

# SCE-RT SDK Distributed Memory Engine (DME) API Release 2.10

Reference

**May 2005**

© 2005 Sony Computer Entertainment Inc.

All rights reserved.

Publication date: May 2005

Sony Computer Entertainment Inc.  
2-6-21, Minami-Aoyama, Minato-ku  
Tokyo 107-0062, Japan

Sony Computer Entertainment America  
919 E. Hillsdale Blvd.  
Foster City, CA 94404, U.S.A

Sony Computer Entertainment Europe  
30 Golden Square  
London W1F 9LD, U.K.


The *SCE-RT SDK Distributed Memory Engine (DME) API Release 2.10 – Reference* is supplied pursuant to and subject to the terms of the Sony Computer Entertainment PlayStation® license agreements.

The *SCE-RT SDK Distributed Memory Engine (DME) API Release 2.10 – Reference* is intended for distribution to and use by only Sony Computer Entertainment licensed Developers and Publishers in accordance with the PlayStation® license agreements.

Unauthorized reproduction, distribution, lending, rental or disclosure to any third party, in whole or in part, of this document is expressly prohibited by law and by the terms of the Sony Computer Entertainment PlayStation® license agreements.

Ownership of the physical property of the document is retained by and reserved by Sony Computer Entertainment. Alteration to or deletion, in whole or in part, of the document, its presentation, or its contents is prohibited.

The information in the *SCE-RT SDK Distributed Memory Engine (DME) API Release 2.10 – Reference* is subject to change without notice. The content of this document is Confidential Information of Sony Computer Entertainment.

 and PlayStation are registered trademarks of Sony Computer Entertainment Inc. All other trademarks are property of their respective owners and/or their licensors.

# Table of Contents

<b>About This Manual</b>	<b>xi</b>
Changes Since Last Release	xi
Related Documentation	xi
Manual Structure	xi
Developer Reference Series	xii
Typographic Conventions	xii
Developer Support	xii
<b>Chapter 1: Defines/Macros</b>	<b>1-1</b>
NET_ADDRESS_LIST_COUNT	1-3
NET_ALL_CLIENTS	1-4
NET_CLIENT_LIMIT	1-5
NET_CLIENT_MASK	1-6
NET_CONNECTION_INVALID	1-7
NET_CONNECTION_UDP	1-8
NET_DEFAULT_RECEIVE_BUFFER_SIZE	1-9
NET_DEFAULT_SEND_BUFFER_SIZE	1-10
NET_DELIVERY_CRITICAL	1-11
NET_FULL_OBJECT_UPDATE	1-12
NET_INVALID_CLIENT_INDEX	1-13
NET_LANFIND_FILTER_APP	1-14
NET_LANFIND_FILTER_PLATFORM	1-15
NET_LATENCY_CRITICAL	1-16
NET_MAX_ADDRESS_STR_LENGTH	1-17
NET_MAX_APPLICATION_CHAR_LEN	1-18
NET_MAX_APPLICATION_NAME_LEN	1-19
NET_MAX_APPLICATION_NAME_SIZE	1-20
NET_MAX_BITMASK_ARRAY	1-21
NET_MAX_CLIENT_NAME_LENGTH	1-22
NET_MAX_CONNECTIONS	1-23
NET_MAX_HOSTNAME_LENGTH	1-24
NET_MAX_IP_LENGTH	1-25
NET_MAX_LANFIND_DETAILS_SIZE	1-26
NET_MAX_MEDIA_CHANNELS	1-27
NET_MAX_NETADDRESS_LENGTH	1-28
NET_MAX_OBJECT_NAME_LENGTH	1-29
NET_MAX_RESPONSES	1-30
NET_MAX_STRUCT_NAME_LENGTH	1-31
NET_NO_CONNECTION	1-32
NET_OBJECT_NOT_FILTERED	1-33
NET_OBJECT_OWNERSHIP_SHARED	1-34
NET_OBJECT_OWNERSHIP_UPDATE	1-35
NET_ORDER_CRITICAL	1-36
NET_SEND_TO_ALL_CLIENTS	1-37
NET_SEND_TO_CLIENT_MASK	1-38
NET_SEND_TO_SERVER	1-39
NET_SERVER_QOS_CRITICAL	1-40
NET_SESSION_KEY_LEN	1-41
NET_TIMESTAMP_STRING_LENGTH	1-42

NET_TOKEN_FREE_OWNER	1-43
NET_VERSION_STRING_LENGTH	1-44
NetRegisterObjectField	1-45
<b>Chapter 2: Enumerated Types</b>	<b>2-1</b>
EnumNetPlatformID	2-3
NetAddressType	2-4
NetCharacterEncodingType	2-5
NetClientEventType	2-6
NetClientStatus	2-7
NetConnectFailureReason	2-8
NetConnectionType	2-9
NetConnectivityType	2-10
NetConnectStatus	2-11
NetDisconnectReason	2-12
NetErrorCode	2-13
NetFieldTypes	2-21
NetLanguageType	2-22
NetMessageClass	2-23
NetObjectLifespan	2-24
NetObjectOwnershipType	2-25
NetOwnershipStatus	2-26
NetSessionType	2-27
NetStreamMediaAudioType	2-28
NetStreamMediaGridType	2-29
NetSystemStatus	2-30
NetThresholdMethod	2-31
NetUpdateType	2-32
<b>Chapter 3: Typedefs</b>	<b>3-1</b>
HDME	3-3
<b>Chapter 4: Structures</b>	<b>4-1</b>
NetAddress	4-3
NetAddressList	4-4
NetAudioDataCharacteristics	4-5
NetBandwidthInfo	4-6
NetBitMask	4-7
NetClientList	4-8
NetClientMetric	4-9
NetColorArray	4-10
NetCompletionData	4-11
NetConnectInParams	4-12
NetConnectionInfo	4-14
NetConnectionStatus	4-16
NetConnectOutParams	4-17
NetData	4-18
NetDisconnectParams	4-19
NetDmeVersion	4-20
NetEnableLanMessagingInParams	4-21
NetErrorThresholdCallbackData	4-22
NetHostPeerToPeerInParams	4-23
NetHostPeerToPeerOutParams	4-25

NetIncomingClientInParams	4-26
NetIncomingClientOutParams	4-27
NetInitializeInParams	4-28
NetInitializeOutParams	4-30
NetJoinInParams	4-31
NetJoinOutParams	4-32
NetLANFindCallbackDataArgs	4-33
NetLANFindExchangeCallbackInArgs	4-34
NetLANFindExchangeCallbackOutArgs	4-35
NetLANFindInParams	4-36
NetLANPeerDesc	4-37
NetLanRawMessageCallbackParams	4-38
NetLANSendRawMessageInParams	4-39
NetLANSendTextMessageInParams	4-40
NetLanTextMessageCallbackParams	4-41
NetLatencyMetricsDataArgs	4-42
NetLatencyMetricsInfo	4-43
NetLatencyMetricsParams	4-44
NetLocalizationParams	4-45
NetMemoryCallbackParams	4-46
NetObjectFilterData	4-47
NetPeerToPeerHostChangeData	4-48
NetRegisterObjectFilterInParams	4-49
NetRegisterObjectFilterOutParams	4-50
NetRemoteClientEventData	4-51
NetResolveAddrData	4-52
NetResolveAddrInParams	4-53
NetResolveAddrOutParams	4-54
NetRGBArray	4-55
NetSendMessageInParams	4-56
NetSendMessageOutParams	4-57
NetSMChangeData	4-58
NetStreamMediaAudioPlayData	4-59
NetStreamMediaAudioRecordData	4-60
NetStreamMediaChannelInfo	4-61
NetStreamMediaChannelStateData	4-62
NetStreamMediaClientInfo	4-63
NetStreamMediaCustomVideoPlayData	4-64
NetStreamMediaCustomVideoRecordData	4-65
NetStreamMediaIgnoreData	4-66
NetStreamMediaParams	4-67
NetStreamMediaVideoPlayData	4-69
NetStreamMediaVideoRecordData	4-70
NetSystemStatusData	4-71
NetTokenOwnershipNotifyData	4-72
NetTokenParams	4-73
NetTypeBroadcastSchedule	4-74
NetTypeClient	4-75
NetTypeClientConnectCallbackData	4-76
NetTypeConnectCallbackData	4-77
NetTypeDataStream	4-78

