x0600,
    UDP_RX_ICMP = 0x02
};
```

Constants

`UDP_CHECKSUM`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

`UDP_RX_ICMP`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransportProviders.h`.

XTI-Level Options and Generic Options

Specifies constant names for XTI-level options.

```
enum {
    XTI_DEBUG = 0x0001,
    XTI_LINGER = 0x0080,
    XTI_RCVBUF = 0x1002,
    XTI_RCVLOWAT = 0x1004,
    XTI_SNDBUF = 0x1001,
    XTI_SNDLOWAT = 0x1003,
    XTI_PROTOTYPE = 0x1005,
    OPT_CHECKSUM = 0x0600,
    OPT_RETRYCNT = 0x0601,
    OPT_INTERVAL = 0x0602,
    OPT_ENABLEEOM = 0x0603,
    OPT_SELFSEND = 0x0604,
    OPT_SERVERSTATUS = 0x0605,
    OPT_ALERTENABLE = 0x0606,
    OPT_KEEPALIVE = 0x0008
};
```

Constants

`XTI_DEBUG`

A 32 bit constant specifying whether debugging is enabled. Debugging is disabled if the option is specified with no value. This option is an absolute requirement.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`XTI_LINGER`

A value defined by a linger structure (page 571) that specifies whether the option is turned on (T_YES) or off (T_NO) and specifies a linger period in seconds. This option is an absolute requirement; however, you do not have to specify a value for the l_linger field of the linger structure.

You use this option to extend the execution of the OTCloseProvider function for some specified amount of time. The delay allows data still queued in the endpoint's internal send buffer to be sent before the endpoint provider is closed. If you call the OTCloseProvider function and the send buffer is not empty, the endpoint provider attempts to send the remaining data during the linger period, before closing. Open Transport discards any data remaining in the send buffer after the linger period has elapsed.

Consult the documentation for your protocol to determine the valid range of values for the linger period.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`XTI_RCVBUF`

A 32-bit integer specifying the size of the endpoint's internal buffer allocated for receiving data. You can increase the size of this buffer for high-volume connections or decrease the buffer to limit the possible backlog of incoming data.

This option is not an absolute requirement. Consult the documentation for your protocol to determine the valid range of values for the buffer size.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`XTI_RCVLOWAT`

A 32-bit integer specifying the low-water mark for the receive buffer—that is, the number of bytes that must accumulate in the endpoint's internal receive buffer before you are advised that data has arrived via a T_DATA event. Choosing a value that is too low might result in your application's getting an excessive number of T_DATA events and doing unnecessary reads. Choosing a value that is too high might result in Open Transport running out of memory and disabling incoming data packets.

This option is not an absolute requirement. Consult the documentation for your protocol to determine the valid range of values for the low-water mark.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`XTI_SNDBUF`

A 32-bit integer specifying the size of the endpoint's internal buffer allocated for sending data. Specifying a value that is too low might result in Open Transport doing more sends than necessary and wasting processor time; specifying a value that is too high might cause flow control problems.

This option is not an absolute requirement. Consult the documentation for your protocol to determine the valid range of values for the buffer size.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`XTI_SNDLOWAT`

A 32-bit integer specifying the low-water mark for the send buffer—that is, the number of bytes that must accumulate in the endpoint's internal send buffer before Open Transport actually sends the data. Choosing a value that is too low might result in Open Transport's doing too many sends and wasting processor time. Choosing a value that is too high might result in flow control problems. A value that is slightly lower than the largest packet size defined for the endpoint is a good choice.

This option is not an absolute requirement. Consult the documentation for your protocol to determine the valid range of values for the low-water mark.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`XTI_PROTOTYPE`

The protocol type used by the endpoint. The option is supported by the RawIP endpoint.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`OPT_CHECKSUM`

A 32-bit constant specifying whether checksums are performed. Specify 1 to turn the option on and 0 to turn it off. If you turn it on, a checksum is calculated when a packet is sent and recalculated when the packet is received. If the checksum values match, the client receiving the packet can be fairly certain that data has not been corrupted or lost during transmission. If the checksum values don't match, the receiver discards the packet.

This option is usually implemented by the lowest-level protocol, although you might be allowed to set it at a higher level. For example, if you use an ATP endpoint, you can set checksumming at the ATP level, even though it is implemented by the underlying DDP protocol.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`OPT_RETRYCNT`

A 32-bit integer specifying the number of times a function can attempt packet delivery before returning with an error. A value of 0 means that the function should attempt packet delivery an infinite number of times.

This option is usually implemented by connection-oriented endpoints or connectionless transaction-based endpoints to enable reliable delivery of data. Such protocols normally set a default value for this option.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`OPT_INTERVAL`

A 32-bit integer specifying the interval of time in milliseconds that should elapse between attempts to deliver a packet. The number of attempts is defined by the OPT_RETRYCNT option.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`OPT_ENABLEEOM`

An 32-bit integer specifying end-of-message capability. If you set this option, you enable the use of the T_MORE flag with the OTSnd function to mark the end of a logical unit. This option has meaning only for connection-oriented protocols. A value of 0 clears the option; a value of 1 sets it.

This option is not association-related.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`OPT_SELFSEND`

A 32-bit integer allowing you to send broadcast packets to yourself. A value of 0 clears the option; a value of 1 sets it.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`OPT_SERVERSTATUS`

A string that sets the server's status. The maximum length is protocol dependent. For more information, consult the documentation for the protocol you are using.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`OPT_ALERTENABLE`

A keepalive structure that specifies whether "keep alive" is turned on (T_YES) or off (T_NO) and specifies the timeout period in minutes.

Connection-oriented protocols can use this option to check that the connection is maintained. If a connection is established but there is no data being transferred, you can specify a time limit within which Open Transport checks to see that the remote end of the connection is still alive. If it is not, Open Transport tears down the connection.

This option is association-related.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

`OPT_KEEPALIVE`

Enables or disables protocol alerts.

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

Discussion

Open Transport defines XTI-level options. These options are not association-related. If the protocol you are using supports these options, you can negotiate them while the endpoint is in any state. The protocol level for all of these options is XTI_GENERIC.

Open Transport also defines some generic options that you can use with any protocol that supports them, listed in this enumeration starting with `OPT_CHECKSUM`. The protocol level for each of these options is the same as the name of the protocol that supports them.

XTI_GENERIC

```
enum {
    XTI_GENERIC = 0xFFFF
};
```

Constants

`XTI_GENERIC`

Available in Mac OS X v10.0 and later.

Declared in `OpenTransport.h`.

Result Codes

The most common result codes returned by Open Transport are listed below.

Result Code	Value	Description
kOTNoError	0	No Error occurred Available in Mac OS X v10.0 and later.
kOTBadAddressErr	-3150	XTI2OSStatus(TBADADDR) A Bad address was specified Available in Mac OS X v10.0 and later.
kOTBadOptionErr	-3151	XTI2OSStatus(TBADOPT) A Bad option was specified Available in Mac OS X v10.0 and later.
kOTAccessErr	-3152	XTI2OSStatus(TACCES) Missing access permission Available in Mac OS X v10.0 and later.
kOTBadReferenceErr	-3153	XTI2OSStatus(TBADF) Bad provider reference Available in Mac OS X v10.0 and later.
kOTNoAddressErr	-3154	XTI2OSStatus(TNOADDR) No address was specified Available in Mac OS X v10.0 and later.
kOTOutStateErr	-3155	XTI2OSStatus(TOUTSTATE) Call issued in wrong state Available in Mac OS X v10.0 and later.
kOTBadSequenceErr	-3156	XTI2OSStatus(TBADSEQ) Sequence specified does not exist Available in Mac OS X v10.0 and later.
kOTSysErrorErr	-3157	XTI2OSStatus(TSYSERR) A system error occurred Available in Mac OS X v10.0 and later.
kOTLookErr	-3158	XTI2OSStatus(TLOOK) An event occurred - call Look() Available in Mac OS X v10.0 and later.
kOTBadDataErr	-3159	XTI2OSStatus(TBADDATA) An illegal amount of data was specified Available in Mac OS X v10.0 and later.
kOTBufferOverflowErr	-3160	XTI2OSStatus(TBUFOVFLW) Passed buffer not big enough Available in Mac OS X v10.0 and later.
kOTFlowErr	-3161	XTI2OSStatus(TFLOW) Provider is flow-controlled Available in Mac OS X v10.0 and later.

Result Code	Value	Description
kOTNoDataErr	-3162	XTI2OSStatus(TNODATA) No data available for reading Available in Mac OS X v10.0 and later.
kOTNoDisconnectErr	-3163	XTI2OSStatus(TNODIS) No disconnect indication available Available in Mac OS X v10.0 and later.
kOTNoUDErrErr	-3164	XTI2OSStatus(TNOUDERR) No Unit Data Error indication available Available in Mac OS X v10.0 and later.
kOTBadFlagErr	-3165	XTI2OSStatus(TBADFLAG) A Bad flag value was supplied Available in Mac OS X v10.0 and later.
kOTNoReleaseErr	-3166	XTI2OSStatus(TNOREL) No orderly release indication available Available in Mac OS X v10.0 and later.
kOTNotSupportedErr	-3167	XTI2OSStatus(TNOTSUPPORT) Command is not supported Available in Mac OS X v10.0 and later.
kOTStateChangeErr	-3168	XTI2OSStatus(TSTATECHNG) State is changing - try again later Available in Mac OS X v10.0 and later.
kOTNoStructureTypeErr	-3169	XTI2OSStatus(TNOSTRUCTYPE) Bad structure type requested for OTAlloc Available in Mac OS X v10.0 and later.
kOTBadNameErr	-3170	XTI2OSStatus(TBADNAME) A bad endpoint name was supplied Available in Mac OS X v10.0 and later.
kOTBadQLenErr	-3171	XTI2OSStatus(TBADQLEN) A Bind to an in-use addr with qlen > 0 Available in Mac OS X v10.0 and later.
kOTAddressBusyErr	-3172	XTI2OSStatus(TADDRBUSY) Address requested is already in use Available in Mac OS X v10.0 and later.
kOTIndOutErr	-3173	XTI2OSStatus(TINDOUT) Accept failed because of pending listen Available in Mac OS X v10.0 and later.
kOTProviderMismatchErr	-3174	XTI2OSStatus(TPROVMISMATCH) Tried to accept on incompatible endpoint Available in Mac OS X v10.0 and later.

Result Code	Value	Description
kOTResQLenErr	-3175	XTI2OSStatus(TRESQLEN) Available in Mac OS X v10.0 and later.
kOTResAddressErr	-3176	XTI2OSStatus(TRESADDR) Available in Mac OS X v10.0 and later.
kOTQFullErr	-3177	XTI2OSStatus(TQFULL) Available in Mac OS X v10.0 and later.
kOTProtocolErr	-3178	XTI2OSStatus(TPROTO) An unspecified provider error occurred Available in Mac OS X v10.0 and later.
kOTBadSyncErr	-3179	XTI2OSStatus(TBADSYNC) A synchronous call at interrupt time Available in Mac OS X v10.0 and later.
kOTCanceledErr	-3180	XTI2OSStatus(TCANCELED) The command was cancelled Available in Mac OS X v10.0 and later.
kEPERMErr	-3200	Permission denied Available in Mac OS X v10.0 and later.
kENOENTErr	-3201	No such file or directory Available in Mac OS X v10.0 and later.
kOTNotFoundErr	-3201	OT generic not found error Available in Mac OS X v10.0 and later.
kENORSRCErr	-3202	No such resource Available in Mac OS X v10.0 and later.
kEINTRErr	-3203	Interrupted system service Available in Mac OS X v10.0 and later.
kEIOErr	-3204	I/O error Available in Mac OS X v10.0 and later.
kENXIOErr	-3205	No such device or address Available in Mac OS X v10.0 and later.
kEBADFErr	-3208	Bad file number Available in Mac OS X v10.0 and later.
kEAGAINErr	-3210	Try operation again later Available in Mac OS X v10.0 and later.

Result Code	Value	Description
kENOMEMErr	-3211	Not enough space Available in Mac OS X v10.0 and later.
kOTOutOfMemoryErr	-3211	OT ran out of memory, may be a temporary Available in Mac OS X v10.0 and later.
kEACCESErr	-3212	Permission denied Available in Mac OS X v10.0 and later.
kEFAULTErr	-3213	Bad address Available in Mac OS X v10.0 and later.
kEBUSYErr	-3215	Device or resource busy Available in Mac OS X v10.0 and later.
kEEXISTErr	-3216	File exists Available in Mac OS X v10.0 and later.
kOTDuplicateFoundErr	-3216	OT generic duplicate found error Available in Mac OS X v10.0 and later.
kENODEVErr	-3218	No such device Available in Mac OS X v10.0 and later.
kEINVALErr	-3221	Invalid argument Available in Mac OS X v10.0 and later.
kENOTTYErr	-3224	Not a character device Available in Mac OS X v10.0 and later.
kEPIPEErr	-3231	Broken pipe Available in Mac OS X v10.0 and later.
kERANGEErr	-3233	Message size too large for STREAM Available in Mac OS X v10.0 and later.
kEDEADLKErr	-3234	or a deadlock would occur Available in Mac OS X v10.0 and later.
kEWOULDBLOCKErr	-3234	Call would block, so was aborted Available in Mac OS X v10.0 and later.
kEALREADYErr	-3236	Available in Mac OS X v10.0 and later.
kENOTSOCKErr	-3237	Socket operation on non-socket Available in Mac OS X v10.0 and later.

Result Code	Value	Description
kEDESTADDRREQErr	-3238	Destination address required Available in Mac OS X v10.0 and later.
kEMSGSIZEErr	-3239	Message too long Available in Mac OS X v10.0 and later.
kEPROTOTYPEErr	-3240	Protocol wrong type for socket Available in Mac OS X v10.0 and later.
kENOPROTOOPTErr	-3241	Protocol not available Available in Mac OS X v10.0 and later.
kEPROTONOSUPPORTErr	-3242	Protocol not supported Available in Mac OS X v10.0 and later.
kESOCKTNOSUPPORTErr	-3243	Socket type not supported Available in Mac OS X v10.0 and later.
kEOPNOTSUPPErr	-3244	Operation not supported on socket Available in Mac OS X v10.0 and later.
kEADDRINUSEErr	-3247	Address already in use Available in Mac OS X v10.0 and later.
kEADDRNOTAVAILErr	-3248	Can't assign requested address Available in Mac OS X v10.0 and later.
kENETDOWNErr	-3249	Network is down Available in Mac OS X v10.0 and later.
kENETUNREACHErr	-3250	Network is unreachable Available in Mac OS X v10.0 and later.
kENETRESETErr	-3251	Network dropped connection on reset Available in Mac OS X v10.0 and later.
kECONNABORTEDErr	-3252	Software caused connection abort Available in Mac OS X v10.0 and later.
kECONNRESETErr	-3253	Connection reset by peer Available in Mac OS X v10.0 and later.
kENOBUFSErr	-3254	No buffer space available Available in Mac OS X v10.0 and later.

Result Code	Value	Description
kEISCONNErr	-3255	Socket is already connected Available in Mac OS X v10.0 and later.
kENOTCONNErr	-3256	Socket is not connected Available in Mac OS X v10.0 and later.
kESHUTDOWNErr	-3257	Can't send after socket shutdown Available in Mac OS X v10.0 and later.
kETOOMANYREFSErr	-3258	Too many references: can't splice Available in Mac OS X v10.0 and later.
kETIMEDOUTErr	-3259	Connection timed out Available in Mac OS X v10.0 and later.
kECONNREFUSEDErr	-3260	Connection refused Available in Mac OS X v10.0 and later.
kEHOSTDOWNErr	-3263	Host is down Available in Mac OS X v10.0 and later.
kEHOSTUNREACHErr	-3264	No route to host Available in Mac OS X v10.0 and later.
kEPROTOErr	-3269	Available in Mac OS X v10.0 and later.
kETIMEErr	-3270	Available in Mac OS X v10.0 and later.
kENOSRErr	-3271	Available in Mac OS X v10.0 and later.
kEBADMSGErr	-3272	Available in Mac OS X v10.0 and later.
kECANCELErr	-3273	Available in Mac OS X v10.0 and later.
kENOSTRErr	-3274	Available in Mac OS X v10.0 and later.
kENODATAErr	-3275	Available in Mac OS X v10.0 and later.
kEINPROGRESSErr	-3276	Available in Mac OS X v10.0 and later.
kESRCHErr	-3277	Available in Mac OS X v10.0 and later.
kENOMSGErr	-3278	Available in Mac OS X v10.0 and later.
kOTClientNotInittedErr	-3279	Available in Mac OS X v10.0 and later.
kOTPortHasDiedErr	-3280	Available in Mac OS X v10.0 and later.
kOTPortWasEjectedErr	-3281	Available in Mac OS X v10.0 and later.

Result Code	Value	Description
kOTBadConfigurationErr	-3282	Available in Mac OS X v10.0 and later.
kOTConfigurationChangedErr	-3283	Available in Mac OS X v10.0 and later.
kOTUserRequestedErr	-3284	Available in Mac OS X v10.0 and later.
kOTPortLostConnection	-3285	Available in Mac OS X v10.0 and later.
kModemOutOfMemory	-14000	Available in Mac OS X v10.0 and later.
kModemPreferencesMissing	-14001	Available in Mac OS X v10.0 and later.
kModemScriptMissing	-14002	Available in Mac OS X v10.0 and later.

Deprecated Open Transport Functions

A function identified as deprecated has been superseded and may become unsupported in the future.

Deprecated in Mac OS X v10.4

CloseOpenTransportInContext

Unregisters your application or code resource connection to Open Transport. (Deprecated in Mac OS X v10.4.)

```
void CloseOpenTransportInContext (
   OTClientContextPtr clientContext
);
```

Parameters

clientContext

Availability

Available in CarbonLib 1.0 and later.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Carbon Porting Notes

The `CloseOpenTransportinContext` function acts like the pre-Carbon `CloseOpenTransport` function except that it takes an additional parameter, an `OTClientContextPtr`, which can be `NULL` for applications. Other types of clients must provide a valid client context pointer.

Declared In

`OpenTransport.h`

DisposeOTListSearchUPP

Disposes of a universal procedure pointer (UPP) to a list search callback. (Deprecated in Mac OS X v10.4.)

```
void DisposeOTListSearchUPP (
   OTListSearchUPP userUPP
);
```

Parameters

userUPP

Availability

Available in CarbonLib 1.0 and later.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Declared In
`OpenTransport.h`

DisposeOTNotifyUPP

Disposes of a universal procedure pointer (UPP) to a notification callback. (Deprecated in Mac OS X v10.4.)