NetTypeDoubleVector2	4-79
NetTypeDoubleVector3	4-80
NetTypeField	4-81
NetTypeFloatVector2	4-82
NetTypeFloatVector3	4-83
NetTypeIntVector2	4-84
NetTypeIntVector3	4-85
NetTypeLookupParams	4-86
NetTypeLookupResponse	4-87
NetTypeObject	4-88
NetTypeOwnershipRequestData	4-89
NetTypeOwnershipUpdateData	4-90
NetTypeShortVector2	4-91
NetTypeShortVector3	4-92
NetTypeStructure	4-93
NetTypeSystemMessageData	4-94
NetUpdateConnErrors	4-95
NetUpdateError	4-96
NetVideoDataCharacteristics	4-97
RSA_KEY	4-98
RSA_KEYPAIR	4-99
<b>Chapter 5: Callback Functions</b>	<b>5-1</b>
NET_LAN_RAW_MESSAGE_CALLBACK	5-3
NET_LAN_TEXT_MESSAGE_CALLBACK	5-4
NetFreeCallback	5-5
NetLANFindCallback	5-6
NetLANFindExchangeCallback	5-7
NetMallocCallback	5-8
NetReallocCallback	5-9
NetTypeClientConnectCallback	5-10
NetTypeCompletionCallback	5-11
NetTypeConnectCallback	5-12
NetTypeDataStreamEndCallback	5-13
NetTypeDataStreamFilterCallback	5-14
NetTypeDataStreamUpdateCallback	5-15
NetTypeErrorThresholdCallback	5-16
NetTypeLatencyMetricsCallback	5-17
NetTypeLookupCallback	5-18
NetTypeMessageParser	5-19
NetTypeObjectCallback	5-20
NetTypeObjectFilterCallback	5-21
NetTypeObjectUpdateCallback	5-22
NetTypeOwnershipRequestCallback	5-23
NetTypeOwnershipUpdateCallback	5-24
NetTypePeerToPeerHostChangeCallback	5-25
NetTypePingCallback	5-26
NetTypeRemoteClientEventCallback	5-27
NetTypeResolveAddrCallback	5-28
NetTypeSMChangeCallback	5-29
NetTypeStreamMediaAudioPlayCallback	5-30

NetTypeStreamMediaAudioRecordCallback	5-31
NetTypeStreamMediaCustomVideoPlayCallback	5-32
NetTypeStreamMediaCustomVideoRecordCallback	5-33
NetTypeStreamMediaVideoPlayCallback	5-34
NetTypeStreamMediaVideoRecordCallback	5-35
NetTypeSystemMessageCallback	5-36
NetTypeSystemStatusCallback	5-37
NetTypeTokenOwnershipNotifyCallback	5-38
<b>Chapter 6: Functions</b>	<b>6-1</b>
KM_GetSoftwareID	6-3
NetAddressToStringAddress	6-4
NetAddToDataStream	6-5
NetAssignLocalServer	6-6
NetBitMaskIsSet	6-7
NetBitMaskSet	6-8
NetBitMaskUnSet	6-9
NetClose	6-10
NetConnect	6-11
NetCreateObject	6-12
NetDataStreamBytesFree	6-13
NetDisableLanMessaging	6-14
NetDisconnect	6-15
NetEnableLANMessaging	6-16
NetEndDataStream	6-17
NetFreeAllObjects	6-18
NetFreeObject	6-19
NetGenerateClientList	6-20
NetGenerateJoinedClientList	6-21
NetGetAverageDelayToClient	6-22
NetGetBandwidthInfo	6-23
NetGetBufferStatus	6-24
NetGetBuildTimeStamp	6-25
NetGetClient	6-26
NetGetClientIpAddress	6-27
NetGetClientStatus	6-28
NetGetClientVersion	6-29
NetGetConnectionStatus	6-30
NetGetConnectStatus	6-31
NetGetDataStream	6-32
NetGetHostByName	6-33
NetGetLatencyMetrics	6-34
NetGetLocalTime	6-35
NetGetMyClientIndex	6-36
NetGetMyIpAddress	6-37
NetGetMyNetAddress	6-38
NetGetNetUpdateErrors	6-39
NetGetObject	6-40
NetGetPeerToPeerHostClientIndex	6-41
NetGetServerVersion	6-42
NetGetSessionMasterClientIndex	6-43

NetGetTime	6-44
NetGetValidClientCount	6-45
NetHostPeerToPeer	6-46
NetIncomingClient	6-47
NetInitialize	6-48
NetJoin	6-49
NetLANFind	6-50
NetLANFindCancel	6-51
NetLANFindEnableExchange	6-52
NetLANSendRawMessage	6-53
NetLANSendTextMessage	6-54
NetLANSetDefaultEnableMessagingInParams	6-55
NetLANSetDefaultSendRawMessageInParams	6-56
NetLANSetDefaultSendTextMessageInParams	6-57
NetLANSetUserName	6-58
NetLeave	6-59
NetObjectField	6-60
NetOpenDataStream	6-61
NetPing	6-62
NetPingIP	6-63
NetPingNetAddress	6-64
NetRegisterApplicationMessage	6-65
NetRegisterDataStream	6-66
NetRegisterMemoryCallbacks	6-67
NetRegisterMessage	6-68
NetRegisterObjectFilter	6-69
NetRegisterObjectStart	6-70
NetRegisterRemoteObjectCallback	6-71
NetRegisterStructure	6-72
NetReleaseObjectPrivateOwnership	6-73
NetRequestCreateRemoteNamedObject	6-74
NetRequestObjectPrivateOwnership	6-75
NetResolveAddr	6-76
NetSendApplicationMessage	6-77
NetSendAppMessage	6-78
NetSendFieldUpdates	6-79
NetSendMessage	6-80
NetSendMyClientUpdate	6-81
NetSendObjectFullUpdate	6-82
NetSetDefaultAppMessageParams	6-83
NetSetDefaultBitMask	6-84
NetSetDefaultConnectParams	6-85
NetSetDefaultDisconnectParams	6-86
NetSetDefaultHostPeerToPeerParams	6-87
NetSetDefaultIncomingClientParams	6-88
NetSetDefaultInitializeParams	6-89
NetSetDefaultJoinParams	6-90
NetSetDefaultLANFindParams	6-91
NetSetDefaultLatencyMetricsParams	6-92
NetSetDefaultLookupParams	6-93
NetSetDefaultMemoryCallbackParams	6-94



NetSetDefaultRegisterObjectFilterParams	6-95
NetSetDefaultResolveAddrParams	6-96
NetSetDefaultStreamMediaParams	6-97
NetSetFieldChanged	6-98
NetSetMyClientObject	6-99
NetSetMyClientReceiveBroadcast	6-100
NetSetNATServiceAddr	6-101
NetSetSendAggregationInterval	6-102
NetStreamMediaEndRecording	6-103
NetStreamMediaGetChannelInfo	6-104
NetStreamMediaGetClientInfo	6-105
NetStreamMediaGetCurrentChannelState	6-106
NetStreamMediaJoinChannel	6-107
NetStreamMediaQuitChannel	6-108
NetStreamMediaSetDefaultIgnoreParams	6-109
NetStreamMediaSetIgnoreState	6-110
NetTick	6-111
NetTokenListRelease	6-112
NetTokenListRequest	6-113
NetTokenQuery	6-114
NetTokenRelease	6-115
NetTokenRequest	6-116
NetTokenSystemQuery	6-117
NetUpdate	6-118
NetUseACodecGSM	6-119
NetUseACodecLPC	6-120
NetUseACodecLPC10	6-121
NetUseClientServer	6-122
NetUseCommAdhoc	6-123
NetUseCommEEnet	6-124
NetUseCommInet	6-125
NetUseCommSockets	6-126
NetUseCrypt	6-127
NetUseDme	6-128
NetUseObjects	6-129
NetUsePeer2Peer	6-130
NetUseStreamMedia	6-131
NetUseVCodecJPEG	6-132

**Index****I-1**

This page intentionally left blank

---

## About This Manual

This is the *SCE-RT SDK Distributed Memory Engine (DME) API Release 2.10 – Reference*.

The SCE-RT Distributed Memory Engine (DME) builds upon the RTIME Interactive Networking Engine. This document provides an overview of the SCE-RT DME, including examples.

The DME is an interface to the SCE-RT networking engine. It simplifies the development of multi-user interactive games and applications. Games can be developed in either peer-to-peer or client/server architectures.

Please forward any questions about this document to [scert-support@scea.com](mailto:scert-support@scea.com).

## Changes Since Last Release

Please review `dme_changes.txt` in the `scert/1st_read` directory of the SCE-RT SDK distribution.

## Related Documentation

Related documentation for the SCE-RT SDK Distributed Memory Engine (DME) API Release 2.10 – Reference consists of the following:

*SCE-RT Medius API – Reference*

*SCE-RT Medius Game Communication Library (MGCL) API – Reference*

You should read this manual in conjunction with:

*SCE-RT Medius Game Communication Library (MGCL) – Overview*

*SCE-RT DME – Overview*

*SCE-RT Medius – Overview*

**Note:** the Developer Support Websites (<https://www.ps2-pro.com/> and <https://psp.scedev.net>) post current developments regarding the Network Gaming Service and also provides notice of future documentation releases and upgrades.

## Manual Structure

Section	Description
Ch. 1: Defines/Macros	Describes Defines/Macros for the DME
Ch. 2: Enumerated Types	Describes Enumerated Types for the DME
Ch. 3: Typedefs	Describes Typedefs for the DME
Ch. 4: Structures	Describes Structures for the DME
Ch. 5: Callback Functions	Describes Callback Functions for the DME
Ch. 6: Functions	Describes Functions for the DME
Index	Provides an Index for the DME

## Developer Reference Series

This manual is part of the *Developer Reference Series*, a series of technical reference volumes covering all aspects of PlayStation® development. The complete series is listed below:

Manual	Description
SCE-RT_SDK_DME_API_Overview	Distributed Memory Engine (DME) Overview. Used for in-game networked data management.
SCE-RT_SDK_DME_API_Reference	API for the DME
SCE-RT_SDK_MEDIUS_API_Overview	Medius client API overview. Used for user authentication, lobby chat, and player matching functionality.
SCE-RT_SDK_MEDIUS_API_Reference	API for the Medius client.
SCE-RT_SDK_MGCL_API_Overview	Medius Game Communication Library (MGCL) Overview. Used for game hosting and peer-to-peer play.
SCE-RT_SDK_MGCL_API_Reference	API for the MGCL.

## Typographic Conventions

Certain Typographic Conventions are used throughout this manual to clarify the meaning of the text:

Convention	Meaning
<code>courier</code>	Indicates literal program code.
<i>italic</i>	Indicates names of parameters and structure members (in structure/function definitions only).
<b>bold</b>	Indicates data types and structure/function names (in structure/function definitions only).
<a href="#">blue</a>	Indicates function name.
<a href="#">blue</a>	Indicates a hyperlink.

## Developer Support

### Sony Computer Entertainment America (SCEA)

SCEA developer support is available to licensees in North America only. You may obtain developer support or additional copies of this documentation by contacting the following addresses:

Order Information	Developer Support
Attn: Developer Tools Coordinator Sony Computer Entertainment America 919 East Hillsdale Blvd. Foster City, CA 94404, U.S.A. Tel: (650) 655-8000	E-mail: <a href="mailto:scert-support@scea.com">scert-support@scea.com</a> <a href="mailto:scea_support@ps2-pro.com">scea_support@ps2-pro.com</a> Web: <a href="https://www.ps2-pro.com/">https://www.ps2-pro.com/</a> <a href="https://psp.scedev.net">https://psp.scedev.net</a> Developer Support Hotline: (650) 655-5566 (Call Monday through Friday, 8 a.m. to 5 p.m., PST/PDT)

**Sony Computer Entertainment Europe (SCEE)**

SCEE developer support is available to licensees only in the PAL television territories (including Europe and Australasia). You may obtain developer support or additional copies of this documentation by contacting the following addresses:

Order Information	Developer Support
Attn: Development Tools Manager Sony Computer Entertainment Europe 13 Great Marlborough Street London W1F 7HP, U.K. Tel: +44 (0) 20 7859-5000	E-mail: <a href="mailto:scee_support@ps2-pro.com">scee_support@ps2-pro.com</a> Web: <a href="https://www.ps2-pro.com/">https://www.ps2-pro.com/</a> <a href="https://psp.scedev.net">https://psp.scedev.net</a> Developer Support Hotline: +44 (0) 20 7911-7711 (Call Monday through Friday, 9 a.m. to 6 p.m., GMT/BST)

This page intentionally left blank.

# Chapter 1: Defines/Macros

This page intentionally left blank.



## NET\_ADDRESS\_LIST\_COUNT

Macro: Address list count.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	rt_nettypes.h	1.00	June 12, 2001

### Syntax

```
#define NET_ADDRESS_LIST_COUNT 17
```

### Description

This macro determines the number of addresses in an address list.

### Notes

N/A

### Example

N/A

### See also

[NetAddressList](#)

## NET\_ALL\_CLIENTS

Macro: All clients flag.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	June 12, 2001

### Syntax

```
#define NET_ALL_CLIENTS -1
```

### Description

This macro defines the target flag for sending broadcast messages.

### Notes

N/A

### Example

N/A

### See also

N/A

## NET\_CLIENT\_LIMIT

Macro: Client limit.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	June 12, 2001

### Syntax

```
#define NET_CLIENT_LIMIT 256
```

### Description

This macro defines the maximum number of clients available in a SCE-RT world.

### Notes

N/A

### Example

N/A

### See also

N/A

## NET\_CLIENT\_MASK

Macro: Client Mask.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	June 12, 2001

### Syntax

```
#define NET_CLIENT_MASK NET_INVALID_CLIENT_INDEX
```

### Description

This macro defines the default bitwise operator mask used by a client mask.

### Notes

N/A

### Example

N/A

### See also

N/A

## NET\_CONNECTION\_INVALID

Macro: Connection Invalid.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	June 12, 2001

### Syntax

```
#define NET_CONNECTION_INVALID NULL
```

### Description

This macro denotes an invalid connection handle (HDME).

### Notes

N/A

### Example

N/A

### See also

N/A

## NET\_CONNECTION\_UDP

Macro: Connection index for UDP.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	June 12, 2001

### Syntax

```
#define NET_CONNECTION_UDP NET_NO_CONNECTION
```

### Description

This macro defines the connection index for messages received by UDP.

### Notes

N/A

### Example

N/A

### See also

N/A

## NET\_DEFAULT\_RECEIVE\_BUFFER\_SIZE

Macro: Default receive buffer size.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	June 12, 2001

### Syntax

```
#define NET_DEFAULT_RECEIVE_BUFFER_SIZE 8192
```

### Description

This macro defines the default SCE-RT receive buffer size.

### Notes

N/A

### Example

N/A

### See also

N/A

## NET\_DEFAULT\_SEND\_BUFFER\_SIZE

Macro: Default send buffer size.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	June 12, 2001

### Syntax

```
#define NET_DEFAULT_SEND_BUFFER_SIZE 8192
```

### Description

This macro defines the default SCE-RT send buffer size.

### Notes

N/A

### Example

N/A

### See also

N/A



## NET\_DELIVERY\_CRITICAL

Macro: Transport flag - delivery critical.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	dme.h	1.00	June 12, 2001

### Syntax

```
#define NET_DELIVERY_CRITICAL 0x40
```

### Description

This transport flag guarantees delivery of the message being sent.

### Notes

N/A

### Example

N/A

### See also

[NetSendMessage\(\)](#), [NetSendApplicationMessage\(\)](#), [NetStreamMediaAudioRecordData](#),  
[NetStreamMediaCustomVideoRecordData](#)

## NET\_FULL\_OBJECT\_UPDATE

Macro: Full NetObject update flag.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	June 12, 2001

### Syntax

```
#define NET_FULL_OBJECT_UPDATE -1
```

### Description

This macro defines the flag received when a NetObject full update has been received. When the FieldIndex of a [NetTypeObjectUpdateCallback\(\)](#) is set to NET\_FULL\_OBJECT\_UPDATE, then the application will know that all fields within the given NetObject will have been updated and/or changed.

### Notes

N/A

### Example

N/A

### See also

[NetTypeObjectUpdateCallback\(\)](#)

## NET\_INVALID\_CLIENT\_INDEX

Macro: Invalid client index flag.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	June 12, 2001

### Syntax

```
#define NET_INVALID_CLIENT_INDEX 256
```

### Description

This macro defines the invalid client index flag. A client index is often set to this value during error conditions depending on how client index is being used. This also helps with the bitmask client mask macros.

### Notes

N/A

### Example

N/A

### See also

N/A

## NET\_LANFIND\_FILTER\_APP

Macro: LANFind filter based on ApplicationID.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_lanfind.h	2.09	November 11, 2004

### Syntax

```
#define NET_LANFIND_FILTER_APP 0x1
```

### Description

This macro defines the bitmask used to set LANFind to search for games only with the same ApplicationID.

### Notes

[NetSetDefaultLANFindParams\(\)](#) will, by default, set the Filter field of [NetLANFindInParams](#) to NET\_LANFIND\_FILTER\_PLATFORM. To search for all applications and all platforms, set the Filter field to zero.

### Example

N/A

### See also

[NetLANFindInParams](#)

## NET\_LANFIND\_FILTER\_PLATFORM

Macro: LANFind filter based on platform.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_lanfind.h	2.09	November 11, 2004

### Syntax

```
#define NET_LANFIND_FILTER_PLATFORM 0x2
```

### Description

This macro defines the bitmask used to set LANFind to search for games only on the same platform.

### Notes

[NetSetDefaultLANFindParams\(\)](#) will, by default, set the Filter field of [NetLANFindInParams](#) to NET\_LANFIND\_FILTER\_PLATFORM. To search for all applications and all platforms, set the Filter field to zero.