```
void DisposeOTNotifyUPP (
   OTNotifyUPP userUPP
);
```

Parameters

userUPP

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`OpenTransport.h`

DisposeOTProcessUPP

Disposes of a universal procedure pointer (UPP) to a process callback. (Deprecated in Mac OS X v10.4.)

```
void DisposeOTProcessUPP (
   OTProcessUPP userUPP
);
```

Parameters

userUPP

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`OpenTransport.h`

InitOpenTransportInContext

Initializes the parts of Open Transport for use by the application or code resource. (Deprecated in Mac OS X v10.4.)

```
OSStatus InitOpenTransportInContext (
   OTInitializationFlags flags,
   OTClientContextPtr *outClientContext
);
```

Parameters

flags

Tells Open Transport whether your code is an application or a plug-in.

outClientContext

Returns the client context pointer.

Return Value

A result code. See "Open Transport Result Codes" (page 354).

Discussion

In Carbon, the `InitOpenTransportInContext` function acts like the pre-Carbon InitOpenTransport function, except that it takes parameters that specify initialization context explicitly.

Use the flags parameter to tell Open Transport whether your code is an application or some other target (for example, a plug-in that runs in an application context but is not the application itself). The second parameter returns the client context pointer, which you must pass to other asset-creation routines. For more information, see Understanding Open Transport Asset Tracking at http://developer.apple.com/technotes/tn/tn1173.html.

Availability

Available in CarbonLib 1.0 and later.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`OpenTransport.h`

InvokeOTListSearchUPP

Calls a list search callback. (Deprecated in Mac OS X v10.4.)

```
Boolean InvokeOTListSearchUPP (
   const void *ref,
   OTLink *linkToCheck,
   OTListSearchUPP userUPP
);
```

Parameters

ref

linkToCheck

userUPP

Availability

Available in CarbonLib 1.0 and later.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Declared In

`OpenTransport.h`

InvokeOTNotifyUPP

Calls a notification callback. (Deprecated in Mac OS X v10.4.)

```
void InvokeOTNotifyUPP (
   void *contextPtr,
   OTEventCode code,
   OTResult result,
   void *cookie,
   OTNotifyUPP userUPP
);
```

Parameters

contextPtr

code

result

cookie

userUPP

Availability

Available in CarbonLib 1.0 and later.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Declared In

OpenTransport.h

InvokeOTProcessUPP

Calls a process callback. (Deprecated in Mac OS X v10.4.)

```
void InvokeOTProcessUPP (
   void *arg,
   OTProcessUPP userUPP
);
```

Parameters

arg

userUPP

Availability

Available in CarbonLib 1.0 and later.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Declared In

OpenTransport.h

NewOTListSearchUPP

Creates a new universal procedure pointer (UPP) to a list search callback. (Deprecated in Mac OS X v10.4.)

```
OTListSearchUPP NewOTListSearchUPP (
   OTListSearchProcPtr userRoutine
);
```

Parameters

userRoutine

Return Value
See the description of the `OTListSearchUPP` data type.

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`OpenTransport.h`

NewOTNotifyUPP

Creates a new universal procedure pointer (UPP) to a notification callback. (Deprecated in Mac OS X v10.4.)

```
OTNotifyUPP NewOTNotifyUPP (
   OTNotifyProcPtr userRoutine
);
```

Parameters

userRoutine

Return Value
See the description of the `OTNotifyUPP` data type.

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`OpenTransport.h`

NewOTProcessUPP

Creates a new universal procedure pointer (UPP) to a process callback. (Deprecated in Mac OS X v10.4.)

```
OTProcessUPP NewOTProcessUPP (
   OTProcessProcPtr userRoutine
);
```

Parameters

userRoutine

Return Value
See the description of the `OTProcessUPP` data type.

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.

Declared In
`OpenTransport.h`

OTAccept

Accepts an incoming connection request. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTAccept (
   EndpointRef listener,
   EndpointRef worker,
   TCall *call
);
```

Parameters
listener
worker
call

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTAckSends

Specifies that a provider make an internal copy of data being sent and that it notify you when it has finished sending data. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTAckSends (
   ProviderRef ref
);
```

Parameters
ref

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
By default, providers make an internal copy of data before sending it and they do not acknowledge sends. If you use the `OTAckSends` function to specify that the provider acknowledge sends and you call a function that sends data, the provider does not copy the data before sending it. Instead, it reads data directly from your buffer while sending. For this reason, you must not change the contents of your buffer until the provider is no longer using it. The provider lets you know that it has finished using the buffer by calling your notifier function and passing `T_MEMORYRELEASED` event code for the `code` parameter, a pointer to the buffer that was sent in the `cookie` parameter, and the size of the buffer in the `result` parameter.

If you have not installed a notifier function for the provider, this function returns the `kOTAccessErr` result.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTAddFirst

Places a link at the front of a FIFO list. (Deprecated in Mac OS X v10.4.)

```
void OTAddFirst (
   OTList *list,
   OTLink *link
);
```

Parameters
list
link

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTAddLast

Adds a link to the end of a FIFO list. (Deprecated in Mac OS X v10.4.)

```
void OTAddLast (
   OTList *list,
   OTLink *link
);
```

Parameters

list

link

Availability

Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

OpenTransport.h

OTAllocInContext

Allocates a data structure of a specified type. (Deprecated in Mac OS X v10.4.)

```
void * OTAllocInContext (
   EndpointRef ref,
   OTStructType structType,
   UInt32 fields,
   OSStatus *err,
   OTClientContextPtr clientContext
);
```

Parameters

ref

structType

fields

err

clientContext

Discussion

In general, Apple recommends that you avoid the OTAllocInContext call because using it extensively causes your program to allocate and deallocate many memory blocks, with each extra memory allocation costing time.

Under Carbon, OTAllocInContext takes a client context pointer. Applications may pass NULL after calling InitOpenTransport(kInitOTForApplicationMask, ...). Non-applications must always pass a valid client context.

Availability

Available in CarbonLib 1.0 and later.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

OpenTransport.h

OTAllocMemInContext

Allocates memory using an explicit client context. (Deprecated in Mac OS X v10.4.)

```
void * OTAllocMemInContext (
   OTByteCount size,
   OTClientContextPtr clientContext
);
```

Parameters

size

clientContext

Availability

Available in CarbonLib 1.0 and later.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

OpenTransport.h

OTAsyncOpenAppleTalkServicesInContext

Opens an asynchronous AppleTalk service provider in context. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTAsyncOpenAppleTalkServicesInContext (
   OTConfigurationRef cfig,
   OTOpenFlags flags,
   OTNotifyUPP proc,
   void *contextPtr,
   OTClientContextPtr clientContext
);
```

Parameters

cfig

flags

proc

contextPtr

clientContext

Return Value

A result code. See "Open Transport Result Codes" (page 354).

Discussion

Applications may pass a NULL context pointer but nonapplications must always pass a valid client context pointer.

You receive a client context pointer when you call the function `InitOpenTransportInContext` (page 362).

Availability

Available in CarbonLib 1.0 and later.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In
`OpenTransportProviders.h`

OTAsyncOpenEndpointInContext

Opens an endpoint and installs a notifier callback function for the endpoint. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTAsyncOpenEndpointInContext (
   OTConfigurationRef config,
   OTOpenFlags oflag,
   TEndpointInfo *info,
   OTNotifyUPP upp,
   void *contextPtr,
   OTClientContextPtr clientContext
);
```

Parameters

config

oflag

info

upp

contextPtr

clientContext

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
Applications may pass a NULL context pointer but nonapplications must always pass a valid client context pointer.

You receive a client context pointer when you call the function `InitOpenTransportInContext` (page 362).

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTAsyncOpenInternetServicesInContext

Opens the TCP/IP service provider and returns an Internet services reference. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTAsyncOpenInternetServicesInContext (
   OTConfigurationRef cfig,
   OTOpenFlags oflag,
   OTNotifyUPP upp,
   void *contextPtr,
   OTClientContextPtr clientContext
);
```

Parameters

cfig

oflag

upp

contextPtr

clientContext

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
Applications may pass a NULL context pointer but nonapplications must always pass a valid client context pointer.

You receive a client context pointer when you call the function `InitOpenTransportInContext` (page 362).

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransportProviders.h`

OTAsyncOpenMapperInContext

Creates an asynchronous mapper and installs a notifier function for the mapper provider. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTAsyncOpenMapperInContext (
   OTConfigurationRef config,
   OTOpenFlags oflag,
   OTNotifyUPP upp,
   void *contextPtr,
   OTClientContextPtr clientContext
);
```

Parameters

config

oflag

upp

contextPtr

clientContext

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
Applications may pass a NULL context pointer but nonapplications must always pass a valid client context pointer.

You receive a client context pointer when you call the function `InitOpenTransportInContext` (page 362).

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTATalkGetInfo

Obtains information about the AppleTalk environment for a given node. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTATalkGetInfo (
   ATSvcRef ref,
   TNetbuf *info
);
```

Parameters
ref
info

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
The `OTATalkGetInfo` function returns the information contained in the `AppleTalkInfo` data structure that describes your current AppleTalk environment. This includes your network number and node ID, the network number and node ID of a local router, and the current network range for the extended network to which the machine is connected.

If you execute this function asynchronously, Open Transport calls your notifier with a `T_GETATALKINFOCOMPLETE` completion event to signal the function's completion and uses your notifier's `cookie` parameter for the AppleTalk information. The `cookie` parameter actually holds a pointer to a `TNetbuf` structure, which points in turn to a buffer containing the `AppleTalkInfo` structure. The maximum size of this buffer is 22 bytes.

If the machine is multihomed—that is, if multiple network numbers and node numbers are associated with the same machine—the `OTATalkGetInfo` function returns information about the node whose network number and node ID are selected in the AppleTalk control panel.

Availability
Available in CarbonLib 1.0.2 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransportProviders.h`

OTATalkGetLocalZones

Obtains a list of the zones available on your network. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTATalkGetLocalZones (
   ATSvcRef ref,
   TNetbuf *zones
);
```

Parameters

ref

zones

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
The `OTATalkGetLocalZones` function returns a list of the zone names in your application's network if it is an extended network. These are all the zones to which your node can belong. If your application is in a nonextended network, this function returns only one zone name, the same one returned by the `OTATalkGetMyZone` function.

If you execute this function asynchronously, Open Transport calls your notifier function with a `T_GETLOCALZONESCOMPLETE` completion event to signal the function's completion and uses your notifier's `cookie` parameter for the list of zones. The `cookie` parameter actually holds a pointer to a `TNetbuf` structure, which points to a buffer containing a list of zone names, each of which is stored as a Pascal-style string. Using a Pascal-style string for the zone name is redundant since you can determine the length of the string from the `maxlen` field of the `TNetbuf` structure, but the other zone-related calls use Pascal-style strings, so this call also uses them for consistency.

Each string can be up to 32 characters in length, and if you add a length byte, each can have a maximum size of 33 bytes. As there can be a maximum of 254 zones on an extended network, the maximum size of the buffer is 8382 bytes.

Because zone names are often less than 32 characters long and AppleTalk service providers don't pad short names, 6 KB bytes is likely to be a safe value for the buffer's size, defined by the `TNetbuf->maxlen` field.

Availability
Available in CarbonLib 1.0.2 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransportProviders.h`

OTATalkGetMyZone

Obtains the AppleTalk zone name of the node on which your application is running. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTATalkGetMyZone (
   ATSvcRef ref,
   TNetbuf *zone
);
```

Parameters

ref

zone

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
The `OTATalkGetMyZone` function gets the name of your application's AppleTalk zone. If you call this function asynchronously, Open Transport calls your application's notifier with a `T_GETMYZONECOMPLETE` completion event to signal the function's completion and uses your notifier's `cookie` parameter for the zone name. More precisely, the `cookie` parameter points to a `TNetbuf` structure that in turn points to a buffer containing the zone name, which is stored as a Pascal-style string. The string can be up to 32 characters in length, so with the addition of a length byte, the buffer can have a maximum size of 33 bytes. Using a Pascal-style string for the zone name is redundant since you can determine the length of the string from the `maxlen` field of the `TNetbuf` structure, but the other zone-related calls use Pascal-style strings, so this call also uses them for consistency.

Availability
Available in CarbonLib 1.0.2 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransportProviders.h`

OTATalkGetZoneList

Obtains a list of all the zones available on the AppleTalk internet. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTATalkGetZoneList (
   ATSvcRef ref,
   TNetbuf *zones
);
```

Parameters

ref

zones

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
The OTATalkGetZoneList function returns a list of all the zones on the AppleTalk internet to which your network belongs.

If you execute this function asynchronously, Open Transport calls your notifier function with a `T_GETZONELISTCOMPLETE` completion event to signal the function's completion and uses your notifier's `cookie` parameter for the list of zones. The `cookie` parameter actually holds a pointer to a `TNetbuf` structure,

which points to a buffer containing a list of zone names, each of which is a Pascal-style string. Using a Pascal-style string for the zone name is redundant since you can determine the length of the string from the `maxlen` field of the `TNetbuf` structure, but the other zone-related calls use Pascal-style strings, so this call also uses them for consistency.

Each string can be up to 32 characters in length, and if you add a length byte, each can have a maximum size of 33 bytes. As AppleTalk internets can have a number of extended networks, you need to allocate a buffer (using the `TNetbuf->maxlen` field) that holds as much as 64 KB of memory. To keep the buffer size as small and efficient as possible, you can set up a large buffer, test for the `kOTBufferOverflowErr` error, and then increase the size of the buffer and reissue the call if this error is returned.

Availability
Available in CarbonLib 1.0.2 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransportProviders.h`

OTAtomicAdd16

Atomically adds a 16-bit value to a memory location. (Deprecated in Mac OS X v10.4.)

```
SInt16 OTAtomicAdd16 (
   SInt32 toAdd,
   SInt16 *dest
);
```

Parameters
toAdd
dest

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTAtomicAdd32

Atomically adds a 32-bit value to a memory location. (Deprecated in Mac OS X v10.4.)

```
SInt32 OTAtomicAdd32 (
   SInt32 toAdd,
   SInt32 *dest
);
```

Parameters

toAdd

dest

Availability

Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`OpenTransport.h`

OTAtomicAdd8

Atomically adds an 8-bit value to a memory location. (Deprecated in Mac OS X v10.4.)

```
SInt8 OTAtomicAdd8 (
   SInt32 toAdd,
   SInt8 *dest
);
```

Parameters

toAdd

dest

Availability

Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`OpenTransport.h`

OTAtomicClearBit

Clears a bit in a byte. (Deprecated in Mac OS X v10.4.)

```
Boolean OTAtomicClearBit (
   UInt8 *bytePtr,
   OTByteCount bitNumber
);
```

Parameters

bytePtr

bitNumber

Availability

Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

OpenTransport.h

OTAtomicSetBit

Sets a specified bit in a byte. (Deprecated in Mac OS X v10.4.)

```
Boolean OTAtomicSetBit (
   UInt8 *bytePtr,
   OTByteCount bitNumber
);
```

Parameters

bytePtr

bitNumber

Availability

Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

OpenTransport.h

OTAtomicTestBit

Tests a bit in a byte and returns its current state. (Deprecated in Mac OS X v10.4.)

```
Boolean OTAtomicTestBit (
   UInt8 *bytePtr,
   OTByteCount bitNumber
);
```

Parameters

bytePtr

bitNumber

Availability

Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`OpenTransport.h`

OTBind

Assigns an address to an endpoint. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTBind (
   EndpointRef ref,
   TBind *reqAddr,
   TBind *retAddr
);
```

Parameters

ref

reqAddr

If you specify `NIL` for the `reqAddr` parameter, Open Transport chooses a protocol address for you and requests 0 as the endpoint's maximum number of concurrent outstanding connect indications.

If you want Open Transport to assign an address for you, setthe `addr.len` field of the `TBind` structure to 0.

retAddr

You can set this parameter to `nil` if you do not care to know what address the endpoint is bound to or what the negotiated value of `qlen` is.

Return Value

A result code. See "Open Transport Result Codes" (page 354).

Discussion

You call the `OTBind` function to request an address that an endpoint be bound to. You can either use the reqAddr parameter to request that the endpoint be bound to a specific address or allow the endpoint provider to assign an address dynamically by passing nil for this parameter. Consult the documentation for the top-level protocol you are using to determine whether it is preferable to have the address assigned dynamically. The function returns the address to which the endpoint is actually bound in the retAddr parameter. This might be different from the address you requested, if you requested a specific address.

If you are binding a connection-oriented endpoint, you must use the reqAddr->qlen field to specify the number of connection requests that may be outstanding for this endpoint. The retAddr->qlen field specifies, on return, the actual number of connection requests allowed for the endpoint. This number might be smaller than the number you requested. Note that when the endpoint is actually connected, the number might be further decreased by negotiations taking place at that time.

If you call the OTBind function asynchronously and you have not installed a notifier function, the only way to determine when the function completes is to poll the endpoint using the OTGetEndpointState function. This function returns a kOTStateChangeErr until the bind completes. When the endpoint is bound, the state is either T_UNBND if the bind failed, or T_IDLE if it succeeded.

You can cancel an asynchronous bind that is still in progress by calling the OTUnbind function.

You must not bind more than one connectionless endpoint to a single address. Some connection-oriented protocols let you bind two or more endpoints to the same address. In such instances, you must use only one of the endpoints to listen for connection requests for that address. When binding the endpoint listening for a connection, you must set the reqAddr->qlen field of the OTBind function to a value greater than or equal to 1. When binding the other endpoints, you must set the reqAddr->qlen field to 0.

If you accept a connection for an endpoint that is also listening for connection requests, the address of that endpoint is deemed "busy" for the duration of the connection, and you must not bind another endpoint for listening to that same address. This requirement prevents more than one endpoint bound to the same address from accepting connection requests. If you have to bind another listening endpoint to the same address, you must first use the OTUnbind function to unbind the first endpoint or use the OTCloseProvider function to close it.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTBufferDataSize

Obtains the size of the no-copy receive buffer. (Deprecated in Mac OS X v10.4.)

```
OTByteCount OTBufferDataSize (
   OTBuffer *buffer
);
```

Parameters
buffer

Return Value
See the description of the `OTByteCount` data type.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransportProtocol.h`

OTCancelSynchronousCalls

Cancels any currently executing synchronous function for a specified provider. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTCancelSynchronousCalls (
   ProviderRef ref,
   OSStatus err
);
```

Parameters

ref

err

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
The `OTCancelSynchronousCalls` function cancels any currently executing synchronous function for the provider that you specify. The provider need not be in synchronous mode when you call this function.

Typically, you would call the `OTCancelSynchronousCalls` function at interrupt time by installing a Time Manager task that executes after a given amount of time has passed. You could do this to prevent a synchronous function from hanging the system.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTCancelTimerTask

Cancels a task that was already scheduled for execution. (Deprecated in Mac OS X v10.4.)