### Example

N/A

### See also

[NetLANFindInParams](#)

## NET\_LATENCY\_CRITICAL

Macro: Transport flag - Overrides aggregation.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	June 12, 2001

### Syntax

```
#define NET_LATENCY_CRITICAL 0x80
```

### Description

This transport flag overrides send buffer aggregation for the given message. Skipping aggregation means that this message will be sent as quickly as possible. Skipping aggregation costs more in network bandwidth, so care must be taken when defining what messages should be set to Latency Critical.

### Notes

N/A

### Example

N/A

### See also

[NetSendMessage\(\)](#), [NetSendApplicationMessage\(\)](#), [NetStreamMediaAudioRecordData](#),  
[NetStreamMediaCustomVideoRecordData](#)

## NET\_MAX\_ADDRESS\_STR\_LENGTH

Macro: Address string maximum length.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_nettypes.h	1.00	June 12, 2001

### Syntax

```
#define NET_MAX_ADDRESS_STR_LENGTH 18
```

### Description

This macro defines the maximum length of an IP address (16 bytes) or MAC Address (18 bytes) (including the null terminator).

### Notes

N/A

### Example

N/A

### See also

N/A

## NET\_MAX\_APPLICATION\_CHAR\_LEN

Macro: Application character maximum length.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	June 12, 2001

### Syntax

```
#define NET_MAX_APPLICATION_CHAR_LEN (3)
```

### Description

This macro defines the maximum number of bytes per character - UTF-8

### Notes

N/A

### Example

N/A

### See also

N/A



## NET\_MAX\_APPLICATION\_NAME\_LEN

Macro: Application name maximum length.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	dme.h	1.00	June 12, 2001

### Syntax

```
#define NET_MAX_APPLICATION_NAME_LEN (24)
```

### Description

This macro defines the maximum length for the application's name.

### Notes

N/A

### Example

N/A

### See also

[NetInitializeInParams](#), [NetLANPeerDesc](#)

## NET\_MAX\_APPLICATION\_NAME\_SIZE

Macro: Application name maximum size.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	June 12, 2001

### Syntax

```
#define NET_MAX_APPLICATION_NAME_SIZE  
(NET_MAX_APPLICATION_NAME_LEN*NET_MAX_APPLICATION_CHAR_LEN)
```

### Description

This macro defines the maximum allowable size of the application's name.

### Notes

N/A

### Example

N/A

### See also

[NetInitializeInParams](#), [NetLANPeerDesc](#)

## NET\_MAX\_BITMASK\_ARRAY

Macro: Number of bitmask bytes in a bitmask array.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_nettypes.h	1.00	June 12, 2001

### Syntax

```
#define NET_MAX_BITMASK_ARRAY 8
```

### Description

This macro defines the number of bitmask bytes in a bitmask array.

### Notes

N/A

### Example

N/A

### See also

[NetBitMask](#)

## NET\_MAX\_CLIENT\_NAME\_LENGTH

Macro: Client name maximum length.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	June 12, 2001

### Syntax

```
#define NET_MAX_CLIENT_NAME_LENGTH 12
```

### Description

This macro defines the maximum length of a DME client name.

### Notes

N/A

### Example

N/A

### See also

[NetJoinInParams](#), [NetTypeClient](#)

## NET\_MAX\_CONNECTIONS

Macro: Maximum number of SCE-RT connections.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	June 12, 2001

### Syntax

```
#define NET_MAX_CONNECTIONS 4
```

### Description

This macro defines the maximum number of simultaneous SCE-RT connections that can be connected at any given time.

### Notes

N/A

### Example

N/A

### See also

[NetUpdateConnErrors](#)

## NET\_MAX\_HOSTNAME\_LENGTH

Macro: Hostname maximum length.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	June 12, 2001

### Syntax

```
#define NET_MAX_HOSTNAME_LENGTH 256
```

### Description

This macro defines the maximum length of the host name for performing DNS lookups.

### Notes

N/A

### Example

N/A

### See also

[NetTypeLookupParams](#)

## NET\_MAX\_IP\_LENGTH

Macro: IP maximum length.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	rt_nettypes.h	1.00	June 12, 2001

### Syntax

```
#define NET_MAX_IP_LENGTH 16
```

### Description

This macro defines the length of an IP address string including the null terminator.

### Notes

N/A

### Example

N/A

### See also

N/A

## NET\_MAX\_LANFIND\_DETAILS\_SIZE

Macro: LANFind details size.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	rt_lanfind.h	2.09	November 11, 2004

### Syntax

```
#define NET_MAX_LANFIND_DETAILS_SIZE (MAX_LANFIND_DETAILS_SIZE)
```

### Description

This macro defines the maximum size that can be used for the Details field as seen in [NetLANFindInParams](#).

### Notes

N/A

### Example

N/A

### See also

[NetLANFindExchangeCallbackOutArgs](#), [NetLANFindInParams](#)



## NET\_MAX\_MEDIA\_CHANNELS

Macro: Maximum number of Media Channels provided.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_media.h	1.00	June 12, 2001

### Syntax

```
#define NET_MAX_MEDIA_CHANNELS 64
```

### Description

This macro determines the maximum number of stream media channels.

### Notes

N/A

### Example

N/A

### See also

N/A

## NET\_MAX\_NETADDRESS\_LENGTH

Macro: Net address maximum length.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_nettypes.h	1.00	June 12, 2001

### Syntax

```
#define NET_MAX_NETADDRESS_LENGTH NET_MAX_IP_LENGTH
```

### Description

This macro defines the maximum length of a network address.

### Notes

N/A

### Example

N/A

### See also

N/A

## NET\_MAX\_OBJECT\_NAME\_LENGTH

Macro: NetObject name maximum length.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	June 12, 2001

### Syntax

```
#define NET_MAX_OBJECT_NAME_LENGTH 16
```

### Description

This macro defines the maximum size of the Name field in the [NetTypeObject](#) structure.

### Notes

N/A

### Example

N/A

### See also

[NetTypeObject](#)

## NET\_MAX\_RESPONSES

Macro: Maximum number of DNS responses.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	dme.h	1.00	June 12, 2001

### Syntax

```
#define NET_MAX_RESPONSES 16
```

### Description

This macro defines the maximum number of DNS lookup responses.

### Notes

N/A

### Example

N/A

### See also

[NetTypeLookupResponse](#)

## NET\_MAX\_STRUCT\_NAME\_LENGTH

Macro: Message structure's maximum name length.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	June 12, 2001

### Syntax

```
#define NET_MAX_STRUCT_NAME_LENGTH 32
```

### Description

This macro defines the maximum length of the Name field in the [NetTypeStructure](#) structure.

### Notes

N/A

### Example

N/A

### See also

[NetTypeStructure](#)

## NET\_NO\_CONNECTION

Macro: No connection flag.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	June 12, 2001

### Syntax

```
#define NET_NO_CONNECTION NULL
```

### Description

Definition of a Connection Handle (HDME) that does not contain a connection.

### Notes

N/A

### Example

N/A

### See also

N/A

## NET\_OBJECT\_NOT\_FILTERED

Macro: NetObject not filtered flag.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	June 12, 2001

### Syntax

```
#define NET_OBJECT_NOT_FILTERED 0
```

### Description

This macro defines the flag used if a NetObject is not to be filtered.

### Notes

N/A

### Example

N/A

### See also

[NetTypeObject](#)

## NET\_OBJECT\_OWNERSHIP\_SHARED

Macro: NetObject ownership shared flag.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	rt_object.h	1.00	June 12, 2001

### Syntax

```
#define NET_OBJECT_OWNERSHIP_SHARED -1
```

### Description

This macro defines the flag that determines if the given NetObject is shared. It is shared if the value of the OwnerClientIndex field is NET\_OBJECT\_OWNERSHIP\_SHARED.

### Notes

N/A

### Example

N/A

### See also

[NetTypeObject](#)



## NET\_OBJECT\_OWNERSHIP\_UPDATE

Macro: NetObject ownership update flag.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	June 12, 2001

### Syntax

```
#define NET_OBJECT_OWNERSHIP_UPDATE -2
```

### Description

This macro defines the flag that determines if a NetObject ownership has changed. If the FieldIndex field of NetTypeObjectUpdateCallback is set to NET\_OBJECT\_OWNERSHIP\_UPDATE then we know that the NetObject's OwnerClientIndex (client index of who owns the NetObject) has changed.

### Notes

N/A

### Example

N/A

### See also

[NetTypeObject](#), [NetTypeObjectUpdateCallback](#)

## NET\_ORDER\_CRITICAL

Macro: Transport flag - order critical.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	June 12, 2001

### Syntax

```
#define NET_ORDER_CRITICAL 0x10
```

### Description

This transport flag ensures that the order of messages will be received in the proper sequence on the receiving client. The receiving client's receive buffer may have message out of order; however, it will not pass a message up to the application layer unless the given message is the expected message next in sequence.

### Notes

N/A

### Example

N/A

### See also

[NetSendMessage\(\)](#), [NetSendApplicationMessage\(\)](#), [NetStreamMediaAudioRecordData](#),  
[NetStreamMediaCustomVideoRecordData](#)

## NET\_SEND\_TO\_ALL\_CLIENTS

Macro: Send to all clients flag.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	June 12, 2001

### Syntax

```
#define NET_SEND_TO_ALL_CLIENTS NET_ALL_CLIENTS
```

### Description

This macro defines the flag that is used for sending messages to all clients.

### Notes

N/A

### Example

N/A

### See also

[NetSendMessage\(\)](#), [NetSendApplicationMessage\(\)](#)

## NET\_SEND\_TO\_CLIENT\_MASK

Macro: Send messages based on ClientMask.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

```
#define NET_SEND_TO_CLIENT_MASK NET_CLIENT_MASK
```

### Description

Used to specify whether or not the ClientMask field of the [NetClientList](#) structure contains a bitmask of the clients.

### Notes

N/A

### Example

N/A

### See also

[NetClientList](#)

## NET\_SEND\_TO\_SERVER

Macro: Send message to server.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	June 12, 2001

### Syntax

```
#define NET_SEND_TO_SERVER 65535
```

### Description

This macro defines the flag used to determine if a message is to be sent to the server.

### Notes

N/A

### Example

N/A

### See also

[NetSendMessage\(\)](#), [NetPing\(\)](#)

## NET\_SERVER\_QOS\_CRITICAL

Macro: Transport flag - Quality of service critical.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	June 12, 2001

### Syntax

```
#define NET_SERVER_QOS_CRITICAL 0x1
```

### Description

This transport flag allows the server to act based on other transport flags.

### Notes

Reserved as a server only transport flag (clients should not use this flag).

### Example

N/A

### See also

N/A

## NET\_SESSION\_KEY\_LEN

Macro: Session key length.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	dme.h	1.00	June 12, 2001

### Syntax

```
#define NET_SESSION_KEY_LEN 17
```

### Description

This macro defines the length of the [NetConnectionInfo](#) Session Key. This length includes the NULL terminator.

### Notes

N/A

### Example

N/A

### See also

[NetConnectionInfo](#)

## NET\_TIMESTAMP\_STRING\_LENGTH

Macro: Timestamp string length.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	June 12, 2001

### Syntax

```
#define NET_TIMESTAMP_STRING_LENGTH 64
```

### Description

Maximum size of ReturnedTimeStamp parameter passed into NetGetBuildTimeStamp

### Notes

N/A

### Example

N/A

### See also

[NetGetBuildTimeStamp](#)



## NET\_TOKEN\_FREE\_OWNER

Macro: NetToken free flag.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

```
#define NET_TOKEN_FREE_OWNER 65535
```

### Description

This macro defines the flag that indicates if a given token is free.

### Notes

N/A

### Example

N/A

### See also

[NetTokenOwnershipNotifyData](#)

## NET\_VERSION\_STRING\_LENGTH

Macro: DME Version string length.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	dme.h	1.00	June 12, 2001

### Syntax

```
#define NET_VERSION_STRING_LENGTH 16
```

### Description

Format v.v.v.v. Getting both client and server versions take this form.

### Notes

N/A

### Example

N/A

### See also

[NetDmeVersion](#), [NetInitializeOutParams](#), [NetGetClientVersion\(\)](#), [NetGetServerVersion\(\)](#)

## NetRegisterObjectField

This macro (which takes the form of a function) registers a NetObject field.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtobjectPS2.a	rt_object.h	1.00	September 12, 2001

### Syntax

**#define NetRegisterObjectField(**

*ReturnField*,

Returned assigned NetObject field index.

*Struct*,

Instantiated structure representing NetObject.

*Field*,

Field of structure to register as a NetObject field.

*FieldType*,

Data type used to define the field for NetObject registration.

*ElementCount*,

Number of consecutive elements (i.e., array size).

*BroadcastSchedule*

Pointer to the field specific broadcast schedule (when to propagate data to the network).

**) NetObjectField(&ReturnField, (int) &((Struct).Field) - (int) &(Struct), \ sizeof((Struct).Field), (FieldType), (ElementCount), &(BroadcastSchedule));**

### Description

This macro (which takes the form of a function) registers a NetObject field.

### Notes

The NetRegisterObjectField macro is a more convenient method of registering object fields.

### Example

N/A

### See also

[NetRegisterObjectField](#), [NetTypeBroadcastSchedule](#), [NetCreateObject\(\)](#)

This page intentionally left blank.

## Chapter 2: Enumerated Types

This page intentionally left blank.

## EnumNetPlatformID

Enumeration: Supported platforms.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	dme.h	1.31	December 8, 2003

### Enumeration

```
typedef enum {
    EnumNetPlatformID_Unknown = 0,           (0) Unknown platform type.
    EnumNetPlatformID_PS2 = 1,               (1) PlayStation 2 platform.
    EnumNetPlatformID_PSP = 2,               (2) PlayStation Portable platform.
    ExtraEnumNetPlatformID = 0xffffffff      Ensures that all values are stored as 32-bit integers
                                              on all compilers.
} EnumNetPlatformID;
```

### Description

Field of [NetLANPeerDesc](#) that declares all possible supported platforms.

### Notes

N/A

### Example

N/A

### See also

[NetLANPeerDesc](#)

**NetAddressType**

Enumeration: Determines the type of address being used.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_nettypes.h	1.18	November 11, 2004

**Enumeration**

```

typedef enum {
    NetAddressNone = 0,
    NetAddressTypeExternal = 1,
    NetAddressTypeInternal = 2,
    NetAddressTypeNATService = 3,
    NetAddressTypeBinaryExternal = 4,
    NetAddressTypeBinaryInternal = 5,
    NetAddressTypeBinaryExternalVport = 6,
    NetAddressTypeBinaryInternalVport = 7,
    NetAddressTypeBinaryNATServices = 8,
    ExtraNetAddressType = 0xFFFFFFFF
} NetAddressType;

```

(0) This value is used to specify "Not in use"

(1) ASCII string representation of a client's public IPv4 address.

(2) ASCII string representation of a client's private IPv4 address.

(3) ASCII string representation of a NAT resolution server's IPv4 address.

(4) 4-byte binary representation of a client's public IPv4 address.

(5) 4-byte binary representation of a client's private IPv4 address.

(6) 4-byte binary representation of a client's public IPv4 address. The Port parameter contains a 2-byte virtual port in 2 high bytes and the actual network port in the 2 low bytes.

(7) 4-byte binary representation of a client's public IPv4 address. The Port parameter contains a 2-byte virtual port in 2 high bytes and the actual network port in the 2 low bytes.

(8) Contains two 4-byte binary representations of NAT resolution servers IPv4 addresses stored back to back.

Ensures that all values are stored as 32-bit integers on all compilers.

**Description**

The values in this enumeration are used for determining the type of connection being made.

**Notes**

For all binary representations of IPv4 and PSP Adhoc addresses, the data is stored as a series of octets, read from left to right. Thus for an IPv4 address: IP Address 1.2.3.4 would become 0x01, 0x02, 0x03, 0x04.

**Example**

N/A

**See also**

[NetAddressToStringAddress\(\)](#), [NetAddress](#)



## NetCharacterEncodingType

Enumeration: Supported character-encoding types.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.31	December 8, 2003

### Enumeration

```
typedef enum {
    NetCharacterEncodingNone = 0,           (0) No encoding type specified
    NetCharacterEncodingISO8859_1 = 1,      (1) (ISO-8859-1) ISO Latin 1 character set.
    NetCharacterEncodingUTF_8 = 2,          (2) (UTF-8) Unicode Transformation Format-8.
    ExtraNetCharacterEncodingType = 0xfffff Ensures that all values are stored as 32-bit integers
                                              on all compilers.
} NetCharacterEncodingType;
```

### Description

The values in this enumeration are used to identify the type of character-encoding used for system messages.

### Notes

N/A

### Example

N/A

### See also

[NetLocalizationParams](#), [NetTypeSystemMessageData](#)

## NetClientEventType

Enumeration: DME client event types.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	<a href="#">rt_object.h</a>	1.24	December 1, 2002

### Enumeration

```
typedef enum {
    NetClientEventJoin = 0,           (0) Client is joining
    NetClientEventLeave = 1,          (1) Client is leaving
    ExtraNetClientEventType = 0xffff Ensures that all values are stored as 32-bit integers
                                     on all compilers.
} NetClientEventType;
```

### Description

The values in this enumeration are used to identify a type of event in the remote client event callback (registered as a part of [NetJoin\(\)](#)).