```
Boolean OTCancelTimerTask (
   OTTimerTask timerTask
);
```

Parameters

timerTask

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransportProtocol.h`

OTCanMakeSyncCall

Checks whether you can call a synchronous function. (Deprecated in Mac OS X v10.4.)

```
Boolean OTCanMakeSyncCall (
   void
);
```

Parameters

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTClearBit

Clears a bit atomically. (Deprecated in Mac OS X v10.4.)

```
Boolean OTClearBit (
   UInt8 *bitMap,
   OTByteCount bitNo
);
```

Parameters
bitMap
bitNo

Discussion
OTClearBit is available to client and kernel code, but only to native architecture clients.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransportProtocol.h`

OTCloneConfiguration

Copies an `OTConfiguration` structure. (Deprecated in Mac OS X v10.4.)

```
OTConfigurationRef OTCloneConfiguration (
   OTConfigurationRef cfig
);
```

Parameters

cfig

Return Value
See the description of the `OTConfigurationRef` data type.

Discussion
The `OTCloneConfiguration` function copies the `OTConfiguration` structure that you specify in the `cfig` parameter and returns a pointer to the copy. Because the internal format of an `OTConfiguration` structure is private, you must use the `OTCloneConfiguration` function to obtain two identical structures. For example, you can use this function when another application passes you a configuration structure that you want to reuse but for which you do not have the original configuration string. By cloning the structure, you have access to an additional copy of the configuration even without knowing its configuration string.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTCloseProvider

Closes a provider of any type—endpoint, mapper, or service provider. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTCloseProvider (
   ProviderRef ref
);
```

Parameters

ref

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
The `OTCloseProvider` function closes the provider that you specify in the `ref` parameter. Closing the provider deletes all memory reserved for it in the system heap, deletes its resources, and cancels any provider functions that are currently executing.

Open Transport does not guarantee that all outstanding functions have completed before it closes the provider. It is ultimately your responsibility to make sure that all provider functions that you care about have finished executing, before you close and delete a provider.

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
OpenTransport.h

OTCompareAndSwap16

Atomically compares two 16-bit values and changes one of these values if they are the same. (Deprecated in Mac OS X v10.4.)

```
Boolean OTCompareAndSwap16 (
   UInt32 oldValue,
   UInt32 newValue,
   UInt16 *dest
);
```

Parameters

oldValue

newValue

dest

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
OpenTransport.h

OTCompareAndSwap32

Atomically compares two 32-bit values and changes one of these values if they are the same. (Deprecated in Mac OS X v10.4.)

```
Boolean OTCompareAndSwap32 (
   UInt32 oldValue,
   UInt32 newValue,
   UInt32 *dest
);
```

Parameters

oldValue

newValue

dest

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
OpenTransport.h

OTCompareAndSwap8

Atomically compares two 8-bit values and changes one of these values if they are the same. (Deprecated in Mac OS X v10.4.)

```
Boolean OTCompareAndSwap8 (
   UInt32 oldValue,
   UInt32 newValue,
   UInt8 *dest
);
```

Parameters

oldValue

newValue

dest

Availability

Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

OpenTransport.h

OTCompareAndSwapPtr

Atomically compares the value of a pointer at a memory location and atomically swaps it with a second pointer value if the compare is successful. (Deprecated in Mac OS X v10.4.)

```
Boolean OTCompareAndSwapPtr (
   void *oldValue,
   void *newValue,
   void **dest
);
```

Parameters

oldValue

newValue

dest

Availability

Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

OpenTransport.h

OTCompareDDPAddresses

Compares two DDP address structures. (Deprecated in Mac OS X v10.4.)

```
Boolean OTCompareDDPAddresses (
   const DDPAddress *addr1,
   const DDPAddress *addr2
);
```

Parameters

addr1

addr2

Discussion

The `OTCompareDDPAddresses` function compares two DDP addresses for equality and returns `true` if the two addresses match. It cannot compare NBP or combined DDP-NBP addresses; using these address types always returns `false`. This function uses the zero-matches-anything AppleTalk rule when doing the matching, which means that a value of 0 in any field results in an acceptable match.

Availability

Available in CarbonLib 1.0 and later.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`OpenTransportProviders.h`

OTConnect

Requests a connection to a remote peer. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTConnect (
   EndpointRef ref,
   TCall *sndCall,
   TCall *rcvCall
);
```

Parameters

ref

sndCall

rcvCall

This parameter is only meaningful for synchronous calls to the OTConnect function. See `TCall` data type.

Return Value

A result code. See "Open Transport Result Codes" (page 354).

Discussion

If the endpoint is in synchronous mode, the OTConnect function returns after the connection is established and fills in the fields of the TCall structure (referenced by the rcvCall parameter) with the actual values associated with this connection. These might be different from the values you specified using the sndCall parameter.

If the OTConnect function returns with the kOTLookErr result, this might be either because of a pending T_LISTEN or T_DISCONNECT event. That is, either a connection request from another endpoint has interrupted execution of the function, or the remote endpoint has rejected the connection. If you don't have a notifier installed, you can call the OTLook function to identify the event that caused the kOTLookErr result. If the

event is T_LISTEN, you must accept or reject the incoming request and then continue processing the OTConnect function by calling OTRcvConnect. If the event is T_DISCONNECT, you must call the OTRcvDisconnect function to clear the error condition—that is, to deallocate memory and place the endpoint in the correct state.

If the endpoint is in asynchronous mode, the OTConnect function returns before the connection is established with a kOTNoDataErr result to indicate that the connection is in progress. When the connection is established, the endpoint provider calls your notifier, passing T_CONNECT for the code parameter. In response, you must call the OTRcvConnect function to read the connection parameters that would have been returned using the rcvCall parameter if the endpoint were in synchronous mode.

It is possible that the remote address returned in the addr field of the rcvCall parameter is not the same as the address you requested using the sndCall->addr field. This happens when the connection is accepted for a different endpoint than the one receiving the connection request.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTCountDataBytes

Returns the amount of data currently available to be read. (Deprecated in Mac OS X v10.4.)

```
OTResult OTCountDataBytes (
   EndpointRef ref,
   OTByteCount *countPtr
);
```

Parameters
ref
countPtr

Return Value
See the description of the `OTResult` data type.

Discussion
If the function returns sucessfully, the countPtr parameter points to a buffer containing the amount of data currently available to be read. This does not mean that the buffer contains all the data that was sent. That is, there might be additional data to read after you do the first read.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTCreateConfiguration

Creates a structure defining a provider's configuration. (Deprecated in Mac OS X v10.4.)

Modified

```
OTConfigurationRef OTCreateConfiguration (
   const char *path
);
```

Parameters

path

A pointer to a character string describing the provider.

Return Value

See the description of the `OTConfigurationRef` data type.

Discussion

The `OTCreateConfiguration` function creates a configuration structure that defines the software modules, hardware ports, and options that Open Transport is to use when you call a function to open a provider. This is a private structure, defined by the `OTConfiguration` data type . To create one, you use the `path` parameter to pass the `OTCreateConfiguration` function a string describing the provider service desired.

The simplest possible value of the `path` parameter is a single protocol module name of the highest-level protocol you want to use; for example, "tcp." If you do not specify a complete communications path, the Open Transport software uses default settings to construct the rest of the path. For example, if you specify "adsp" for the `path` parameter, Open Transport defaults to using the AppleTalk DataStream Protocol (ADSP) protocol module layered above the Datagram Delivery Protocol (DDP) protocol module and with LocalTalk on the default port, which is the printer port.

If you want to identify a particular port in the configuration string, you use the port name to do so (described in the section "About Port Information," beginning on page 6-5). More typically, however, you leave this value blank— for example, using a string with only "adsp" or "adsp, ddp," which configures the provider with whatever port is specified in the control panel.

To specify more than one protocol module, separate the module names with commas. You can also specify values for options by putting them in parentheses after the protocol name; for example, "adsp, ddp (Checksum=1)" specifies that ADSP is to run on top of DDP and that the checksum option is enabled.

If Open Transport cannot parse the list that you pass in the `path` parameter, the `OTCreateConfiguration` function returns `((OTConfiguration*)-1L)`. If there is insufficient memory to create an `OTConfiguration` structure, the `OTCreateConfiguration` function returns `NULL`.

The OTCreateConfiguration function returns a pointer to the configuration structure it creates. You pass this pointer as a parameter to the open-provider functions such as the OTOpenEndpoint or OTOpenMapper functions.

Availability

Modified in Carbon. Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Carbon Porting Notes

Passing inline options to `OTCreateConfiguration`-for example, `OTCreateConfiguration("tcp(NoDelay=1)")`-is not supported on Mac OS X. Instead, you should explicitly set any options using the function `OTOptionManagement`.

Declared In

`OpenTransport.h`

OTCreateDeferredTaskInContext

Creates a reference to a task that can be scheduled to run at deferred task time. (Deprecated in Mac OS X v10.4.)

```
long OTCreateDeferredTaskInContext (
   OTProcessUPP upp,
   void *arg,
   OTClientContextPtr clientContext
);
```

Parameters

upp

arg

clientContext

Availability

Available in CarbonLib 1.0 and later.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`OpenTransport.h`

OTCreatePortRef

Creates a port reference that describes a port's hardware characteristics. (Deprecated in Mac OS X v10.4.)

```
OTPortRef OTCreatePortRef (
   OTBusType busType,
   OTDeviceType devType,
   OTSlotNumber slot,
   UInt16 other
);
```

Parameters

busType

The type of bus to which the hardware port is connected; for example, a NuBus or PCI bus. See "The Port Reference" for possible values for this parameter.

devType

The type of hardware device connected to the port, such as LocalTalk or Ethernet. See "The Port Reference"for possible values for this parameter.

slot
other

The port's multiport identifier—that is, a numeric value that distinguishes between ports when more than one hardware port is connected to a given slot.

Return Value
See the description of the `OTPortRef` data type.

Discussion
The `OTCreatePortRef` function creates a port reference structure, which is a 32-bit value that describes a port's hardware characteristics: its device and bus type, its physical slot number, and, where applicable, its multiport identifier.

Once you have created a port reference, you can use the `OTFindPortByRef` function to find a specific port with that particular set of characteristics.

To create a port reference, you use the `OTCreatePortRef` function. You must know all the port's hardware characteristics: its device and bus type, its slot number, and its multiport identifier (if it has one). You cannot use wildcards to fill in any element you don't know, although you can use a device type of 0 to allow matches on every kind of device type (following the zero-matches-everything rule). Possible device and bus types are described in the section "The Port Reference."

To create a port reference for a pseudodevice, use 0 as the value for the bus type, slot number, and multiport identifier, and use the constant `kOTPseudoDevice` for the device type.

Open Transport has predefined variants of the OTCreatePortRef function for the most commonly used hardware devices, such as the NuBus, PCI, and PCMCIA devices. These three variants are listed here:

```
#define OTCreateNuBusPortRef(devType, slot, other)\
OTCreatePortRef(kOTNuBus, devType, slot, other)
#define OTCreatePCIPortRef(devType, slot, other)\
OTCreatePortRef(kOTPCIBus, devType, slot, other)
#define OTCreatePCMCIAPortRef(devType, slot, other)\
OTCreatePortRef(kOTPCMCIABus, devType, slot, other)
```

Once you have identified the port structure you want, you can access the information in its port reference, by using the `OTGetDeviceTypeFromPortRef`, `OTGetBusTypeFromPortRef`, and `OTGetSlotFromPortRef` functions.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTCreateTimerTaskInContext

Creates a task to be scheduled. (Deprecated in Mac OS X v10.4.)

```
long OTCreateTimerTaskInContext (
   OTProcessUPP upp,
   void *arg,
   OTClientContextPtr clientContext
);
```

Parameters

upp

arg

clientContext

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransportProtocol.h`

OTDelay

Delays processing for a specified number of seconds. This function is only provided for compatibility with the UNIX `sleep` function. (Deprecated in Mac OS X v10.4.)

```
void OTDelay (
   UInt32 seconds
);
```

Parameters

seconds
The number of seconds to delay.

Discussion
The `OTDelay` function delays processing for the number of seconds specified in the `seconds` parameter. While the delay is occurring, `OTDelay` continuously calls the `OTIdle` function.

You can only call the `OTDelay` function from within an application at system task time. This function is only provided for compatibility with the UNIX `sleep` function to assist with portability of UNIX code.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTDeleteName

Removes a previously registered entity name. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTDeleteName (
   MapperRef ref,
   TNetbuf *name
);
```

Parameters

ref

name

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
If the name-registration protocol defined using the config parameter to the OTOpenMapper or OTAsyncOpenMapper function supports dynamic name and address registration, you can use the OTDeleteName function to delete a registered name.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
OpenTransport.h

OTDeleteNameByID

Removes a previously registered name as specified by its name ID. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTDeleteNameByID (
   MapperRef ref,
   OTNameID nameID
);
```

Parameters

ref

nameID

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
If the name-registration protocol defined using the config parameter to the OTOpenMapper or OTAsyncOpenMapper function supports dynamic name and address registration, you can use the OTDeleteNameByID function to delete a registered name.

If the mapper is in asynchronous mode, the OTDeleteNameByID function returns immediately. When the function completes execution, the mapper provider calls the notifier function, passing T_DELNAMECOMPLETE for the code parameter, and a pointer to the id parameter in the cookie parameter.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTDequeue

Removes an element from a list. (Deprecated in Mac OS X v10.4.)

```
void * OTDequeue (
   void **listHead,
   OTByteCount linkOffset
);
```

Parameters

listHead

linkOffset

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTDestroyConfiguration

Deletes an `OTConfiguration` structure. (Deprecated in Mac OS X v10.4.)

```
void OTDestroyConfiguration (
   OTConfigurationRef cfig
);
```

Parameters

cfig

Discussion
The `OTDestroyConfiguration` function deletes the `OTConfiguration` structure that you specify in the cfig parameter and releases all associated memory.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTDestroyDeferredTask

Destroys a deferred task created with the `OTCreateDeferredTask` function. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTDestroyDeferredTask (
   OTDeferredTaskRef dtCookie
);
```

Parameters

dtCookie

Return Value

A result code. See "Open Transport Result Codes" (page 354).

Discussion

The `OTDestroyDeferredTask` function makes the `dtCookie` reference invalid and frees any resources allocated to the task when it was created. You can call this function at any time when you no longer need to schedule the deferred task object. If `dtCookie` is invalid (a value of 0), the function returns `kOTNoError` and does nothing.

If you try to destroy a deferred task that is still scheduled, the `kEAgainErr` error can occur. This is a rare situation that can only happen when you try to destroy the task from within an interrupt service routine or within another deferred task.

Availability

Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`OpenTransport.h`

OTDestroyTimerTask

Disposes of a timer task. (Deprecated in Mac OS X v10.4.)

```
void OTDestroyTimerTask (
   OTTimerTask timerTask
);
```

Parameters

timerTask

Availability

Available in CarbonLib 1.0 and later.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`OpenTransportProtocol.h`

OTDontAckSends

Specifies that a provider copy data before sending it. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTDontAckSends (
   ProviderRef ref
);
```

Parameters

ref

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
By default, providers do not acknowledge sends. You need to call the `OTDontAckSends` function only if you have used the `OTAckSends` function to turn on send-acknowledgment for a provider.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTElapsedMicroseconds

Calculates the time elapsed in microseconds since a specified time. (Deprecated in Mac OS X v10.4.)

```
UInt32 OTElapsedMicroseconds (
   OTTimeStamp *startTime
);
```

Parameters

startTime

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTElapsedMilliseconds

Calculates the time elapsed in milliseconds since a specified time. (Deprecated in Mac OS X v10.4.)

```
UInt32 OTElapsedMilliseconds (
   OTTimeStamp *startTime
);
```

Parameters

startTime

Availability

Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In

OpenTransport.h

OTEnqueue

Adds an element to a list. (Deprecated in Mac OS X v10.4.)

```
void OTEnqueue (
   void **listHead,
   void *object,
   OTByteCount linkOffset
);
```

Parameters

listHead

object

linkOffset

Availability

Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In

OpenTransport.h

OTEnterNotifier

Limits the notifications that can be sent to your notifier. (Deprecated in Mac OS X v10.4.)

```
Boolean OTEnterNotifier (
   ProviderRef ref
);
```

Parameters

ref

Availability

Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTExtractNBPName

Extracts the name part of an NBP name from an NBP entity structure. (Deprecated in Mac OS X v10.4.)

```
void OTExtractNBPName (
   const NBPEntity *entity,
   char *name
);
```

Parameters

entity

name

A pointer to the string buffer in which to store the name portion of an NBP name string that you wish to extract from the NBP entity.

Discussion
The `OTExtractNBPName` function extracts the name part of an NBP name from the specified NBP entity structure and stores it into the string buffer specified by the `name` parameter. This function inserts a backslash (\) in front of any backslash, colon (:), or at-sign (@) it finds in an NBP name so that mapper functions can use a correctly formatted NBP name.

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransportProviders.h`

OTExtractNBPType

Extracts the type part of an NBP name from an NBP entity structure. (Deprecated in Mac OS X v10.4.)

```
void OTExtractNBPType (
   const NBPEntity *entity,
   char *typeVal
);
```

Parameters

entity

typeVal

A pointer to the string buffer in which to store the type portion of an NBP name string that you wish to extract from the NBP entity.

Discussion
The `OTExtractNBPType` function extracts the type part of an NBP name from the specified NBP entity structure and stores it into the string buffer specified by the `type` parameter. This function inserts a backslash (\) in front of any backslash, colon (:), or at-sign (@) it finds in an NBP name so that mapper functions can use a correctly formatted NBP name.

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransportProviders.h`

OTExtractNBPZone

Extracts the zone part of an NBP name from an NBP entity structure. (Deprecated in Mac OS X v10.4.)

```
void OTExtractNBPZone (
   const NBPEntity *entity,
   char *zone
);
```

Parameters
entity
zone
A pointer to the string buffer in which to store the type portion of an NBP name string that you wish to extract from the NBP entity.

Discussion
The `OTExtractNBPZone` function extracts the zone part of an NBP name from the specified NBP entity structure and stores it into the string buffer specified by the `zone` parameter. This function inserts a backslash (\) in front of any backslash, colon (:), or at-sign (@) it finds in an NBP name so that mapper functions can use a correctly formatted NBP name.

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransportProviders.h`

OTFindAndRemoveLink

Finds a link in a FIFO list and removes it. (Deprecated in Mac OS X v10.4.)