### Notes

N/A

### Example

N/A

### See also

[NetRemoteClientEventData](#)

## NetClientStatus

Enumeration: DME client status types.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	<a href="#">rt_nettypes.h</a>	1.11	November 11, 2004

### Enumeration

```
typedef enum {
    ClientStatusNone,
    ClientStatusNotConnected,
    ClientStatusConnected,
    ClientStatusJoining,
    ClientStatusJoined,
    ClientStatusJoinedSessionMaster,
    ExtraNetClientStatus = 0xffffffff
} NetClientStatus;
```

(0) No ClientStatus is available.

(1) Client is not connected.

(2) Client is connected, but has not called [NetJoin\(\)](#).

(3) Client is in the process of joining, and is now receiving its first batch of object and field updates.

(4) The client is now fully synchronized with the game, and has received all initial object creation callbacks, etc.

(5) The client is fully joined and is *also* the Session Master.

Ensures that all values are stored as 32-bit integers on all compilers.

### Description

The values in this enumerated data type are used for determining the status of a client.

### Notes

N/A

### Example

N/A

### See also

[NetTypeClientConnectCallbackData](#), [NetGetClientStatus\(\)](#)

# NetConnectFailureReason

Enumeration: [NetConnect\(\)](#) failure reasons.

Link to file	Include file	Introduced	Last modified
<a href="#">librtbasePS2.a</a>	<a href="#">rt_nettypes.h</a>	1.03	November 11, 2004

## Enumeration

```
typedef enum {
    ConnectFailureReasonNone,           (0) No failure reason given
    ConnectFailureReasonError,          (1) An error occurred
    ConnectFailureReasonClientVer,      (2) Incompatible client version
    ConnectFailureReasonServerVer,      (3) Incompatible server version
    ConnectFailureReasonFull,           (4) World is full
    ConnectFailureReasonWorldID,        (5) Server does not support world id
    ConnectFailureReasonAuth,           (6) Failed to authenticate signature
    ConnectFailureReasonEncryption,     (7) Encryption failure
    ConnectFailureReasonAccessKey,      (8) Access Key failure
    ConnectFailureReasonAuxUDPFailure,  (9) Failed to establish aux udp
    ExtraConnectFailureReason = 0xffff  Ensures that all values are stored as 32-bit integers
                                        on all compilers
} NetConnectFailureReason;
```

## Description

The values in this enumeration are used for determining the type of failure condition if the connection failed.

## Notes

N/A

## Example

N/A

## See also

[NetTypeConnectCallbackData](#)

## NetConnectionType

Enumeration: Determines the type of connection being used.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_nettypes.h	1.18	November 11, 2004

### Enumeration

```
typedef enum {
    NetConnectionNone = 0,
    NetConnectionTypeClientServerTCP = 1,
    NetConnectionTypePeerToPeerUDP = 2,
    NetConnectionTypeClientServerTCPAuxUDP = 3,
    NetConnectionTypeClientListenerTCP = 4,
    ExtraNetConnectionType = 0xfffff
} NetConnectionType;
```

(0) This value is used to specify that no information is present

(1) This specifies a connection to a Server via TCP

(2) This specifies a connection to another peer via UDP..

(3) This specifies a connection to a Server via TCP and UDP. The UDP connection is normal UDP: there is no reliability or in-order guarantee.

(4) This specifies a connection to a Server via TCP. This is reserved for SCE-RT "Spectator" functionality.

Ensures that all values are stored as 32-bit integers on all compilers.

### Description

The values in this enumerated data type are used for determining the type of connection being made.

### Notes

N/A

### Example

N/A

### See also

[NetConnectionInfo](#)

# NetConnectivityType

Enumeration: Network connectivity information.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	dme.h	2.09	November 11, 2004

## Enumeration

```
typedef enum {  
    NetConnectivityNone,                (0) Default invalid connectivity type set by  
                                         NetSetDefaultInitializeParams.  
    NetConnectivityInternet,           (1) Connectivity to the full routable Internet is  
                                         available.  
    NetConnectivityLAN,                (2) Local Area Network gaming is the only option  
                                         and connectivity to the internet is currently not  
                                         possible.  
    ExtraNetConnectivityType = 0xFFFFFFFF  
} NetConnectivityType;
```

## Description

This is the connectivity type that will be used for this networking session.

## Notes

N/A

## Example

N/A

## See also

[NetInitializeInParams](#), [NetGetConnectivityType\(\)](#)

## NetConnectStatus

NetConnectStatus Enumerated Values indicate the current state of a connection.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_nettypes.h	1.03	November 11, 2004

### Enumeration

```
typedef enum {
    ConnectStatusClosed = 0,
    ConnectStatusDisconnected = 0,
    ConnectStatusOpen = 1,
    ConnectStatusPending = 2,
    ConnectStatusFailed = 3,
    ConnectStatusNeedDisconnect = 4,
    ConnectStatusDisconnecting = 5,
    ExtraConnectStaus = 0xffff
} NetConnectStatus;
```

(0) Disconnected from server (this enum type will be deprecated at some point).

(0) Disconnected from server

(1) Connected to server

(2) Connection to server pending

(3) Failed to connect to server

(4) NetDisconnect needs to be called

(5) NetDisconnect in progress

Ensures that all values are stored as 32-bit integers on all compilers.

### Description

The values in this enumerated data type are used for determining the current state of a connection.

### Notes

N/A

### Example

N/A

### See also

[NetConnectionStatus](#), [NetGetConnectionStatus\(\)](#), [NetGetConnectStatus\(\)](#), [NetTypeConnectCallbackData](#)

NetDisconnectReason

Enumeration: Reasons for a NetDisconnect.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_nettypes.h	2.09	November 11, 2004

Enumeration

```

typedef enum {
    NetDisconnectNone = 0,                (0) No specific reason was given
    NetDisconnectNormal = 1,              (1) Disconnect normally
    NetDisconnectConnectFail = 2,         (2) Disconnect due to failure to connect
    NetDisconnectStreamMediaFail = 3,     (3) Disconnect due to failure to enable
                                         StreamingMedia
    NetDisconnectUpdateFail = 4,          (4) Disconnect due to failure to update client
    NetDisconnectInactivity = 5,          (5) Disconnect due to session timeout
    NetDisconnectShutdown = 6,            (6) Disconnect due to shutdown
    NetDisconnectMessageLengthMismatch = 7, (7) Disconnect due maximum message length
                                         mismatch
    NetDisconnectAppDefinedStart = 128,   (128) This is the start of enum for application
                                         specified-disconnect reason
    MaxDisconnectReason = 255,            (255) This is the maximum number of disconnect
                                         reason
    ExtraNetDisconnectReason = 0xffffffff This forces enumerated types to be 32 bit integers
                                         on all compilers
} NetDisconnectReason;

```

Description

The NetDisconnectReason enumerated values indicate the reason a connection attempt failed.

Notes

N/A

Example

N/A

See also

[NetDisconnectParams](#)



## NetErrorCode

NetErrorCode Enumerated Values indicating causes of errors.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_nettypes.h	1.00	March 1, 2005

### Enumeration

```
typedef enum {
```

**NetErrorNone,**

(0) This is the default 'OK' return code. No error occurred.

**NetErrorNotConnected,**

(1) The function call can not be called while the associated HDME is not in a connected state. Currently only returned by NetGetPeerToPeerHostClientIndex.

**NetErrorConnectionLost,**

(2) The client's TCP connection to a server was lost. This could either be due to a client side sensing a timeout condition, or the server forcibly disconnecting the client. This error will usually be returned by a call to NetUpdate, but may be returned by other DME calls as well. If the error is returned by NetUpdate, use NetGetNetUpdateErrors to determine the HDME that is associated with the error. The application should determine the HDME associated with the error and cleanup with a call to NetDisconnect.

See [NetUpdate\(\)](#), [NetGetNetUpdateErrors\(\)](#), and [NetDisconnect\(\)](#).

**NetErrorConnectionFailed,**

(3) The connection attempt failed for some reason. This error is usually returned via the LocalConnectCallback passed into NetConnect. In most cases this error occurs because the game is full or the client could not allocate enough resources to establish the connection. This error could also occur when connecting to a server if there are security key configuration errors on either the client or the server. Lastly, this is the default 'catch all' return code for a call to NetConnect or NetHostPeerToPeer. The application should determine the HDME associated with the error and cleanup with a call to NetDisconnect.

See [NetConnect\(\)](#), [NetHostPeerToPeer\(\)](#), [NetDisconnect\(\)](#), and [NetUpdate\(\)](#).

**NetErrorClientRejected,**

(4) Deprecated return code. Not currently used.

**NetErrorDisconnectFailed,**

(5) Deprecated return code. Not currently used.

**NetErrorNewClient,**

(6) Deprecated return code. Not currently used.