```
OTLink * OTFindAndRemoveLink (
   OTList *list,
   OTListSearchUPP proc,
   const void *ref
);
```

Parameters

list

proc

ref

Return Value
See the description of the `OTLink` data type.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTFindLink

Finds a link in a FIFO list and returns a pointer to it. (Deprecated in Mac OS X v10.4.)

```
OTLink * OTFindLink (
   OTList *list,
   OTListSearchUPP proc,
   const void *ref
);
```

Parameters

list

proc

ref

Return Value
See the description of the `OTLink` data type.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTFindOption

Finds a specific option in an options buffer. (Deprecated in Mac OS X v10.4.)

```
TOption * OTFindOption (
   UInt8 *buffer,
   UInt32 buflen,
   OTXTILevel level,
   OTXTIName name
);
```

Parameters

buffer

A pointer to the buffer containing the option to be found.

buflen

The size of the buffer containing the option to be found.

level

name

Return Value

See the description of the `TOption` data type.

Discussion

Given a buffer such as might be returned by the `OTOptionManagement` function or by any endpoint function that returns a buffer containing option information, you can use the `OTFindOption` function to find a specific option in the buffer.

Availability

Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`OpenTransport.h`

OTFindPort

Obtains information about a port that corresponds to a given port name. (Deprecated in Mac OS X v10.4.)

```
Boolean OTFindPort (
   OTPortRecord *portRecord,
   const char *portName
);
```

Parameters

portName

A pointer to a port structure that contains information about the port you specified with the `portName` parameter.

portName

The name of the port about which you want information.

Discussion

The `OTFindPort` function returns information about a port that corresponds to a given port name. Each port in a system has a unique port name, which you can obtain through a previous call or set of calls to the `OTGetIndexedPort` function.

You must allocate the port structure; the function fills this structure with information about the port indicated by the `portName` parameter. If the function returns `false`, the contents of the structure are not significant.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTFindPortByRef

Obtains information about a port that corresponds to its given port reference. (Deprecated in Mac OS X v10.4.)

```
Boolean OTFindPortByRef (
   OTPortRecord *portRecord,
   OTPortRef ref
);
```

Parameters
portRecord
ref

Discussion
The `OTFindPortByRef` function returns information about a port identified by its port reference. A port reference is a 32-bit value that describes a port's hardware characteristics: its bus and device type, its physical slot number, and, where applicable, its multiport identifier. This identifier differentiates between multiple hardware ports on a given slot.

You must allocate the port structure; the function fills this structure with information about the port indicated by the `ref` parameter. If the function returns `false`, the contents of the structure are not significant.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTFree

Frees memory allocated using the OTAlloc function. (Deprecated in Mac OS X v10.4.)

```
OTResult OTFree (
   void *ptr,
   OTStructType structType
);
```

Parameters

ptr

A pointer to the structure to be deallocated. This is the pointer returned by the OTAlloc function.

structType

Return Value
See the description of the `OTResult` data type.

Discussion
In order to use the OTFree function, you must not have changed the memory allocated by the OTAlloc function for the structure specified by the structType parameter or for any of the buffers to which it points.

You are responsible for passing a structType parameter that exactly matches the type of structure being freed.

The OTFree function, along with the OTAlloc function, is provided mainly forcompatibility with XTI.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTFreeMem

Frees memory allocated with the `OTAllocMem` function. (Deprecated in Mac OS X v10.4.)

```
void OTFreeMem (
   void *mem
);
```

Parameters

mem

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTGetBusTypeFromPortRef

Extracts the value of the bus type from a port reference. (Deprecated in Mac OS X v10.4.)

```
UInt16 OTGetBusTypeFromPortRef (
   OTPortRef ref
);
```

Parameters

ref

Discussion

The `OTGetBusTypeFromPortRef` function extracts the bus type value from a port reference with unknown hardware values. You can obtain such a port reference when another application passes one to you or when you use the `OTGetIndexedPort` function to access a port structure into which another application has put its own port reference.

Availability

Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`OpenTransport.h`

OTGetClockTimeInSecs

Returns the number of seconds that have elapsed since system boot time. (Deprecated in Mac OS X v10.4.)

```
UInt32 OTGetClockTimeInSecs (
   void
);
```

Parameters

Availability

Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`OpenTransport.h`

OTGetDeviceTypeFromPortRef

Extracts the value of the hardware device type from a port reference. (Deprecated in Mac OS X v10.4.)

```
OTDeviceType OTGetDeviceTypeFromPortRef (
   OTPortRef ref
);
```

Parameters

ref

Return Value
See the description of the `OTDeviceType` data type.

Discussion
The `OTGetDeviceTypeFromPortRef` function extracts the device type value from a port reference with unknown hardware values. You can obtain such a port reference when another application passes one to you or when you use the `OTGetIndexedPort` function to access a port structure into which another application has put its own port reference.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTGetEndpointInfo

Obtains information about an endpoint that has been opened. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTGetEndpointInfo (
   EndpointRef ref,
   TEndpointInfo *info
);
```

Parameters

ref

info

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
The OTGetEndpointInfo function returns information about

- the maximum size of buffers used to specify an endpoint's address and option values
- the maximum size of normal and expedited data you can transfer using this endpoint or, for transaction-based endpoints, the maximum size of requests and replies
- the size of data you can transfer when initiating or tearing down a connection
- the services supported by the endpoint
- any additional characteristics of this endpoint

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.

Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTGetEndpointState

Obtains the current state of an endpoint. (Deprecated in Mac OS X v10.4.)

```
OTResult OTGetEndpointState (
   EndpointRef ref
);
```

Parameters

ref

Return Value
See the description of the `OTResult` data type.

Discussion
The OTGetEndpointState function returns an integer greater than or equal to 0 indicating the state of the specified endpoint. The endpoint state enumeration describes possible endpoint states and lists their decimal value.

If the function fails, it returns a negative integer specifying the error code. You must open an endpoint before you can determine its state.

You might need to know an endpoint's state in order to determine whether a function has completed or whether the endpoint is in an appropriate state for the function that you want to call next.

This function returns endpoint state information immediately, whether the endpoint is in synchronous or asynchronous mode.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTGetFirst

Returns a pointer to the first element in a FIFO list. (Deprecated in Mac OS X v10.4.)

```
OTLink * OTGetFirst (
   OTList *list
);
```

Parameters

list

Return Value
See the description of the `OTLink` data type.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTGetIndexedLink

Returns a pointer to the link at a specified position in a FIFO list. (Deprecated in Mac OS X v10.4.)

```
OTLink * OTGetIndexedLink (
   OTList *list,
   OTItemCount index
);
```

Parameters

list

index

Return Value
See the description of the `OTLink` data type.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTGetIndexedPort

Iterates through the ports available on your computer. (Deprecated in Mac OS X v10.4.)

```
Boolean OTGetIndexedPort (
   OTPortRecord *portRecord,
   OTItemCount index
);
```

Parameters

portRecord

index

Discussion

The `OTGetIndexedPort` function returns information about the ports available on your local system. To iterate through all the ports on your computer, call the function repeatedly, incrementing the `index` parameter each time (starting with 0) until the function returns false. Each time the function returns true, it fills in the port structure that you provide with information about a specific port. You can use this information, for example, when specifying a provider configuration string for the `OTCreateConfiguration` function.

You must allocate the port structure; the function fills this structure with information about the port indicated by the `index` parameter. If the function returns `false`, the contents of the structure are not significant.

Availability

Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`OpenTransport.h`

OTGetLast

Returns the last element in a FIFO list. (Deprecated in Mac OS X v10.4.)

```
OTLink * OTGetLast (
   OTList *list
);
```

Parameters

list

Return Value

See the description of the `OTLink` data type.

Availability

Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`OpenTransport.h`

OTGetNBPEntityLengthAsAddress

Obtains the size of an NBP entity structure. (Deprecated in Mac OS X v10.4.)

```
OTByteCount OTGetNBPEntityLengthAsAddress (
   const NBPEntity *entity
);
```

Parameters

entity

Return Value

See the description of the `OTByteCount` data type.

Discussion

The `OTGetNBPEntityLengthAsAddress` function obtains the number of bytes needed to store an NBP entity structure into an NBP or combined DDP-NBP address structure.

Availability

Available in CarbonLib 1.0 and later.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`OpenTransportProviders.h`

OTGetProtAddress

Obtains the address to which an endpoint is bound and, if the endpoint is currently connected, obtains the address of its peer. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTGetProtAddress (
   EndpointRef ref,
   TBind *boundAddr,
   TBind *peerAddr
);
```

Parameters

ref

boundAddr

If you are calling this function only to determine the address of the peer endpoint, you can set the boundAddr parameter to NIL.

The boundAddr->qlen field is ignored. See `EndpointRef` data type.

peerAddr

Return Value

A result code. See "Open Transport Result Codes" (page 354).

Discussion

The OTGetProtAddress function returns the address to which an endpoint is bound in the boundAddr parameter and, if the endpoint is currently connected, the address of its peer in the peerAddr parameter. Not all endpoints support this function. A value of T_XPG4_1 in the flags field of the `TEndpointInfo` (page 174) structure indicates that the endpoint does support this function.

You are responsible for initializing the buffers required to hold the local and peer addresses. The addr field of the `TEndpointInfo` structure specifies the maximum amount of memory needed to store the address of an endpoint. Use this value to set the size of the buffers.

The information returned by the OTGetProtAddress function is affected by the state of the endpoint specified by the ref parameter. If the endpoint is in the T_UNBND state, the boundAddr->addr.len field is set to 0. If the endpoint is not in the T_DATAXFER state, the peerAddr->addr.len field is set to 0.

If the endpoint is in asynchronous mode and a notifier is not installed, it is not possible to determine when the function completes.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTGetSlotFromPortRef

Extracts slot information from a port reference. (Deprecated in Mac OS X v10.4.)

```
OTSlotNumber OTGetSlotFromPortRef (
   OTPortRef ref,
   UInt16 *other
);
```

Parameters

ref

other

A pointer to a 16-bit buffer you provide into which the function places a value that distinguishes between ports when more than one hardware port is connected to a given slot. Specify `NULL` for this parameter if you do not want the function to return this information.

Return Value
See the description of the `OTSlotNumber` data type.

Discussion
The `OTGetSlotFromPortRef` function extracts slot information from a port reference with unknown hardware values. You can obtain such a port reference when another application passes one to you or when you use the `OTGetIndexedPort` function to access a port structure into which another application has put its own port reference.

Note that the slot numbers are physical; that is, they are the slot numbers returned by the Slot Manager and not the slots seen in various network configuration applications. Physical slot numbers depend on the type of card installed. For example, NuBus cards number their slots 9–13, which appear in the AppleTalk or TCP control panels as slots 1–5.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTGetTimeStamp

Obtains the current timestamp. (Deprecated in Mac OS X v10.4.)

```
void OTGetTimeStamp (
   OTTimeStamp *currentTime
);
```

Parameters

currentTime

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTIdle

Idles your computer. (Deprecated in Mac OS X v10.4.)

```
void OTIdle (
   void
);
```

Discussion
You can call the `OTIdle` function while you are waiting for asynchronous provider operations to complete. It is not necessary for the correct operation of Open Transport to call this function, but it provides compatibility for existing programs that use an idling function.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTInetAddressToName

Determines the canonical domain name of the host associated with an internet address. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTInetAddressToName (
   InetSvcRef ref,
   InetHost addr,
   InetDomainName name
);
```

Parameters

ref

addr

name

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
If you call this function asynchronously, the TCP/IP service provider calls your notifier function with the `T_DNRADDRTONAMECOMPLETE` completion event code when the function completes. The `cookie` parameter to the notifier function contains a pointer to the `InetHost` structure you specified in the `addr` parameter. If you had more than one simultaneous outstanding call to the `OTInetAddressToName` function, you can use this information to determine which call has completed execution.

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransportProviders.h`

OTInetGetInterfaceInfo

Returns internet address information about the local host. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTInetGetInterfaceInfo (
   InetInterfaceInfo *info,
   SInt32 val
);
```

Parameters

info

val

An index into the local host's array of configured IP interfaces. Specify 0 for information about the first interface. Specify `kDefaultInetInterface` to get information about the primary interface.

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
Because the architecture of Open Transport TCP/IP provides for multihoming, in principle a given host can receive packets simultaneously through more than one network interface. For each IP interface configured for the local host, the `OTInetGetInterfaceInfo` function provides the internet address and subnet mask, a default gateway (that is, a gateway, if any exists, that can be used to route any packet to all destinations outside the locally connected subnet), and a domain name server, if any is known. The function also returns

the version number of the `OTInetGetInterfaceInfo` function and, if available, the broadcast address for each interface. If the broadcast address is not available, you can determine it from the internet address and subnet mask.

Because multihoming has not been implemented in the initial release of Open Transport, the `OTInetGetInterfaceInfo` function never returns information for more than one interface.

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransportProviders.h`

OTInetGetSecondaryAddresses

Returns the active secondary IP addresses. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTInetGetSecondaryAddresses (
   InetHost *addr,
   UInt32 *count,
   SInt32 val
);
```

Parameters
addr
count
val

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransportProviders.h`

OTInetHostToString

Converts an an address in InetHost format into a character string in dotted-decimal notation. (Deprecated in Mac OS X v10.4.)

```
void OTInetHostToString (
   InetHost host,
   char *str
);
```

Parameters

host

str

A pointer to a C string containing an IP address in dotteddecimal notation (for example, "12.13.14.15"). You must allocate storage for this string and provide the pointer to the function.

Availability

Available in CarbonLib 1.0 and later.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`OpenTransportProviders.h`

OTInetMailExchange

Returns mail-exchange-host names and preference information for a domain name you specify. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTInetMailExchange (
   InetSvcRef ref,
   char *name,
   UInt16 *num,
   InetMailExchange *mx
);
```

Parameters

ref

name

A pointer to a host name, partially qualified domain name, or fully qualified domain name for which you want mail exchange information.

num

A pointer to the number of elements in the array pointed to by the `mx` parameter. When the function completes, it sets the number pointed to by the `num` parameter to the actual number of elements filled in.

mx

Return Value

A result code. See "Open Transport Result Codes" (page 354).

Discussion

In order to deliver mail, a mail application must determine the fully qualified domain name of the host to which the mail should be sent. That host might be the final destination of the mail, a mail server, or a router. The domain name system servers maintain mail-exchange resource records that pair domain names with the hosts that can accept mail for that domain. Each domain name can be paired with any number of host

names; each record containing such a pair also contains a preference number. The mailer sends the mail to the host with the lowest preference number first and tries the others in turn until the mail is delivered or until the mailer decides that the mail is undeliverable.

The `OTInetMailExchange` function returns mail-exchange-host and preference information for the domain name you specify. You must then determine the address of the host and how best to deliver the mail. You can specify as many elements to the array of `InetMailExchange` structures as you wish.

If you call this function asynchronously, the TCP/IP service provider calls your notifier function with the `T_DNRMAILEXCHANGECOMPLETE` completion event code when the function completes. The `cookie` parameter to the notifier function contains the array pointer you specified in the `mx` parameter. If you had more than one simultaneous outstanding call to the `OTInetMailExchange` function, you can use this information to determine which call has completed execution.

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransportProviders.h`

OTInetQuery

Executes a generic DNS query. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTInetQuery (
   InetSvcRef ref,
   char *name,
   UInt16 qClass,
   UInt16 qType,
   char *buf,
   OTByteCount buflen,
   void **argv,
   OTByteCount argvlen,
   OTFlags flags
);
```

Parameters
ref
name
qClass
qType
buf
buflen
argv
argvlen
flags

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransportProviders.h`

OTInetStringToAddress

Resolves a domain name to its equivalent internet addresses. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTInetStringToAddress (
   InetSvcRef ref,
   char *name,
   InetHostInfo *hinfo
);
```

Parameters

ref

name

A pointer to the domain name you want to resolve. This can be a host name, a partially qualified domain name, a fully qualified domain name, or an internet address in dotted-decimal format.

hinfo

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
Because the architecture of Open Transport TCP/IP provides for multihoming, a single host can be associated with multiple internet addresses. You can use the `OTInetStringToAddress` function to return multiple addresses for multihomed hosts.

Because multihoming has not been implemented in the initial release of Open Transport, the `OTInetStringToAddress` function never returns more than one address.

If you specify an internet address in dotted-decimal format for the `hinfo` parameter, the `OTInetStringToAddress` function places that address in the `InetHostInfo.name` field instead of a canonical domain name.

If you call the `OTInetStringToAddress` function asynchronously, the TCP/IP service provider calls your notifier function with the `T_DNRSTRINGTOADDRCOMPLETE` completion event code when the function completes. The `cookie` parameter to the notifier function contains the pointer you specified in the `hinfo` parameter. If you had more than one simultaneous outstanding call to the `OTInetStringToAddress` function, you can use this information to determine which call has completed execution.

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
OpenTransportProviders.h

OTInetStringToHost

Converts an IP address string from dotted-decimal notation or hexadecimal notation to an `InetHost` data type. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTInetStringToHost (
   const char *str,
   InetHost *host
);
```

Parameters

str

A pointer to a character string containing an IP address in either dotted-decimal notation (for example, "12.13.14.15") or hexadecimal notation (for example, "0x0c0d0e0f").

host

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
OpenTransportProviders.h

OTInetSysInfo

Returns details about a host's processor and operating system. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTInetSysInfo (
   InetSvcRef ref,
   char *name,
   InetSysInfo *sysinfo
);
```

Parameters

ref

name

The name of the host about which you want information. This can be a host name (including the local host), a partially qualified domain name, or a fully qualified domain name.

sysinfo

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
The information returned by this function is maintained by the domain name server. If you call this function asynchronously, the TCP/IP service provider calls your notifier function with the `T_DNRSYSINFOCOMPLETE` completion event code when the function completes. The `cookie` parameter to the notifier function contains a pointer to the `InetSysInfo` structure you specified in the `sysinfo` parameter. If you had more than one simultaneous outstanding call to the `OTInetSysInfo` function, you can use this information to determine which call has completed execution.

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransportProviders.h`

OTInitDDPAddress

Initializes a DDP address structure. (Deprecated in Mac OS X v10.4.)

```
void OTInitDDPAddress (
   DDPAddress *addr,
   UInt16 net,
   UInt8 node,
   UInt8 socket,
   UInt8 ddpType
);
```

Parameters

addr

net
The network number you wish to specify. Set to 0 to default to the local network.

node
The node ID you wish to specify. Set to 0 to default to the local node.

socket
The socket number you wish to specify. Set to 0 to allow Open Transport to assign a socket dynamically when you use this address to bind an endpoint.

ddpType
The DDP type you wish to specify. Set to 0 unless you are using DDP.

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransportProviders.h`

OTInitDDPNBPAddress

Initializes a combined DDP-NBP address structure. (Deprecated in Mac OS X v10.4.)