**NetErrorBadSessionMaster,**

(7) The application requested session master OwnershipPrivate when calling [NetJoin\(\)](#), but a session master already exists for the game. This error is currently only returned via the application's LocalJoinCallback that is passed into [NetJoin\(\)](#).

See [NetJoin\(\)](#).

**NetErrorBadIndex,**

(8) The application attempted to use an index that is currently invalid. The index could either be a ClientIndex, ObjectIndex, StreamIndex, FieldIndex, or StructureIndex. A variety of DME functions may return this error code. Verify that the function returning the error has valid input parameters.

**NetErrorBadDataStreamChannel,**

(9) An invalid NetDataStream index was used. Verify that the client has properly completed their [NetJoin\(\)](#) phase and has been granted a DataStream block. Ensure that the client is the owner of the data stream and has called NetOpenDataStream. Validate stream index input parameters.

See [NetJoin\(\)](#), [NetOpenDataStream\(\)](#), and [NetAddToDataStream\(\)](#).

**NetErrorBadPointer,**

(10) A NULL pointer was encountered by the function processing. Either a NULL pointer parameter was passed in, or a passed in index refers to a NULL data member. A variety of DME API calls will return this error code. Validate pointer and index style input parameters.

**NetErrorObjectNotShared,**

(11) The request for private ownership of the NetObject failed because the object is not in a shared state. This error code is only returned by a call to NetRequestObjectPrivateOwnership.

See [NetRequestObjectPrivateOwnership\(\)](#).

**NetErrorBadPacketReceived,**

(12) A bad packet was encountered during NetUpdate processing. In most cases this error is caused by message aggregation combined with a poorly written message handler. A message handler must return the correct message size, or else the next message in the chain will not be processed properly and will cause this error. Validate that all application defined message handlers return the correct size. Pay special attention to message handlers that deal with variable size messages. See [NetRegisterMessage\(\)](#) and [NetRegisterApplicationMessages\(\)](#). Another less common cause of this error code is an invalid packet coming from a DNS server in response to a call to [NetGetHostByName\(\)](#).

See [NetTypeMessageParser](#), [NetRegisterMessage](#), [NetRegisterApplicationMessage\(\)](#), [NetUpdate\(\)](#), and [NetGetHostByName\(\)](#).

**NetErrorSendFailed,**

(13) The sending of either an application defined message or an internal DME message has failed. The most common cause of this error is a full send buffer condition, which results in a client failure to buffer the outgoing message. Verify that the application specified buffers are sufficiently sized for the expected send rate. Validate that the application send rate is not obscene. Ensure that the application is calling [NetUpdate\(\)](#) at least once per frame. If [NetUpdate\(\)](#) is not called then the send buffers will not be flushed out to the networking stack. Lastly, if the networking stack is timesharing the processor with the application validate that enough time is being relinquished to the stack and other kernel threads.

See [NetUpdate\(\)](#), [NetSendMessage\(\)](#), [NetSendApplicationMessage\(\)](#), and [NetSendAppMessage\(\)](#).

**NetErrorTimedOut,**

(14) The request processing timed out waiting for a response. This error can be associated with connections, [NetPing\(\)](#) requests, or DNS lookup requests. If the error is associated with a HDME, then the application must call [NetDisconnect\(\)](#) to properly cleanup any memory allocated for the HDME.

See [NetUpdate\(\)](#), [NetPing\(\)](#), [NetGetHostByName\(\)](#), and [NetConnect\(\)](#).

**NetErrorBadConnectionIndex,**

(15) An invalid HDME pointer was passed into the function. Most likely a NULL HDME was used, but it is also possible that [NetJoin](#) or [NetObject](#) related functions will return this error because the HDME is not the one the application used to [NetJoin](#).

See HDME.

**NetErrorBadMode,**

(16) The current configuration of the DME client does not allow for the function call to proceed. This error can be caused by not calling a prerequisite [NetUse\\*](#) function, calling [NetJoin\(\)](#) while already joined, calling [NetLeave\(\)](#) while not joined, or calling a function that is not supported by the current [NetConnectivityType](#) setting.

See HDME.

**NetErrorDmeNotInitialized,**

(17) The function called requires that the DME be initialized via the call to [NetInitialize\(\)](#) before it can be used.

See [NetInitialize\(\)](#) and [NetClose\(\)](#).

**NetErrorInitFailed,**

(18) This error code will only be returned by a call to [NetInitialize\(\)](#). The call to [NetInitialize\(\)](#) failed for one of many reasons. Verify the input parameters to [NetInitialize\(\)](#), and the settings of the [NetInitializeInParams](#) structure. If encryption is being used, be sure that valid keys have been passed in and that [NetUseCrypt\(\)](#) has been called. Ensure that there is enough heap space available for the DME to allocate the resources it needs at startup time. Memory usage can vary wildly depending on how the application wishes to use the DME and which features are enabled. Lastly, be sure that the application properly setup the input parameters structure by calling [NetSetDefaultInitializeParams\(\)](#).

See [NetInitialize\(\)](#), [NetSetDefaultInitializeParams\(\)](#), [NetUseCrypt\(\)](#), [NetRegisterMemoryCallbacks\(\)](#), and [NetInitializeInParams\(\)](#).

**NetErrorNoFreeObject,**

(19) There are no more free object slots available. Check the application's [NetObjectCount](#) specified in the [NetJoinInParams](#) and increase if necessary. This error could also mean that the application has reached the MAX\_OBJECT\_FILTERS limit with the call to [NetRegisterObjectFilter\(\)](#).

See [NetJoin](#), [NetCreateObject](#), and [NetRegisterObjectFilter](#).

**NetErrorOpenDataStreamFailed,**

(20) Creation of a [NetDataStream](#) via the call to [NetOpenDataStream](#) has failed. Check the application's [DataStreamCount](#) specified in the [NetJoinInParams](#) and increase if necessary. Ensure that there is heap memory available for allocation of buffers.

See [NetJoin\(\)](#) and [NetOpenDataStream\(\)](#).

**NetErrorClientNotValid,**

(21) A particular client's [NetTypeClient](#) structure is not yet valid. A [NetTypeClient](#) structure is created for each joined client at [NetJoin\(\)](#) time. Ensure that the client has properly completed the [NetJoin\(\)](#) phase before attempting to reference their [NetTypeClient](#) structure.

See [NetJoin\(\)](#) and [NetGetClient\(\)](#).

**NetErrorMemory,**

(22) A DME request to allocate memory from the heap has failed. Verify that there is adequate heap space for the DME client. Ensure that the proper malloc and free callbacks have been registered if the application is using [NetRegisterMemoryCallbacks](#).

See [NetRegisterMemoryCallbacks](#).

**NetErrorInvalidArg,**

(23) An invalid argument was passed into the function. Verify all input parameters and structure members of input parameters. This error may also fire if the DME encounters a NULL input parameter, even though there are other better suited error codes for such cases.

**NetErrorMsgError,**

(24) Deprecated return code. Not currently used.

<b>NetErrorTooManyPendingEvents,</b>	(25) This error will be returned in response to a call to <a href="#">NetPing()</a> or <a href="#">NetGetHostByName()</a> . <a href="#">NetPing()</a> allows for 127 ping requests to be processed simultaneously. <a href="#">NetGetHostByName()</a> allows for only one hostname lookup at a time. See <a href="#">NetPing()</a> and <a href="#">NetGetHostByName()</a> .
<b>NetErrorUDPNotEnabled,</b>	(26) The function call requires the UDP layer to be initialized by calling <a href="#">NetUsePeerToPeer()</a> and <a href="#">NetInitialize()</a> . Ensure that <a href="#">NetUsePeerToPeer()</a> and <a href="#">NetInitialize()</a> have been called and returned with no errors. See <a href="#">NetUsePeerToPeer()</a> and <a href="#">NetInitialize()</a> .
<b>NetErrorUpdate,</b> <b>NetErrorGamelsFull,</b>	(27) Deprecated return code. Not currently used. (28) Special TCP server connection failure condition. Error only returned by a <a href="#">LocalConnectCallback</a> in the case where the number of client connect requests have exceeded the <a href="#">MaxClients</a> specification passed the <a href="#">NetConnectInParams</a> structure. See <a href="#">NetConnect()</a> , <a href="#">NetConnectInParams</a> , and <a href="#">NetTypeConnectCallback</a> .
<b>NetErrorHostGameFailed,</b>	(29) This error will only be returned by a call to <a href="#">NetHostPeerToPeer</a> . Verify that all input parameters are valid. Also ensure that there is enough available heap memory for the DME to allocate necessary resources. See <a href="#">NetHostPeerToPeer()</a> .
<b>NetErrorUnknown,</b>	(30) Default 'catch-all' error code. This error should always be converted to a more meaningful error code before the function returns, thus the application should never see this error.
<b>NetErrorMsgTooLarge,</b>	(31) The attempt to send the message failed because it exceeded the DME client's maximum unfragmented message size of 512 bytes. Unreliable send paths are most affected by this limit since it is unwise to attempt to fragment unreliable data. Reliable send paths can take advantage of the DME client's built in message fragmentation capabilities and thus avoid this error. See <a href="#">NetSendMessage()</a> , <a href="#">NetSendApplicationMessage()</a> , and <a href="#">NetSendAppMessage()</a> .
<b>NetErrorSecurity,</b>	(32) This error is specific to TCP Client-Server communications, and denotes that a general security error was encountered. In almost all cases this error occurs because the client has security enabled and the server has it disabled, or vice versa. See <a href="#">NetConnect()</a> , <a href="#">NetInitialize()</a> , and <a href="#">NetUseCrypt()</a> .
<b>NetErrorNotImplemented,</b>	(33) The function is currently not implemented, or not supported by the current configuration mode of the DME.