```
OTByteCount OTInitDDPNBPAddress (
   DDPNBPAddress *addr,
   const char *name,
   UInt16 net,
   UInt8 node,
   UInt8 socket,
   UInt8 ddpType
);
```

Parameters

addr

name

A pointer to the NBP string you wish to use for the NBP name.

net

The network number you wish to specify. Set to 0 to default to the local network.

node

The node ID you wish to specify. Set to 0 to default to the local node.

socket

The socket number you wish to specify. Set to 0 to allow Open Transport to assign a socket dynamically when you use this address to bind an endpoint.

ddpType

The DDP type you wish to specify. Set to 0 unless you are using DDP.

Return Value

See the description of the `OTByteCount` data type.

Discussion

The `OTInitDDPNBPAddress` function initializes a combined DDP-NBP address structure with the data provided in the parameters: NBP name, network number, node ID, socket number, and DDP type. The function returns the total size of the address structure, which is the length of the `name` parameter plus the size of a `DDPAddress` structure.

Availability

Available in CarbonLib 1.0 and later.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`OpenTransportProviders.h`

OTInitDNSAddress

Fills in a DNSAddress structure with the data you provide. (Deprecated in Mac OS X v10.4.)

```
OTByteCount OTInitDNSAddress (
   DNSAddress *addr,
   char *str
);
```

Parameters

addr

str

A pointer to a domain name string. This string can be just a host name (otteam), a partially qualified domain name (for example, "otteam.ssw"), a fully qualified domain name (for example, "otteam.ssw.apple.com."), or an internet address in dotteddecimal format (for example, "17.202.99.99"), and can optionally include the port number (for example, "otteam.ssw.apple.com:25" or "17.202.99.99:25").

Return Value
See the description of the `OTByteCount` data type.

Discussion
This function fills in the `fAddressType` field of the `DNSAddress` structure with the value `AF_DNS`, fills in the `fName` field with the address string you specify, and returns the size of the resulting `DNSAddress` structure as an unsigned integer. You can use the `DNSAddress` structure to provide an address when you use a UDP or TCP endpoint. If you do so, the domain name resolver resolves the address for you automatically.

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransportProviders.h`

OTInitInetAddress

Fills in an InetAddress structure with the data you provide. (Deprecated in Mac OS X v10.4.)

```
void OTInitInetAddress (
   InetAddress *addr,
   InetPort port,
   InetHost host
);
```

Parameters

addr

port

host

Discussion
This function fills in the `fAddressType` field of the `InetAddress` structure with the value `AF_INET`. You use the InetAddress structure when providing a TCP or UDP address to the Open Transport functions `OTConnect`, `OTSndURequest`, and `OTBind`. You are not required to use the `OTInitInetAddress` function when creating an `InetAddress` structure; this function is provided for your convenience only.

Availability
Available in CarbonLib 1.0 and later.

Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransportProviders.h`

OTInitNBPAddress

Initializes an NBP address structure. (Deprecated in Mac OS X v10.4.)

```
OTByteCount OTInitNBPAddress (
   NBPAddress *addr,
   const char *name
);
```

Parameters

addr

name

A pointer to the NBP string you wish to use for the NBP name.

Return Value
See the description of the `OTByteCount` data type.

Discussion
The `OTInitNBPAddress` function can be used to initialize an NBP address structure with the NBP name specified in the `name` parameter, which is assumed to already be in the correct string format. The function returns the size of the NBP address structure, which is the size of the `fAddressType` field plus the length of the string in the `name` parameter.

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransportProviders.h`

OTInitNBPEntity

Initializes an NBP entity structure. (Deprecated in Mac OS X v10.4.)

```
void OTInitNBPEntity (
   NBPEntity *entity
);
```

Parameters

entity

Discussion
The `OTInitNBPEntity` function initializes an NBP entity structure, setting the name, type and zone parts of an NBP name to empty strings.

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransportProviders.h`

OTInstallNotifier

Installs a notifier function. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTInstallNotifier (
   ProviderRef ref,
   OTNotifyUPP proc,
   void *contextPtr
);
```

Parameters

ref

proc

For C++ applications, the `proc` parameter must point to either a C function or a static member function. See `OTNotifyUPP` data type.

contextPtr

A context pointer for your use. The provider passes this value unchanged to your notifier function when it calls the function.

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
The `OTInstallNotifier` function installs a notifier function for the provider that you specify. Changing a provider's mode of execution does not affect the notifier function. The notifier function remains installed until you remove it using the `OTRemoveNotifier` function or until you close the provider.

Before calling the `OTInstallNotifier` function, you must open the provider for which you want to install the notifier. If you open a provider asynchronously (for example, with the `OTAsyncOpenEndpoint` function), you must pass a pointer to a notifier function as a parameter to the function used to open the provider. In this case, you don't need to call the `OTInstallNotifier` function unless you want to install a different notifier function. If you do, you must call the `OTRemoveNotifier` function before calling the `OTInstallNotifier` function.

Opening a provider synchronously (for example, with the `OTOpenEndpoint` function) opens the provider but does not install a notifier function for it. If you need a notifier function for a provider opened synchronously, you must call the `OTInstallNotifier` function. This notifier would not return completion events, but would return asynchronous events advising you of the arrival of data, of changes in flow-control restrictions, and so on.

Call the `OTInstallNotifier` function only when no provider functions are executing for the provider that you specify. Otherwise, the `OTInstallNotifier` function returns the result code `kOTStateChangeErr`.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTIoctl

Sends a module-specific command to an Open Transport protocol module. (Deprecated in Mac OS X v10.4.)

```
SInt32 OTIoctl (
   ProviderRef ref,
   UInt32 cmd,
   void *data
);
```

Parameters

ref

cmd

A routine selector for the module-specific command.

data

Data to be used by the module-specific command, or a pointer to such data. The interpretation of the `data` parameter is command specific.

Discussion
The `OTIoctl` function sends a module-specific command to an Open Transport protocol module. The `OTIoctl` function runs synchronously or asynchronously, matching the provider's mode of execution.

If the `OTIoctl` function completes synchronously without error, it returns 0 or a positive integer. The positive integer's meaning is command specific. If the `OTIoctl` function fails while executing synchronously, its return value is a negative integer corresponding to an Open Transport result code.

If the `OTIoctl` function runs asynchronously, it returns immediately with a return value `kOTNoError` or another Open Transport result code. When the function completes execution, Open Transport calls the notifier function you specify, passing the event code `kStreamIoctlEvent` and a `result` parameter indicating the result of the completed `OTIoctl` function. If the value of the `result` parameter is greater than 0, the corresponding result code is defined by the command; otherwise, the value of the `result` parameter corresponds to an Open Transport result code.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTIsAckingSends

Determines whether a provider is acknowledging sends. (Deprecated in Mac OS X v10.4.)

```
Boolean OTIsAckingSends (
   ProviderRef ref
);
```

Parameters

ref

Discussion

The `OTIsAckingSends` function returns `true` if the provider acknowledges sends and `false` if it does not.

Availability

Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In

`OpenTransport.h`

OTIsBlocking

Returns a boolean indicating whether a provider is blocking. (Deprecated in Mac OS X v10.4.)

```
Boolean OTIsBlocking (
   ProviderRef ref
);
```

Parameters

ref

Availability

Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In

`OpenTransport.h`

OTIsInList

Determines whether the specified link is in the specified list. (Deprecated in Mac OS X v10.4.)

```
Boolean OTIsInList (
   OTList *list,
   OTLink *link
);
```

Parameters

list

link

Availability

Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`OpenTransport.h`

OTIsSynchronous

Returns a provider's current mode of execution. (Deprecated in Mac OS X v10.4.)

```
Boolean OTIsSynchronous (
   ProviderRef ref
);
```

Parameters

ref

Discussion

The `OTIsSynchronous` function returns `true` if a provider is in synchronous mode or returns `false` if the provider is in asynchronous mode.

Availability

Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`OpenTransport.h`

OTLeaveNotifier

Allows Open Transport to resume sending primary and completion events. (Deprecated in Mac OS X v10.4.)

```
void OTLeaveNotifier (
   ProviderRef ref
);
```

Parameters

ref

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTLIFODequeue

Removes the first link in a LIFO list and returns a pointer to it. (Deprecated in Mac OS X v10.4.)

```
OTLink * OTLIFODequeue (
   OTLIFO *list
);
```

Parameters

list

Return Value
See the description of the `OTLink` data type.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTLIFOEnqueue

Places a link at the front of a LIFO list. (Deprecated in Mac OS X v10.4.)

```
void OTLIFOEnqueue (
   OTLIFO *list,
   OTLink *link
);
```

Parameters

list

link

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.

Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTLIFOStealList

Removes all links in a LIFO list and returns a pointer to the first link in the list. (Deprecated in Mac OS X v10.4.)

```
OTLink * OTLIFOStealList (
   OTLIFO *list
);
```

Parameters
list

Return Value
See the description of the `OTLink` data type.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTListen

Listens for an incoming connection request. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTListen (
   EndpointRef ref,
   TCall *call
);
```

Parameters
ref
call

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
You use the OTListen function to listen for incoming connection requests. On return, the function fills in the TCall structure referenced by the call parameter with information about the connection request. After retrieving the connection request using the OTListen function, you can reject the request using the OTSndDisconnect function, or you can accept the request using the OTAccept function.

If the endpoint is in synchronous mode and is blocking, the OTListen function returns when a connection request has arrived. If the endpoint is in asynchronous mode or is not blocking, the OTListen function returns any pending connection requests or returns the kOTNoDataErr result if there are no pending connection requests. You can also call the OTListen function from within a notifier function in response to the T_LISTEN event. In this case, the function returns a result immediately.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTLook

Determines the current asynchronous event pending for an endpoint. (Deprecated in Mac OS X v10.4.)

```
OTResult OTLook (
   EndpointRef ref
);
```

Parameters
ref

Return Value
See the description of the `OTResult` data type.

Discussion
You use the OTLook function in one of two cases. First, if the endpoint is in synchronous mode, you can call the OTLook function to poll for incoming data or connection requests. Second, certain asynchronous events might cause a synchronous function to fail with the result kOTLookErr. For example, if you call OTAccept and the endpoint gets a T_DISCONNECT event, the OTAccept function returns with kOTLookErr. In this case, you need to call the OTLook function to determine what event caused the original function to fail. Table 3-7 on page 3-26 lists the functions that might return the kOTLookErr result and the events that can cause these functions to fail.

The OTLook function returns an integer value that specifies the asynchronous event pending for the endpoint specified by the ref parameter. On error, OTLook returns a negative integer corresponding to a result code.

If there are multiple events pending, the OTLook function first looks for one ofthe following events: T_LISTEN, T_CONNECT, T_DISCONNECT, T_UDERR, or T_ORDREL. If it finds more than one of these, it returns them to you in first-in, first-out order. After processing these events, the OTLook function looks for the T_DATA, T_REQUEST, and T_REPLY events. If it finds more than one of these, it returns them to you in first-in, first-out order. You cannot use the OTLook function to poll for completion events.

Unless you are operating exclusively in synchronous mode, it is recommended that you use notifier functions to get information about pending events for an endpoint.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTLookupName

Finds and returns all addresses that correspond to a particular name or name pattern, or confirms that a name is registered. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTLookupName (
   MapperRef ref,
   TLookupRequest *req,
   TLookupReply *reply
);
```

Parameters

ref

req

reply

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
You can use the OTLookupName function to find out whether a name is registered and what address is associated with that name. You use the req parameter to supply the information needed for the search: what name should be looked up and, optionally, what node contains that information, how many matches you expect to find, and how long the search should continue before the function returns. On return, the reply parameter contains the names field that points to the buffer where the matching entries are stored and the rspcount field that specifies the number of matching entries.

For each registered name found, the OTLookupName function stores the following information in the buffer referenced by the names field of the reply parameter:

```
unsigned short addrLen; /* length of address that follows*/
unsigned short nameLen; /* length of name that follows */
unsigned char addr[]; /* address */
unsigned char name[]; /* name, padded to quad-word boundary*/
```

If you are searching for names using a name pattern and you expect that more than one name will be returned to you, you need to parse the reply buffer to extract the matching names.

If you call the OTLookupName function asynchronously, the mapper provider calls your notifier function passing one of two completion codes for the code parameter (T_LKUPNAMERESULT or T_LKUPNAMECOMPLETE) and passing the reply parameter in the cookie parameter. The mapper provider passes the T_LKUPNAMERESULT code each time it stores a name in the reply buffer, and it passes the T_LKUPNAMECOMPLETE code when it is done. When you receive this event, examine the rspcount field to determine whether there is a last name to retrieve from the reply buffer. The use of both codes is a feature that gives you a choice about how to process multiple names when searching for names matching a pattern.

- If you decide to allocate a buffer that is large enough to contain all the names returned, you can ignore the T_LKUPNAMERESULT code and call a function that parses the buffer once the OTLookupName function has completed—that is, once the provider calls your notifier function using the T_LKUPNAMECOMPLETE event.

- If you want to save memory or if you don't know how large a buffer to allocate, you can use the following method to process the names returned. Each time that the T_LKUPNAMERESULT event is passed, you must do something with the reply from the reply buffer. You can copy it somewhere, or you can delete it if it isn't a name you're interested in. Then, from inside your notifier you must set the reply->names.len field or the reply->rspcount field back to 0 (thus allowing the mapper provider to overwrite the original name). This tells the mapper provider that you are ready to receive another name. Accordingly, when the mapper provider has inserted another name into your reply buffer, it calls your notifier passing the T_LKUPNAMERESULT code, and you can process the new entry as you have processed the first entry. This method also saves you the trouble of having to parse through the buffer to extract name and address information.

The cookie parameter to the notifier contains the reply parameter.

The format of the names and protocol addresses are specific to the underlying protocol. Consult the documentation supplied for your protocol for more information.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTMemcmp

Compares the contents of two memory locations. (Deprecated in Mac OS X v10.4.)

```
Boolean OTMemcmp (
   const void *mem1,
   const void *mem2,
   OTByteCount nBytes
);
```

Parameters
mem1
mem2
nBytes

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTMemcpy

Copies data from one memory location to another; the source and destination locations must not overlap. (Deprecated in Mac OS X v10.4.)

```
void OTMemcpy (
   void *dest,
   const void *src,
   OTByteCount nBytes
);
```

Parameters

dest

src

nBytes

Availability

Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

OpenTransport.h

OTMemmove

Copies data from one memory location to another; the source and destination locations may overlap. (Deprecated in Mac OS X v10.4.)

```
void OTMemmove (
   void *dest,
   const void *src,
   OTByteCount nBytes
);
```

Parameters

dest

src

nBytes

Availability

Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

OpenTransport.h

OTMemset

Sets the specified memory range to a specific value. (Deprecated in Mac OS X v10.4.)

```
void OTMemset (
   void *dest,
   OTUInt8Param toSet,
   OTByteCount nBytes
);
```

Parameters

dest

toSet

nBytes

Availability

Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`OpenTransport.h`

OTMemzero

Initializes the specified memory range to 0. (Deprecated in Mac OS X v10.4.)

```
void OTMemzero (
   void *dest,
   OTByteCount nBytes
);
```

Parameters

dest

nBytes

Availability

Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`OpenTransport.h`

OTNextOption

Locates the next `TOption` structure in a buffer. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTNextOption (
   UInt8 *buffer,
   UInt32 buflen,
   TOption **prevOptPtr
);
```

Parameters

buffer

A pointer to the buffer containing the option to be found.

buflen

A long specifying the size of the buffer containing the option to be found.

prevOptPtr

Return Value

A result code. See "Open Transport Result Codes" (page 354).

Discussion

The `OTNextOption` function allows you to parse through a buffer containing `TOption` structures describing an endpoint's option values. Within the buffer, `TOption` structures are aligned to long-word boundaries. This function takes into account this padding when it calculates the beginning address of the next `TOption` structure and it returns that address in the `prevOptPtr` parameter.

The first time you call the option, set the `prevOptPtr` parameter to the beginning address of the buffer. When the function returns, the `prevOptPtr` parameter points to the next (second) option in the buffer. You can continue this process, specifying the value returned for the `prevOptPtr` parameter by the previous invocation of the function, each time you call the function to obtain the beginning address of each option in the buffer.

Availability

Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`OpenTransport.h`

OTOpenAppleTalkServicesInContext

Opens a synchronous AppleTalk service provider. (Deprecated in Mac OS X v10.4.)

```
ATSvcRef OTOpenAppleTalkServicesInContext (
   OTConfigurationRef cfig,
   OTOpenFlags flags,
   OSStatus *err,
   OTClientContextPtr clientContext
);
```

Parameters

cfig

flags

err

clientContext

Return Value
See the description of the `ATSvcRef` data type.

Discussion
Applications may pass a NULL context pointer but nonapplications must always pass a valid client context pointer.

You receive a client context pointer when you call the function `InitOpenTransportInContext` (page 362).

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransportProviders.h`

OTOpenEndpointInContext

Opens an endpoint that operates synchronously. (Deprecated in Mac OS X v10.4.)

```
EndpointRef OTOpenEndpointInContext (
   OTConfigurationRef config,
   OTOpenFlags oflag,
   TEndpointInfo *info,
   OSStatus *err,
   OTClientContextPtr clientContext
);
```

Parameters

config

oflag

info

err

clientContext

Return Value
See the description of the `EndpointRef` data type.

Availability
Available in CarbonLib 1.0 and later.

Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTOpenInternetServicesInContext

Opens the TCP/IP service provider and returns an internet services reference. (Deprecated in Mac OS X v10.4.)

```
InetSvcRef OTOpenInternetServicesInContext (
   OTConfigurationRef cfig,
   OTOpenFlags oflag,
   OSStatus *err,
   OTClientContextPtr clientContext
);
```

Parameters
cfig
oflag
err
clientContext

Return Value
See the description of the `InetSvcRef` data type.

Discussion
Applications may pass a NULL context pointer but nonapplications must always pass a valid client context pointer.

You receive a client context pointer when you call the function `InitOpenTransportInContext` (page 362).

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransportProviders.h`

OTOpenMapperInContext

Creates a synchronous mapper provider and returns a mapper reference. (Deprecated in Mac OS X v10.4.)

```
MapperRef OTOpenMapperInContext (
   OTConfigurationRef config,
   OTOpenFlags oflag,
   OSStatus *err,
   OTClientContextPtr clientContext
);
```

Parameters

config

oflag

err

clientContext

Return Value
See the description of the `MapperRef` data type.

Discussion
Applications may pass a NULL pointer but non-applications must always pass a valid client context pointer.

You receive a client context pointer when you call the function `InitOpenTransportInContext` (page 362).

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTOptionManagement

Determines an endpoint's current or default option values or changes these values. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTOptionManagement (
   EndpointRef ref,
   TOptMgmt *req,
   TOptMgmt *ret
);
```

Parameters

ref

req

ret

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
To use the `OTOptionManagement` function, you must have opened an endpoint using the `OTOpenEndpoint` or `OTAsyncOpenEndpoint` functions.

You use the `OTOptionManagement` function to negotiate, retrieve, or verify an endpoint's protocol options. If the endpoint is in asynchronous mode and you have not installed a notifier function, it is not possible to determine when the function completes.

The action taken by the `OTOptionManagement` function is determined by the setting of the `req->flags` field. The following bulleted items describe the different operations that you can perform and the flag settings that you use to specify these operations.

- To negotiate values for the endpoint, you must call the `OTOptionManagement` function, specifying `T_NEGOTIATE` for the `req->flags` field. The endpoint provider evaluates the requested options, negotiates the values, and returns the resulting values in the option management structure pointed to by the `ret->opt.buf` field. The `status` field of each returned option is set to a constant that indicates the result of the negotiation. These constants are described by the "Open Transport Flags and Status Codes" (page 334) enumeration.

  For any protocol specified, you can negotiate for the default values of all options supported by the endpoint by specifying the value `T_ALLOPT` for the `name` field of the `TOption` structure. This might be useful if you want to change current settings or if negotiations for other values have failed. The success of the negotiations depends partly on the state of the endpoint—that is, simply because these are default values does not guarantee a completely successful negotiation. When the function returns, the resulting values are returned, option by option, in the buffer pointed to by the `ret->opt.buf` field.

- To retrieve an endpoint's default option values, call the `OTOptionManagement` function, specifying `T_DEFAULT` for the `req->flags` field. You must also specify the name of the option (but not its value) in the `TOption` structure that you create for each of the options you are interested in.

  When the function returns, it passes the default values for the options back to you in the buffer pointed to by the `ret->opt.buf` field. For each option, the `status` field contains `T_NOTSUPPORT` if the protocol does not support the option, `T_READONLY` if the option is read-only, and `T_SUCCESS` in all other cases. The overall result of the request is returned in the `ret->flags` field. The meaning of this result is described by the `Open Transport Flags and Status Codes` enumeration.

  When getting an endpoint's default option values, you can specify `T_ALLOPT` for the option name. This returns all supported options for the specified level with their default values. In this case, you must set the `opt.maxlen` field to the maximum size required to hold an endpoint's option information. The `info.opt` field of the `TEndpointInfo` (page 174) structure specifies the maximum size of a buffer used to hold option information for an endpoint.

- To retrieve an endpoint's current option values, call the `OTOptionManagement` function, specifying `T_CURRENT` for the `req->flags` field. For each option in the buffer referenced by the `req->opt.buf` field, specify the name of the option you are interested in. The function ignores any option valuesyou specify.

  When the function returns, it passes the current values for the options back to you in the buffer referenced by the `ret->opt.buf` field. For each option, the `status` field contains `T_NOTSUPPORT` if the protocol does not support the option, `T_READONLY` if the option is read-only, and `T_SUCCESS` in all other cases. The overall result of the request is returned in the `ret->flags` field. The meaning of this result is described by the "Open Transport Flags and Status Codes" (page 334) enumeration.

  When retrieving an endpoint's current option values, you can specify `T_ALLOPT` for the option name. The function returns all supported options for the specified protocol, with their current values. In this case, you must set the `opt.maxlen` field to the maximum size required to hold an endpoint's option information. The `info.opt` field of the `TEndpointInfo` structure specifies the maximum size of a buffer used to hold option information for an endpoint.

- To check whether an endpoint provider supports certain options or option values, you must call the OTOptionManagement function, specifying T_CHECK for the req->flags field. Checking options or their values does not change the current settings of an endpoint's options.
  - To check whether an option is supported, set the `name` field of the `TOption` structure to the option name, but do not specify an option value. When the function returns, the `status` field for the corresponding `TOption` structure in the buffer pointed to by the `ret->opt.buf` field is set to `T_SUCCESS` if the option is supported, `T_NOTSUPPORT` if it is not supported or needs additional client privileges, and `T_READONLY` if it is read-only.
  - To check whether an option value is supported, set the `name` field of the `TOption` structure to the option name, and set the `value` field to the value you want to check. When the function returns, the `status` field for the corresponding `TOption` structure in the buffer pointed to by the `ret->opt.buf` field is set as it would be if you had specified the `T_NEGOTIATE` flag. The overall result of the option checks is returned in the `ret->flags` field, which contains the single worst result of the option checks. The meaning of this result is described by the `Open Transport Flags and Status Codes` enumeration.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTRcv

Reads data sent using a connection-oriented transactionless protocol. (Deprecated in Mac OS X v10.4.)

```
OTResult OTRcv (
   EndpointRef ref,
   void *buf,
   OTByteCount nbytes,
   OTFlags *flags
);
```

Parameters

ref

buf

A pointer to a memory location where the incoming data is to be copied. You must allocate this buffer before you call the function.

nbytes

flags

Return Value
See the description of the `OTResult` data type.

Discussion
You call the `OTRcv` function to read data sent by the peer to which you are connected. If the `OTRcv` function succeeds, it returns an integer (`OTStatus`) specifying the number of bytes received. The function places the data read into the buffer referenced by the buf parameter. If the function fails, it returns a negative integer corresponding to a result code that indicates the reason for the failure. You can call this function to receive either normal or expedited data. If the data is expedited, the `T_EXPEDITED` flag is set in the flags parameter.

If `T_MORE` is set in the flags parameter when the function returns, this means that the buffer you allocated is too small to contain the data to be read and that you must call the `OTRcv` function again. If you have read x bytes with the first call, the next call to the `OTRcv` function begins to read at the (x + 1) byte. Of course, if you need it, you must copy the data in the buffer to another location before calling the function again. Each call to this function that returns with the `T_MORE` flag set means that you must call the function again to get more data. When you have read all the data, the OTRcv function returns with the `T_MORE` flag not set. If the endpoint does not support the concept of a TSDU (Transport Service Data Unit), the `T_MORE` flag is not meaningful and should be ignored. To determine whether the endpoint supports TSDUs, examine the `tsdu` field of the `TEndpointInfo` (page 174) structure. A value of `T_INVALID` means that the endpoint does not support it.

Some protocols allow you to send zero-length data to signal the end of a logical unit. In this case, if you request more than 0 bytes when calling the OTRcv function, the function returns 0 bytes only to signal the end of a TSDU.

If the OTRcv function returns and the T_EXPEDITED bit is set in the flags parameter, this means that you are about to read expedited data. If the number of bytes of expedited data exceeds the number of bytes you specified in the reqCount parameter, both the T_EXPEDITED and the T_MORE bits are set. You must call the OTRcv function until the T_MORE flag is not set to retrieve the rest of the expedited data.

If you are calling the OTRcv function repeatedly to read normal data and a call to the function returns T_EXPEDITED in the flags parameter, the next call to the OTRcv function that returns without the T_EXPEDITED flag set returns normal data at the place it was interrupted. It is your responsibility to remember where that was and to continue processing normal data. You can determine how much normal data you read by maintaining a running total of the number of bytes returned in the OTStatus result.

If the endpoint is in asynchronous mode or is not blocking, the function returns with the kOTNoDataErr result if no data is available. If you have installed a notifier, the endpoint provider calls your notifier and passes T_DATA or T_EXDATA for the code parameter when there is data available. If you have not installed a notifier, you may poll for these events using the OTLook function. Once you receive a T_DATA or T_EXDATA event, you should continue in a loop, calling the OTRcv function until it returns with the kOTNoDataErr result.

If the endpoint is in synchronous mode and is blocking, the endpoint waits for data if none is currently available. You should avoid calling the OTRcv function this way because it might cause processing to hang if no data is available. If you are doing other operations in synchronous mode, you should put the endpoint in nonblocking mode before calling the OTRcv function.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTRcvConnect

Reads the status of an outstanding or completed asynchronous call to the`OTConnect` function. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTRcvConnect (
   EndpointRef ref,
   TCall *call
);
```

Parameters

ref

call

Return Value

A result code. See "Open Transport Result Codes" (page 354).

Discussion

You call the OTRcvConnect function to determine the status of a previously issued OTConnect call. If you want to retrieve information about the connection, you must allocate buffers for the addr field and, if required, the opt and udata fields before you make the call.

If the endpoint is synchronous and blocking, the OTRcvConnect function waits for the connection to be accepted or rejected. If the connection is accepted, the function returns with a kOTNoError result. If the connection is rejected, the function returns with a kOTLookErr result. In this case, you should call the OTLook function to verify that a T_DISCONNECT event is the reason for the kOTLookErr, and then you should call the OTRcvDisconnect function to clear the event.

If the endpoint is asynchronous or nonblocking, the OTRcvConnect function returns with the kOTNoDataErr result if the connection has not yet been established.

Availability

Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`OpenTransport.h`

OTRcvDisconnect

Identifies the cause of a connection break or of a connection rejection, acknowledges and clears the corresponding disconnection event. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTRcvDisconnect (
   EndpointRef ref,
   TDiscon *discon
);
```

Parameters

ref

discon

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
Calling the OTRcvDisconnect function clears the corresponding disconnection event and retrieves any user data sent with the disconnection.

If you do not care about data returned with the disconnection and do not needto know the reason for the disconnection nor the sequence ID, you may specify a nil pointer for the discon parameter. In this case, the provider discards any user data associated with the disconnection.

The OTRcvDisconnect function behaves in the same way for all modes of operation. If there is no disconnection request pending, the function returns with the kOTNoDisconnectErr result. If there is a disconnection request pending, the function returns either the kOTNoError or kOTBufferOverflowErr result. In the latter instance, you need to check the discon field of the `TEndpointInfo` (page 174) structure for your endpoint and make sure that the buffer referenced by the udata.buf field is at least as big as the value specified for the discon field.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTRcvOrderlyDisconnect

Acknowledges a request for an orderly disconnect. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTRcvOrderlyDisconnect (
   EndpointRef ref
);
```

Parameters

ref

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
The OTRcvOrderlyDisconnect function is a service that is not supported by all protocols. If it is, the servtype field of the `TEndpointInfo` (page 174) structure has the value `T_COTS_ORD` or `T_TRANS_ORD` for the endpoint.

After using the OTRcvOrderlyDisconnect function to acknowledge receipt of a disconnection request, there will not be any more data to receive. Calls to the OTRcv function (for a transactionless connection) or to the OTRcvRequest function (for a transaction-based connection) after acknowledging a disconnection request fail with the kOTOutStateErr result. If the endpoint supports a remote orderly disconnect, you can still send data over the connection if you have not yet called the OTSndOrderlyDisconnect function.

The OTRcvOrderlyDisconnect function behaves in the same way in all modes of operation. If there is no disconnection request pending, the function returns with the kOTNoReleaseErr result. It there is a disconnection request pending, the function returns either the kOTNoError or kOTBufferOverflowErr result. In the latter instance, you need to check the discon field of the`TEndpointInfo` (page 174) structure for your endpoint and make sure that the buffer referenced by the udata.buf field is at least as big as the value specified for the discon field.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTRcvUData

Reads data sent by a client using a connectionless transactionless protocol. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTRcvUData (
   EndpointRef ref,
   TUnitData *udata,
   OTFlags *flags
);
```

Parameters
ref
udata
flags

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
When the OTRcvUData function returns, it passes a pointer to a TUnitData structure containing information about the data read and a pointer to a flags variable that is set to indicate whether there is more data to be retrieved. If the buffer pointed to by the udata->udata.buf field is not large enough to hold the current data unit, the endpoint provider fills the buffer and sets the flags parameter to T_MORE to indicate that you must call the OTRcvUData function again to receive additional data. Subsequent calls to the OTRcvUData function return 0 for the length of the address and option buffers until you receive the full data unit. The last unit to be received does not have the T_MORE flag set.

If the endpoint is in asynchronous mode or is not blocking and data is not available, the OTRcvUData function fails with the kOTNoDataErr result. The endpoint provider uses the T_DATA event to notify the endpoint when data becomes available. You can use a notifier function or the OTLook function to retrieve the event. Once you get the T_DATA event, you should continue calling the OTRcvUData function until it returns the kOTNoDataErr result.

It is possible that the provider generates an erroneous T_DATA event. This is the case when the provider calls your notifier, passing T_DATA for the code parameter; but when you execute the OTRcvUData function, it returns with a kOTNoDataErr result. If this happens, you should continue normal processing and assume that the next T_DATA event is genuine.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTRcvUDErr

Clears an error condition indicated by a `T_UDERR` event and returns the reason for the error. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTRcvUDErr (
   EndpointRef ref,
   TUDErr *uderr
);
```

Parameters
ref
uderr

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
You use the OTRcvUDErr function if you have called the OTSndUData function and the endpoint provider has issued the T_UDERR event to indicate that the send operation did not succeed. This usually happens when the endpoint provider cannot determine immediately that you have specified a bad address or option value. For example, assume that you are using AppleTalk and you specify an NBP address. If Open Transport cannot resolve the address, it sends a T_UDERR event to your notifier function. To clear the error condition and determine the cause of the failure, you must call the OTRcvUDErr function.

If the size of the option or error data returned exceeds the size of the allocated buffers, the OTRcvUDErr function returns with the result kOTBufferOverflowErr, but the error indication is cleared anyway.

If you do not need to identify the cause of the failure, you can set the uderr pointer to nil. In this case, the OTRcvUDErr function clears the error indication without reporting any information to you. It is important, nevertheless, that you actually call the OTRcvUDErr function to clear the error condition. If you don't call this function, the endpoint remains in an invalid state for doing other send operations, and the endpoint provider is unable to deallocate memory reserved for internal buffers associated with the send operation.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
OpenTransport.h

OTReadBuffer

Copies data out of a no-copy receive buffer. (Deprecated in Mac OS X v10.4.)

```
Boolean OTReadBuffer (
   OTBufferInfo *buffer,
   void *dest,
   OTByteCount *len
);
```

Parameters

buffer

dest

len

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
OpenTransportProtocol.h

OTRegisterAsClientInContext

Registers your application as a client of Open Transport and gives Open Transport a notifier function it can use to send you events. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTRegisterAsClientInContext (
   OTClientName name,
   OTNotifyUPP proc,
   OTClientContextPtr clientContext
);
```

Parameters

name

proc

clientContext

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Availability
Available in CarbonLib 1.3 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTRegisterName

Registers an entity name on the network. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTRegisterName (
   MapperRef ref,
   TRegisterRequest *req,
   TRegisterReply *reply
);
```

Parameters

ref

req

reply

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
If the name-registration protocol defined using the config parameter to the OTOpenMapper or OTAsyncOpenMapper function supports dynamic name and address registration, you can use the OTRegisterName function to make a name visible on the network to other network devices.

Some protocol implementations under Open Transport allow a client to specify a name rather than an address when binding the endpoint using the OTBind function. Binding an endpoint by name causes the protocol to automatically register the name on the network if the protocol supports dynamic name registration. This is the simpler technique for registering a name and is preferred over creating a mapper provider and then using the OTRegisterName function to register the name.

The format for the requested name and address is specific to the protocol used. Please consult the documentation for the protocol you are using for format information.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTReleaseBuffer

Returns the no-copy receive buffer to the system. (Deprecated in Mac OS X v10.4.)

```
void OTReleaseBuffer (
   OTBuffer *buffer
);
```

Parameters

buffer

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
OpenTransportProtocol.h

OTRemoveFirst

Removes the first link in a FIFO list. (Deprecated in Mac OS X v10.4.)

```
OTLink * OTRemoveFirst (
   OTList *list
);
```

Parameters

list

Return Value
See the description of the `OTLink` data type.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
OpenTransport.h

OTRemoveLast

Removes the last link in a FIFO list. (Deprecated in Mac OS X v10.4.)

```
OTLink * OTRemoveLast (
   OTList *list
);
```

Parameters

list

Return Value
See the description of the `OTLink` data type.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTRemoveLink

Removes the last link in a FIFO list. (Deprecated in Mac OS X v10.4.)

```
Boolean OTRemoveLink (
   OTList *list,
   OTLink *link
);
```

Parameters
list
link

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTRemoveNotifier

Removes a provider's notifier function. (Deprecated in Mac OS X v10.4.)

```
void OTRemoveNotifier (
   ProviderRef ref
);
```

Parameters
ref

Discussion
The `OTRemoveNotifier` function removes the notifier (if any) currently installed for the provider specified by the `ref` parameter.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
OpenTransport.h

OTResolveAddress

Returns the protocol address that corresponds to the name of an endpoint. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTResolveAddress (
   EndpointRef ref,
   TBind *reqAddr,
   TBind *retAddr,
   OTTimeout timeOut
);
```

Parameters

ref

reqAddr

retAddr

timeOut

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
The OTResolveAddress function returns the lowest-level address for your endpoint. Not all endpoints support this function. A value of CAN_RESOLVE_ADDR in the flags field of the `TEndpointInfo` (page 174) structure indicates that the endpoint does support this function. Using this function saves you the trouble of opening and closing a mapper provider if the only reason you have for opening the mapper is to look up the address corresponding to a specific endpoint name. You would still have to open the mapper if you needed to look up a name pattern—that is, if the name included any wildcard characters.

You are responsible for initializing the buffers described by the req and ret parameters required to hold the addresses. To determine how large these buffers should be, examine the addr field of the `TEndpointInfo` structure, which specifies the maximum amount of memory needed to store an address for the endpoint specified by the ref parameter.

If a notifier is not installed, it is not possible to determine when the OTResolveAddress function completes.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
OpenTransport.h

OTReverseList

Reverses the order in which entries are linked in a list. (Deprecated in Mac OS X v10.4.)

```
OTLink * OTReverseList (
   OTLink *list
);
```

Parameters

list

Return Value
See the description of the `OTLink` data type.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTScheduleDeferredTask

Schedules a task for execution at deferred task time. (Deprecated in Mac OS X v10.4.)

```
Boolean OTScheduleDeferredTask (
   OTDeferredTaskRef dtCookie
);
```

Parameters

dtCookie

Discussion
The `OTScheduleDeferredTask` function schedules for execution at the next deferred task time the task associated with the `dtCookie` parameter, which is the reference returned by the `OTCreateDeferredTask` function.

You can call this function at any time. If you have not yet destroyed a task, you can use this function to reschedule the same task more than once.

If you makes multiple calls to the `OTScheduleDeferredTask` function before the task is executed, additional tasks are not scheduled; only one instance of each unique task can only be scheduled at a time.

This function returns `true` if it scheduled the deferred task successfully, `false` if not. If it returns `false` and the `dtCookie` parameter has a valid value (other than 0), then the task is already scheduled to run. If `dtCookie` is invalid (a value of 0), the function returns `false` and does nothing.

If you want to call Open Transport from an interrupt, you can use this function (and the `OTCreateDeferredTask` function) instead of the standard Deferred Task Manager function `DTInstall` to create a deferred task that permits you to call Open Transport function calls. This allows Open Transport to adapt to changes in the underlying operating system without affecting the client's code.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTScheduleTimerTask

Schedules a timer task to be executed at the specified time. (Deprecated in Mac OS X v10.4.)

```
Boolean OTScheduleTimerTask (
   OTTimerTask timerTask,
   OTTimeout milliSeconds
);
```

Parameters

timerTask

milliSeconds

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransportProtocol.h`

OTSetAddressFromNBPEntity

Stores an NBP entity structure as an NBP address string. (Deprecated in Mac OS X v10.4.)

```
OTByteCount OTSetAddressFromNBPEntity (
   UInt8 *nameBuf,
   const NBPEntity *entity
);
```

Parameters

nameBuf

A pointer to the NBP address buffer in which you wish to store the NBP entity.

entity

Return Value
See the description of the `OTByteCount` data type.

Discussion
The `OTSetAddressFromNBPEntity` function stores the information in the NBP entity into the buffer specified by the `nameBuf` parameter in the format required for mapper calls—that is, if you have a backslash (\), a colon (:), or an at-sign (@) in your NBP name, this function inserts a backslash before each so that the mapper functions can handle them correctly. This function returns the number of bytes that were actually used in the buffer.

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In
`OpenTransportProviders.h`

OTSetAddressFromNBPString

Copies an NBP name string into an NBP address buffer. (Deprecated in Mac OS X v10.4.)

```
OTByteCount OTSetAddressFromNBPString (
   UInt8 *addrBuf,
   const char *name,
   SInt32 len
);
```

Parameters

addrBuf

A pointer to the NBP address buffer in which to store the NBP name string.

name

A pointer to the NBP name string you wish to copy into the buffer.

len

The number of characters to copy.

Return Value
See the description of the `OTByteCount` data type.

Discussion
The `OTSetAddressFromNBPString` function copies the string indicated by the `nbpName` parameter into the buffer indicated by the `addrBuf` parameter. The `len` parameter indicates the number of characters to copy. A value of -1 copies the entire `nbpName` string. The function returns the number of bytes actually copied.

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransportProviders.h`

OTSetAsynchronous

Sets a provider's mode of execution to asynchronous. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTSetAsynchronous (
   ProviderRef ref
);
```

Parameters

ref

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
The `OTSetAsynchronous` function causes all functions for the provider specified in the `ref` parameter to run asynchronously. You must install a notifier function for the provider if it needs to receive completion events. You can install a notifier function either before or after calling the `OTSetAsynchronous` function.

Changing a provider's mode of execution does not affect its notifier function, if any; the notifier function remains installed.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTSetBit

Sets a bit atomically. (Deprecated in Mac OS X v10.4.)

```
Boolean OTSetBit (
   UInt8 *bitMap,
   OTByteCount bitNo
);
```

Parameters

bitMap

bitNo

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransportProtocol.h`

OTSetBlocking

Allows a provider to wait or block until it is able to send or receive data. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTSetBlocking (
   ProviderRef ref
);
```

Parameters

ref

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
The `OTSetBlocking` function causes provider functions that send or receive data to wait if current conditions prevent them from completing an operation. By default, a provider is in nonblocking mode, in which case, if a provider function were unable to complete sending or receiving data, it would return immediately with a result that would tell you why the operation was unable to complete.

If a provider is in blocking mode and you call the `OTCloseProvider` function to close the provider, Open Transport gives each Streams module up to 15 seconds to process outgoing commands. It is recommended that you call the `OTSetNonBlocking` function before you call the `OTCloseProvider` function.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTSetBusTypeInPortRef

Sets bus type for a port reference. (Deprecated in Mac OS X v10.4.)

```
OTPortRef OTSetBusTypeInPortRef (
   OTPortRef ref,
   OTBusType busType
);
```

Parameters

ref

busType

Return Value
See the description of the `OTPortRef` data type.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Carbon Porting Notes

OT ports are read only in Carbon. In Mac OS X, code that communicates directly with network interfaces must use the IOKit API.

Declared In
OpenTransport.h

OTSetDeviceTypeInPortRef

Sets device type for a port reference. (Deprecated in Mac OS X v10.4.)

```
OTPortRef OTSetDeviceTypeInPortRef (
   OTPortRef ref,
   OTDeviceType devType
);
```

Parameters

ref

devType

Return Value
See the description of the OTPortRef data type.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Carbon Porting Notes

OT ports are read only in Carbon. In Mac OS X, code that communicates directly with network interfaces must use the IOKit API.

Declared In
OpenTransport.h

OTSetFirstClearBit

Atomcially sets the first clear bit in a specified bit map. (Deprecated in Mac OS X v10.4.)

```
OTResult OTSetFirstClearBit (
   UInt8 *bitMap,
   OTByteCount startBit,
   OTByteCount numBits
);
```

Parameters

bitMap

startBit

numBits

Return Value
See the description of the OTResult data type.

Discussion
Sets the first clear bit in bitMap , starting with startBit and giving up after numBits. Returns the bit number that was set, or a kOTNotFoundErr error if there was no clear bit available

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransportProtocol.h`

OTSetNBPEntityFromAddress

Parses and stores an NBP address into an NBP entity. (Deprecated in Mac OS X v10.4.)

```
Boolean OTSetNBPEntityFromAddress (
   NBPEntity *entity,
   const UInt8 *addrBuf,
   OTByteCount len
);
```

Parameters

entity

addrBuf

A pointer to the address buffer in which to store the NBP name string.

len

Discussion
The `OTSetNBPEntityFromAddress` function parses an NBP address or a combined DDP-NBP address into the NBP name's constituent parts (name, type, and zone) and stores the result in an NBP entity. The function ignores the DDP address part of a combined DDP-NBP address. From the NBP entity, each of the constituent parts of the name can be later retrieved or changed.

This function returns `true` if it worked successfully; it returns `false` if it had to truncate any data—that is, if the address had data that was too long in one of the fields, each of which only holds 32 characters of data. When this occurs, Open Transport still stores the data, but in a truncated form.

Availability
Available in CarbonLib 1.0 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransportProviders.h`

OTSetNBPName

Stores the name part of an NBP name into an NBP entity structure. (Deprecated in Mac OS X v10.4.)

```
Boolean OTSetNBPName (
   NBPEntity *entity,
   const char *name
);
```

Parameters

entity

name

A pointer to the name portion of an NBP name string that you wish to store.

Discussion

The `OTSetNBPName` function stores the NBP name specified by the `name` parameter into the NBP entity structure indicated by the `nbpEntity` parameter, deleting any previous name stored there. This function returns `false` if the `name` parameter is longer than the maximum allowed for a name part of an NBP name (32 characters).

Availability

Available in CarbonLib 1.0 and later.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`OpenTransportProviders.h`

OTSetNBPType

Stores the type part of an NBP name in an NBP entity structure. (Deprecated in Mac OS X v10.4.)

```
Boolean OTSetNBPType (
   NBPEntity *entity,
   const char *typeVal
);
```

Parameters

entity

typeVal

A pointer to the type portion of an NBP name string that you wish to store.

Discussion

The `OTSetNBPType` function stores the NBP type specified by the `type` parameter into the NBP entity structure indicated by the `nbpEntity` parameter, deleting any previous type stored there. The type supplied must not have any escape characters stored in it, although you do not receive any error message if you do use such characters. This function returns `false` if the `type` parameter is longer than the maximum allowed for type part of an NBP name (32 characters).

Availability

Available in CarbonLib 1.0 and later.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`OpenTransportProviders.h`

OTSetNBPZone

Stores the zone part of an NBP name in an NBP entity structure. (Deprecated in Mac OS X v10.4.)

```
Boolean OTSetNBPZone (
   NBPEntity *entity,
   const char *zone
);
```

Parameters

entity

zone

A pointer to the zone portion of an NBP name string that you wish to store.

Discussion

The `OTSetNBPZone` function stores the NBP zone specified by the `zone` parameter into the NBP entity structure indicated by the `nbpEntity` parameter, deleting any previous zone stored there. The zone supplied must not have any of the NBP escape characters stored in it, although you do not receive any error message if you do use such characters. This function returns `false` if the zone parameter is longer than the maximum allowed for zone part of an NBP name (32 characters).

Availability

Available in CarbonLib 1.0 and later.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`OpenTransportProviders.h`

OTSetNonBlocking

Disallows a provider from waiting if it cannot currently complete a function that sends or receives data. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTSetNonBlocking (
   ProviderRef ref
);
```

Parameters

ref

Return Value

A result code. See "Open Transport Result Codes" (page 354).

Discussion

The `OTSetNonBlocking` function causes provider functions to return a result code immediately, instead of waiting for a function that sends or receives data to complete. When you open a provider, its mode of operation is set to nonblocking by default.

Availability

Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTSetSynchronous

Sets a provider's mode of execution to synchronous. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTSetSynchronous (
   ProviderRef ref
);
```

Parameters

ref

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
The `OTSetSynchronous` function causes all provider functions to run synchronously when using the provider that you specify.

Changing a provider's mode of execution does not affect its notifier function, if any is installed for this provider; the notifier function remains installed.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTSnd

Sends data to a remote peer. (Deprecated in Mac OS X v10.4.)

```
OTResult OTSnd (
   EndpointRef ref,
   void *buf,
   OTByteCount nbytes,
   OTFlags flags
);
```

Parameters

ref

buf

A pointer to the data being sent. If you are sending data that is not stored contiguously, this is a pointer to an OTData structure that describes the first data fragment.

nbytes
flags

Return Value
See the description of the `OTResult` data type.

Discussion
You use the OTSnd function to send data to a remote peer. Before you use this function, you must establish a connection with the peer.

If the OTSnd function succeeds, it returns an integer (OSStatus) specifying the number of bytes that were actually sent. If it fails, it returns a negative integer corresponding to a result code that indicates the reason for the failure.

You specify the data to be sent by passing a pointer to the data (buf) and byspecifying the size of the data (nbytes). The maximum size of the data you can send is specified by the `tsdu` field of the `TEndpointInfo` (page 174) structure for the endpoint.

Some protocols use expedited data for control or attention messages. To determine whether the endpoint supports this service, examine the `etsdu` field of the `TEndpointInfo` structure. A positive integer for the `etsdu` field indicates the maximum size in bytes of expedited data that you can send. To send expedited data, you must set the T_EXPEDITED bit of the flags parameter.

If you want to break up the data sent into smaller logical units, you can set the T_MORE bit of the flags parameter to indicate that you are using additional calls to the OTSnd function to send more data that belongs to the same logical unit. To indicate that the last data unit is being sent, you must specify 0 for nbytes and turn off the T_MORE flag. This is the only circumstance under which it is permitted to send a zero-length data unit. If the endpoint does not support the sending of zero-length data, the OTSnd function fails with the kOTBadDataErr result.

If the endpoint is in blocking mode, the OTSnd function returns after it actually sends the data. If flow-control restrictions prevent its sending the data, it retries the operation until it is able to send it. If the endpoint is in nonblocking mode, the OTSnd function returns with the kOTFlowErr result if flow-control restrictions prevent the data from being sent. When the endpoint provider is able to send the data, it returns a T_GODATA event to let you know that it is possible to send data.

The following table shows how the endpoint's mode of execution and blocking status affects the behavior of the OTSnd function.

Table A-1

	Blocking	Nonblocking
Synchronous	The function returns when the provider lifts flow-control restrictions The kOTFlowErr result is never returned.	The function returns immediately. The kOTFlowErr result might be returned.
Asynchronous	The function returns immediately. The kOTFlowErr result is never returned.	The function returns immediately. The kOTFlowErr result might be returned.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTSndDisconnect

Tears down an open connection (abortive disconnect) or rejects an incoming connection request. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTSndDisconnect (
   EndpointRef ref,
   TCall *call
);
```

Parameters

ref

call

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
There are two functions that you can use to tear down a connection: OTSndDisconnect for an abortive disconnect, or OTSndOrderlyDisconnect for an orderly disconnect. It is recommended that you use the OTSndOrderlyDisconnect function for tearing down a connection whenever possible and that you use the OTSndDisconnect function only for rejecting incoming connection requests.

If the endpoint is in asynchronous mode, the OTSndDisconnect function returns immediately with a result of kOTNoError to indicate that the disconnection process has begun and that your notifier function will be called when the process completes.

When the connection has been broken, the provider issues a T_DISCONNECTCOMPLETE event. If you have installed a notifier function, Open Transport calls your notifier and passes this event in the code parameter. The cookie parameter contains the call parameter. If you have not installed a notifier function, you cannot determine when this function completes.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTSndOrderlyDisconnect

Initiates or completes an orderly disconnection. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTSndOrderlyDisconnect (
   EndpointRef ref
);
```

Parameters

ref

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
You call the `OTSndOrderlyDisconnect` function to initiate an orderly release of a connection and to indicate to the peer endpoint that you have no more data to send. After calling this function, you must not send any more data over the connection. However, you can still continue to receive data if the peer endpoint has not yet called the `OTSndOrderlyDisconnect` function.

This function is a service that is not supported by all protocols. If it is supported, the servtype field of the `TEndpointInfo` (page 174) structure has the value `T_COTS_ORD` or `T_TRANS_ORD`.

The `OTSndOrderlyDisconnect` function behaves exactly the same in all modes of operation.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTSndUData

Sends data using a connectionless transactionless endpoint. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTSndUData (
   EndpointRef ref,
   TUnitData *udata
);
```

Parameters

ref

udata

A pointer to a TUnitData structure that specifies thedata to be sent and its destination.

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
If the endpoint is in synchronous, blocking mode, the OTSndUData function returns immediately. If flow-control restrictions prevent its sending the data, it retries the operation until it is able to send it. If the endpoint is in nonblocking mode, the OTSndUData function returns a kOTFlowErr result if flow-control restrictions prevent the data from being sent. When the endpoint provider is able to send the data, it calls your notifier function, passing T_GODATA for the code parameter. You can then call the OTSndUData function from your notifier to send the data. You can also retrieve this event by polling the endpoint using the OTLook function.

Some endpoint providers are not able to detect immediately whether you specified incorrect address or option information. In such cases, the provider calls your notifier function when it detects the error, passing the T_UDERR for the code parameter to advise you that an error has occurred. You can determine the cause of this event by calling the OTRcvUDErr function and examining the value of the uderr->error parameter. It is important that you call the OTRcvUDErr function even if you are not interested in examining the cause of the error. Failing to do this leaves the endpoint in an invalid state for doing other sends and makes the endpoint provider unable to deallocate memory reserved for internal buffers associated with the send.

The next table shows how the endpoint's mode of execution and blocking status affects the behavior of the OTSndUData function.

Table A-2

	Blocking	Nonblocking
Synchronous	The function returns when the provider lifts flow-control restrictions The kOTFlowErr result is never returned.	The function returns immediately. the kOTFlowErr result might be returned
Asynchronous	The function returns immediately. The kOTFlowErr result is never returned.	the function returns immediately. the kOTFlowErr result might be returned.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTStrCat

Concatenates two C strings. (Deprecated in Mac OS X v10.4.)

```
void OTStrCat (
   char *dest,
   const char *src
);
```

Parameters
dest
src

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTStrCopy

Copies a C string. (Deprecated in Mac OS X v10.4.)

```
void OTStrCopy (
   char *dest,
   const char *src
);
```

Parameters

dest

src

Availability

Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`OpenTransport.h`

OTStrEqual

Determines whether two C strings are the same. (Deprecated in Mac OS X v10.4.)

```
Boolean OTStrEqual (
   const char *src1,
   const char *src2
);
```

Parameters

src1

src2

Availability

Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`OpenTransport.h`

OTStrLength

Returns the length of a C string. (Deprecated in Mac OS X v10.4.)

```
OTByteCount OTStrLength (
   const char *str
);
```

Parameters

str

Return Value
See the description of the `OTByteCount` data type.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTSubtractTimeStamps

Subtracts one timestamp value from another. (Deprecated in Mac OS X v10.4.)

```
OTTimeStamp * OTSubtractTimeStamps (
   OTTimeStamp *result,
   OTTimeStamp *startTime,
   OTTimeStamp *endEnd
);
```

Parameters

result
startTime
endEnd

Return Value
See the description of the `OTTimeStamp` data type.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTTestBit

Atomically tests a bit in a specified bit map. (Deprecated in Mac OS X v10.4.)

```
Boolean OTTestBit (
   UInt8 *bitMap,
   OTByteCount bitNo
);
```

Parameters

bitMap

bitNo

Availability

Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`OpenTransportProtocol.h`

OTTimeStampInMicroseconds

Calculates the time elapsed in microseconds since since a specified time. (Deprecated in Mac OS X v10.4.)

```
UInt32 OTTimeStampInMicroseconds (
   OTTimeStamp *delta
);
```

Parameters

delta

Availability

Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`OpenTransport.h`

OTTimeStampInMilliseconds

Calculates the time elapsed in milliseconds since since a specified time. (Deprecated in Mac OS X v10.4.)

```
UInt32 OTTimeStampInMilliseconds (
   OTTimeStamp *delta
);
```

Parameters

delta

Availability

Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTUnbind

Dissociates an endpoint from its address or cancels an asynchronous call to the `OTBind` function. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTUnbind (
   EndpointRef ref
);
```

Parameters

ref

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Discussion
If you call the `OTUnbind` function asynchronously and you have not installed a notifier function, the only way to determine that the endpoint has been unbound is to use the OTGetEndpointState function to poll the state of the endpoint. The function returns the kOTStateChangeErr result when the OTUnbind function returns. If the function succeeds, the state of the endpoint is T_UNBND. If it fails, its state is T_IDLE.

After you unbind an endpoint, you can no longer use it to send or receive information. You can use the OTCloseProvider function to deallocate memory reserved for the endpoint, or you can use the OTBind function to associate it with another address and then resume transferring data or establishing a connection.

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTUnregisterAsClientInContext

Removes your application as a client of Open Transport. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTUnregisterAsClientInContext (
   OTClientContextPtr clientContext
);
```

Parameters

clientContext

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Availability
Available in CarbonLib 1.3 and later.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

OTUseSyncIdleEvents

Allows synchronous idle events to be sent to your notifier. (Deprecated in Mac OS X v10.4.)

```
OSStatus OTUseSyncIdleEvents (
   ProviderRef ref,
   Boolean useEvents
);
```

Parameters

ref

useEvents

Return Value
A result code. See "Open Transport Result Codes" (page 354).

Availability
Available in CarbonLib 1.0 and later when OpenTransport 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`OpenTransport.h`

Unsupported Functions

This section lists functions that are unsupported in Mac OS X. Table B-1 provides information on what you should do in place of using these functions.

Table B-1 Porting notes for unsupported Open Transport functions

Unsupported functions	Porting notes
CloseOpenTransport	
InitOpenTransport	Use InitOpenTransportInContext instead.
InitOpenTransportUtilities	Use InitOpenTransportInContext instead.
OTAddToHashList	Carbon does not support Open Transport hash lists because Apple has not identified a need for them.
OTAlloc	Use OTAllocInContext instead.
OTAllocMem	Use OTAllocMemInContext instead.
OTAllocMsg	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
OTAllocPortMem	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
OTAllocSharedClientMem	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
OTAsyncCreateStream	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
OTAsyncOpenAppleTalkServices	Use OTAsyncOpenAppleTalkServicesInContext instead.
OTAsyncOpenEndpoint	Use OTAsyncOpenEndpointInContext instead.
OTAsyncOpenInternetServices	Use OTAsyncOpenInternetServicesInContext instead.
OTAsyncOpenMapper	Use OTAsyncOpenMapperInContext instead.
OTAsyncOpenProvider	Use the open routine corresponding to the type of provider instead: OTAsyncOpenEndpointInContext, OTAsyncOpenMapperInContext, OTAsyncOpenInternetServicesInContext, OTAsyncOpenAppleTalkServicesInContext

Unsupported functions	Porting notes
OTAsyncStreamOpen	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
OTAsyncStreamPoll	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
OTCalculateHashListMemoryNeeds	Carbon does not support Open Transport hash lists because Apple has not identified a need for them.
OTCanLoadLibraries	This function will not be supported because Apple has not identified a specific developer need for it.
OTCancelReply	Open Transport\qs connection-oriented transaction-based endpoint feature will not be supported.
OTCancelRequest	Open Transport\qs connection-oriented transaction-based endpoint feature will not be supported.
OTCancelSystemTask	This function will not be supported because Apple has not identified a specific developer need for it.
OTCancelUReply	Carbon does not support transaction oriented endpoints.
OTCancelURequest	Carbon does not support transaction-oriented endpoints.
OTCfigAddChild	Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTCfigChangeProviderName	Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTCfigCloneConfiguration	Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTCfigDeleteConfiguration	Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTCfigGetChild	Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTCfigGetInstallFlags	Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTCfigGetOptionNetbuf	Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTCfigGetParent	Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTCfigGetPortRef	Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.

Unsupported functions	Porting notes
OTCfigGetProviderName	Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTCfigIsPort	Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTCfigNewChild	Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTCfigNewConfiguration	Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTCfigNumberOfChildren	Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTCfigPopChild	Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTCfigPushChild	Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTCfigPushNewSingleChild	Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTCfigPushParent	Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTCfigRemoveChild	Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTCfigSetPath	Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTCfigSetPortRef	Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTChangePortState	OT ports are read only in Carbon. In Mac OS X, code that communicates directly with network interfaces must use the IOKit API.
OTCloseMatchingProviders	Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTCloseProviderByStream	Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTCloseProvidersByPortRef	Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTCloseProvidersByUseCount	Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.

Unsupported functions	Porting notes
OTConfiguratorUnloaded	Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTConfigureChildren	Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTCreateControlStream	Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTCreateDeferredTask	Use OTCreateDeferredTaskInContext.
OTCreateOptionString	Apple has not identified a developer need for this function. You can use TOption structures if necessary.
OTCreateOptions	Apple has not identified a developer need for this function. You can use TOption structures if necessary.
OTCreateStateMachine	Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTCreateStream	Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTCreateSystemTask	This function will not be supported because Apple has not identified a specific developer need for it.
OTCreateTimerTask	Use OTCreateTimerTaskInContext.
OTDeleteConfigurator	Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTDestroyStateMachine	Carbon does not support access to Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTDestroySystemTask	This function will not be supported because Apple has not identified a specific developer need for it.
OTEnterGate	This function will not be supported because Apple has not identified a specific developer need for it.
OTEnterInterrupt	Carbon applications cannot be called from a driver
OTErrorToOSStatus	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
OTFindCFMLibraries	Carbon does not support direct manipulation of CFM libraries via the Open Transport API.
OTFindInHashList	Carbon does not support Open Transport hash lists because Apple has not identified a need for them.

Unsupported functions	Porting notes
OTFindPortByDev	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
OTFindPortConflict	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
OTFreePortMem	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
OTFreeSharedClientMem	Carbon does not support Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTGetCFMPointer	Carbon does not support Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTGetCFMSymbol	Carbon does not support Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTGetConfiguratorUserData	Carbon does not support Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTGetMessage	This function will not be supported because Apple has not identified a specific developer need for it.
OTGetPortIconFromPortRef	Carbon does not support access to the Open Transport port name or icon because this information is not available on Mac OS X.
OTGetPriorityMessage	This function will not be supported because Apple has not identified a specific developer need for it.
OTGetProviderPortRef	Due to architectural changes Carbon will not support this function.
OTGetRandomNumber	
OTGetRandomSeed	
OTGetUserPortNameFromPortRef	Mac OS X does not currently support this type of information.
OTHoldThisCFMLibrary	Carbon does not support direct manipulation of CFM or ASLM libraries via the Open Transport API.
OTInetGetDHCPConfigInfo	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
OTInitGate	This function will not be supported because Apple has not identified a specific developer need for it.
OTInitHashList	Carbon does not support Open Transport hash lists because Apple has not identified a developer need for them.

Unsupported functions	Porting notes
OTIsAtInterruptLevel	Carbon cannot be called at interrupt level, so this function would always return the same value.
OTIsDependentPort	Carbon does not support Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTIsInHashList	Carbon does not support Open Transport hash lists because Apple has not identified a developer need for them.
OTIsMasterConfigurator	Carbon does not support Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTIsPortCompatibleWith	Carbon does not support Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTKernelPrintf	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
OTLeaveGate	This function will not be supported because Apple has not identified a specific developer need for it.
OTLeaveInterrupt	Carbon applications cannot be called from a driver
OTLoadASLMLibrary	Carbon does not support direct manipulation of CFM or ASLM libraries via the Open Transport API.
OTLoadCFMLibrary	Carbon does not support direct manipulation of CFM or ASLM libraries via the Open Transport API.
OTNewConfigurator	Carbon does not support Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTNewControlMask	Carbon does not support Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTNotifyAllClients	Carbon does not support Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTNotifyUser	Carbon does not support Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTOpenAppleTalkServices	Use OTOpenAppleTalkServicesInContext instead.
OTOpenEndpoint	Use OTOpenEndpointInContext.
OTOpenEndpointOnStream	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
OTOpenInternetServices	Use OTOpenInternetServicesInContext instead.
OTOpenMapper	Use OTOpenMapperInContext instead.

Unsupported functions	Porting notes
OTOpenProvider	Due to architectural changes, Carbon will not support this function.
OTOpenProviderOnStream	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
OTPeekMessage	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
OTPutBackBuffer	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
OTPutBackPartialBuffer	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
OTPutMessage	This function will not be supported because Apple has not identified a specific developer need for it.
OTPutPriorityMessage	This function will not be supported because Apple has not identified a specific developer need for it.
OTRcvReply	Open Transport\qs connection-oriented transaction-based endpoint feature will not be supported.
OTRcvRequest	Open Transport\qs connection-oriented transaction-based endpoint feature will not be supported.
OTRcvUReply	Carbon does not support transaction-oriented endpoints.
OTRcvURequest	Carbon does not support transaction-oriented endpoints.
OTReadMessage	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
OTReallocMem	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
OTRegisterAsClient	Apple has not identified an application need for this function.
OTRegisterPort	Due to architectural changes, Carbon will not support this function.
OTReleaseCFMConnection	Carbon does not support direct manipulation of CFM or ASLM libraries via the Open Transport API.
OTRemoveFromHashList	Carbon does not support Open Transport hash lists because Apple has not identified a developer need for them.
OTRemoveLinkFromHashList	Carbon does not support Open Transport hash lists because Apple has not identified a developer need for them.
OTRemoveStreamFromProvider	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.

Unsupported functions	Porting notes
OTSMCallStateProc	Carbon does not support Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTSMComplete	Carbon does not support Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTSMCreateControlStream	Carbon does not support Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTSMCreateStream	Carbon does not support Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTSMGetClientData	Carbon does not support Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTSMGetMessage	Carbon does not support Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTSMGetState	Carbon does not support Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTSMInstallCompletionProc	Carbon does not support Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTSMIoctl	Carbon does not support Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTSMOpenStream	Carbon does not support Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTSMPopCallback	Carbon does not support Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTSMPutMessage	Carbon does not support Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTSMReturnToCaller	Carbon does not support Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTSMSetState	Carbon does not support Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTSMWaitForComplete	Carbon does not support Open Transport configuration APIs because the Mac OS X networking stack is not based on STREAMS.
OTScheduleDriverDeferredTask	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
OTScheduleInterruptTask	Carbon applications cannot be called from drivers.
OTScheduleSystemTask	This function will not be supported because Apple has not identified a specific developer need for it.

Unsupported functions	Porting notes
OTSndReply	Open Transport\qs connection-oriented transaction-based endpoint feature will not be supported.
OTSndRequest	Open Transport\qs connection-oriented transaction-based endpoint feature will not be supported.
OTSndUReply	Carbon does not support transaction-oriented endpoints.
OTSndURequest	Carbon does not support transaction-oriented endpoints.
OTStreamClose	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
OTStreamGetMessage	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
OTStreamGetPriorityMessage	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
OTStreamInstallNotifier	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
OTStreamIoctl	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
OTStreamIsBlocking	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
OTStreamIsSynchronous	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
OTStreamOpen	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
OTStreamPipe	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
OTStreamPoll	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
OTStreamPutMessage	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
OTStreamPutPriorityMessage	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
OTStreamRead	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
OTStreamRemoveNotifier	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.

Unsupported functions	Porting notes
OTStreamSetAsynchronous	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
OTStreamSetBlocking	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
OTStreamSetControlMask	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
OTStreamSetNonBlocking	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
OTStreamSetSynchronous	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
OTStreamUseSyncIdleEvents	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
OTStreamWrite	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
OTStrlog	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
OTSync	Due to architectural changes, Carbon will not support this function.
OTTransferProviderOwnership	Due to architectural changes, Carbon will not support this function.
OTUnholdThisCFMLibrary	Carbon does not support direct manipulation of CFM or ASLM libraries via the Open Transport API.
OTUnloadASLMLibrary	Carbon does not support direct manipulation of CFM or ASLM libraries via the Open Transport API.
OTUnregisterAsClient	Apple has not identified a specific developer need for this function.
OTUnregisterPort	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
OTWhoAmI	This function will not be supported because Apple has not identified a specific developer need for it.
OTYieldPortRequest	Carbon does not support sophisticated Open Transport port management.
StoreIntoNetbuf	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
StoreMsgIntoNetbuf	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.

Unsupported functions	Porting notes
UnloadUnusedLibraries	Carbon does not support direct manipulation of CFM or ASLM libraries via the Open Transport API.
adjmsg	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
allocb	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
allocbi	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
allocq	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
backq	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
bcanput	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
bcanputnext	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
bufcall	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
canput	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
canputnext	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
cmn_err	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
copyb	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.

Unsupported functions	Porting notes
copymsg	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
drv_priv	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
dupb	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
dupmsg	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
esballoc	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
esballoca	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
esbbcall	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
flushband	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
flushq	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
freeb	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
freemsg	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
freeq	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
freezestr	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.

Unsupported functions	Porting notes
getadmin	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
getmid	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
getmsg	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
getpmsg	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
getq	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
insq	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_allocq	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_bcmp	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_bufcall	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_close_comm	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_close_detached	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_copy_done	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_copy_set_rval	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.

Unsupported functions	Porting notes
mi_copy_state	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_copyin	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_copyout	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_copyout_alloc	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_detach	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_next_ptr	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_offset_param	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_offset_paramc	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_open_comm	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_open_detached	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_reallocb	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_reuse_proto	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_set_sth_hiwat	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.

Unsupported functions	Porting notes
mi_set_sth_lowat	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_set_sth_maxblk	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_set_sth_wroff	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_sprintf	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_timer	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_timer_alloc	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_timer_cancel	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_timer_free	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_timer_q_switch	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_timer_valid	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_tpi_ack_alloc	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_tpi_conn_con	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_tpi_conn_ind	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.

Unsupported functions	Porting notes
mi_tpi_conn_req	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_tpi_data_ind	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_tpi_data_req	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_tpi_discon_ind	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_tpi_discon_req	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_tpi_err_ack_alloc	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_tpi_exdata_ind	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_tpi_exdata_req	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_tpi_info_req	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_tpi_ok_ack_alloc	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_tpi_ordrel_ind	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_tpi_ordrel_req	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_tpi_uderror_ind	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.

Unsupported functions	Porting notes
mi_tpi_unitdata_ind	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mi_tpi_unitdata_req	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mpnotify	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mps_become_writer	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mps_intr_disable	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
mps_intr_enable	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
msgdsize	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
msgpullup	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
poll	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
pullupmsg	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
put	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
putbq	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
putctl	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.

Unsupported functions	Porting notes
putctl1	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
putctl2	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
puthere	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
putmsg	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
putnext	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
putnextctl	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
putnextctl1	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
putpmsg	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
putq	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
qenable	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
qprocsoff	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
qprocson	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
qreply	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.

Unsupported functions	Porting notes
qsize	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
rmvb	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
rmvq	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
stream_asynchronous	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
stream_blocking	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
stream_close	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
stream_installnotifier	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
stream_ioctl	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
stream_isblocking	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
stream_issynchronous	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
stream_nonblocking	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
stream_open	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
stream_pipe	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
stream_read	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
stream_synchronous	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.
stream_write	Carbon does not support any STREAMS functionality because the STREAMS subsystem is not available on Mac OS X.

Unsupported functions	Porting notes
strlog	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
strqget	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
strqset	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
t_accept	This function will not be supported because Apple has not identified a specific developer need for it.
t_alloc	This function will not be supported because Apple has not identified a specific developer need for it.
t_asynchronous	This function will not be supported because Apple has not identified a specific developer need for it.
t_bind	This function will not be supported because Apple has not identified a specific developer need for it.
t_blocking	This function will not be supported because Apple has not identified a specific developer need for it.
t_cancelreply	This function will not be supported because Apple has not identified a specific developer need for it.
t_cancelrequest	This function will not be supported because Apple has not identified a specific developer need for it.
t_cancelsynchronouscalls	This function will not be supported because Apple has not identified a specific developer need for it.
t_cancelureply	This function will not be supported because Apple has not identified a specific developer need for it.
t_cancelurequest	This function will not be supported because Apple has not identified a specific developer need for it.
t_close	This function will not be supported because Apple has not identified a specific developer need for it.
t_connect	This function will not be supported because Apple has not identified a specific developer need for it.
t_error	This function will not be supported because Apple has not identified a specific developer need for it.
t_free	This function will not be supported because Apple has not identified a specific developer need for it.

Unsupported functions	Porting notes
t_getinfo	This function will not be supported because Apple has not identified a specific developer need for it.
t_getprotaddr	This function will not be supported because Apple has not identified a specific developer need for it.
t_getstate	This function will not be supported because Apple has not identified a specific developer need for it.
t_installnotifier	This function will not be supported because Apple has not identified a specific developer need for it.
t_isnonblocking	This function will not be supported because Apple has not identified a specific developer need for it.
t_issynchronous	This function will not be supported because Apple has not identified a specific developer need for it.
t_listen	This function will not be supported because Apple has not identified a specific developer need for it.
t_look	This function will not be supported because Apple has not identified a specific developer need for it.
t_nonblocking	This function will not be supported because Apple has not identified a specific developer need for it.
t_open	This function will not be supported because Apple has not identified a specific developer need for it.
t_optmgmt	This function will not be supported because Apple has not identified a specific developer need for it.
t_rcv	This function will not be supported because Apple has not identified a specific developer need for it.
t_rcvconnect	This function will not be supported because Apple has not identified a specific developer need for it.
t_rcvdis	This function will not be supported because Apple has not identified a specific developer need for it.
t_rcvrel	This function will not be supported because Apple has not identified a specific developer need for it.
t_rcvreply	This function will not be supported because Apple has not identified a specific developer need for it.
t_rcvrequest	This function will not be supported because Apple has not identified a specific developer need for it.
t_rcvudata	This function will not be supported because Apple has not identified a specific developer need for it.

Unsupported functions	Porting notes
t_rcvuderr	This function will not be supported because Apple has not identified a specific developer need for it.
t_rcvureply	This function will not be supported because Apple has not identified a specific developer need for it.
t_rcvurequest	This function will not be supported because Apple has not identified a specific developer need for it.
t_removenotifier	This function will not be supported because Apple has not identified a specific developer need for it.
t_resolveaddr	This function will not be supported because Apple has not identified a specific developer need for it.
t_snd	This function will not be supported because Apple has not identified a specific developer need for it.
t_snddis	This function will not be supported because Apple has not identified a specific developer need for it.
t_sndrel	This function will not be supported because Apple has not identified a specific developer need for it.
t_sndreply	This function will not be supported because Apple has not identified a specific developer need for it.
t_sndrequest	This function will not be supported because Apple has not identified a specific developer need for it.
t_sndudata	This function will not be supported because Apple has not identified a specific developer need for it.
t_sndureply	This function will not be supported because Apple has not identified a specific developer need for it.
t_sndurequest	This function will not be supported because Apple has not identified a specific developer need for it.
t_sync	This function will not be supported because Apple has not identified a specific developer need for it.
t_synchronous	This function will not be supported because Apple has not identified a specific developer need for it.
t_unbind	This function will not be supported because Apple has not identified a specific developer need for it.
t_usesyncidleevents	This function will not be supported because Apple has not identified a specific developer need for it.

Unsupported functions	Porting notes
testb	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
unbufcall	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
unfreezestr	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.
unlinkb	All of the functions in OpenTransportKernel.h are unsupported by Carbon because the STREAMS subsystem is not available on Mac OS X.

Document Revision History

This table describes the changes to *Open Transport Reference*.

Date	Notes
2005-07-07	Corrected a link to a URL.
2003-05-01	Revised introduction. Added documentation to many data types and constants.
2003-03-01	Added documentation to all functions.
2003-02-01	Updated formatting and linking. Moved unsupported functions to Appendix A.

Document Revision History

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

X