**NetErrorBadServerVersion,**

(34) This error is specific to TCP Client-Server communications, and denotes a version incompatibility between this client version and the server version. The servers have been built to be completely backward compatible so this error will only occur if the client version is newer than the servers. Applications will usually receive this error via their LocalConnectCallback.

See NetTypeConnectCallback and [NetConnect\(\)](#).

**NetErrorCommError,**

(35) An error has occurred at the DME's communications abstraction layer. This error is always fatal and warrants calling NetDisconnect if the error is associated with a HDME.

See [NetUpdate\(\)](#) and [NetDisconnect\(\)](#).

**NetErrorBufferError,****NetErrorUDPError,**

(36) Deprecated return code. Not currently used.

(37) Generic 'catch-all' UDP communications error. This error is returned for most PeerToPeer and LAN-based communications. It can be returned for a variety of reasons, including: send failures, bad indexes, bad states, and memory allocation errors. In general this is not a fatal error condition, but usually points to improper useage of the DME client or other significant problems. A future version of the DME will deprecate this error code in favor of more helpful error codes.

**NetErrorSetDefaultsNotCalled,**

(38) The appropriate NetSetDefault\* function has not been used to properly initialize the input parameters for this function. Even if an application sets all input parameters manually it is recommended that they call NetSetDefault\* anyways. This will properly protect them against unexpected behavior that may result if additional structure members are added and not properly initialized by the application.

See NetSetDefaultInitializeParams, NetSetDefaultConnectParams, NetSetDefaultResolveAddrParams, NetSetDefaultIncomingClientParams, NetSetDefaultMemoryCallbackParams, NetSetDefaultHostPeerToPeerParams, NetSetDefaultLookupParams, and NetSetDefaultGameFindParams.

**NetErrorSizeofParamMismatch,****NetErrorDeprecated,**

(39) Deprecated return code. Not currently used.

(40) Deprecated return code. Not currently used. Deprecated functionality has been completely removed from the API and libraries.

**NetErrorStreamMedia,**

(41) This error could potentially be returned by any NetStreamMedia\* call. In most cases this error occurs because the Streaming Media layer was unable to allocate heap resources necessary to execute properly. Verify that there is enough available heap space for DME useage.

**NetErrorStreamMediaComm,**

(42) Deprecated return code. Not currently used.

**NetErrorMutex,**

(43) The API call was unable to lock the global DME mutex. The DME API is not re-entrant and mutex locking has been implemented to prevent multiple threads from using the DME simultaneously. In most cases one thread will block and wait for the other thread to complete its DME processing. If this error is encountered either the DME wasn't initialized properly with `NetInitialize` or the kernel is returning an error related to the mutex. In either case it is recommended that the application completely shut down the DME with `NetClose` and re-initialize.

**NetErrorServerError,**

(44) The DNS server encountered an error while trying to process a request generated by the call to `NetGetHostByName`. Verify all input parameters, and ensure that the DNS server has been configured properly.

See [NetGetHostByName\(\)](#).

**NetErrorHostnameError,**

(45) A host name (DNS) lookup has failed due to a problem with the host name itself, not the actual lookup process. Please review the host name passed into `NetGetHostByName` and correct any errors that may cause it to parse improperly. Ensure that the DNS server configuration contains an entry for the hostname, and that enough time has been allowed for it to propagate across the Internet.

See [NetGetHostByName\(\)](#).

**NetErrorLookupError,**

(46) The DNS server was unable to resolve the requested name to an address. Verify that the input name passed in to `NetGetHostByName` is valid and ensure that the DNS server has been configured properly.

See [NetGetHostByName\(\)](#).

**NetErrorTimebaseError,**

(47) The timebase can currently only be enabled on one HDME at a time. This error will fire if the timebase is not enabled on the connection, or if a new connection attempts to enable timebase corrections while another HDME already has it enabled. Currently only [NetGetTime\(\)](#), [NetConnect\(\)](#), or [NetHostPeerToPeer\(\)](#) will return this error.

See [NetGetTime\(\)](#), [NetConnect\(\)](#), and [NetHostPeerToPeer\(\)](#).

**NetErrorTokenNotEnabled,****NetErrorTokenError,**

(48) Deprecated return code. Not currently used.

(49) Either the token system is not enabled on the HDME or it is not ready to use if it is enabled. If the token system is not enabled, enable it by setting `bUseToken` in the [NetTokenParams](#) of the [NetConnectInParams](#) or [NetHostPeerToPeerInParams](#) structures. If the token system is enabled then the application must wait for the `NetSystemTokenReady` event via the `NetTypeSystemStatusCallback`.

See [NetTokenParams](#) and `NetTypeSystemStatusCallback`.

**NetErrorBadPort,**

(50) The `UdpBindPort` parameter was outside of the range of `MIN_UDP_BINDPORT` and `MAX_UDP_BINDPORT`.

See `NetErrorBadPort`.

**NetErrorBadConnectivityType,**

(51) The `ConnectivityType` was not set properly in the [NetInitializeInParams](#) that were passed into [NetInitialize\(\)](#).

See [NetInitialize\(\)](#).

**ExtraNetErrorCode = 0xfffff**

Guarantees that all values in this enumeration are 32-bit integer values in all compilers.

**} NetErrorCode;**

### Description

The `NetErrorCode` enumerated data type defines all possible DME API result values. Every DME API returns a value of `NetErrorCode`.

### Notes

N/A

### Example

N/A

### See also

[NetGetNetUpdateErrors\(\)](#)



## NetFieldTypes

Enumeration: Predefined SCE-RT variable types.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	June 12, 2001

### Enumeration

```
typedef enum {
    FieldTypeSignedChar,           (0) Signed character field.
    FieldTypeUnsignedChar,        (1) Unsigned character field.
    FieldTypeSignedShort,         (2) Signed short field.
    FieldTypeUnsignedShort,       (3) Unsigned short field.
    FieldTypeSignedInt,           (4) Signed integer field.
    FieldTypeUnsignedInt,         (5) Unsigned integer field.
    FieldTypeFloat,               (6) Float field.
    FieldTypeDouble,              (7) Double precision field.
    FieldTypeShortVector2,        (8) Field error based on distance
    FieldTypeShortVector3,        (9) Field error based on distance
    FieldTypeIntVector2,          (10) Field error based on distance
    FieldTypeIntVector3,          (11) Field error based on distance
    FieldTypeFloatVector2,        (12) Field error based on distance
    FieldTypeFloatVector3,        (13) Field error based on distance
    FieldTypeDoubleVector2,       (14) Field error based on distance
    FieldTypeDoubleVector3,       (15) Field error based on distance
    FieldTypeChildStructure,      (16) Field offset for user defined structures
    ExtraNetFieldTypes = 0xfffff  Ensures that all values are stored as 32-bit integers
                                   on all compilers.
} NetFieldTypes;
```

### Description

The values in this enumeration list the data types that an application can use to define fields for object registration.

### Notes

N/A

### Example

N/A

### See also

[NetRegisterObjectField](#)

## NetLanguageType

Enumeration: Supported languages.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.31	December 8, 2003

### Enumeration

```
typedef enum {
    NetLanguageNone = 0,
    NetLanguageUSEnglish = 1,
    NetLanguageUKEnglish = 2,
    NetLanguageJapanese = 3,
    NetLanguageKorean = 4,
    NetLanguageItalian = 5,
    NetLanguageSpanish = 6,
    NetLanguageGerman = 7,
    NetLanguageFrench = 8,
    NetLanguageDutch = 9,
    NetLanguagePortuguese = 10,
    NetLanguageChinese = 11,
    NetLanguageTaiwanese = 12,
    NetLanguageFinnish = 13,
    NetLanguageNorwegian = 14,
    ExtraNetLanguageType = 0xfffff
} NetLanguageType;
```

(0) Implies that no language type is specified.

(1) US English.

(2) UK English.

(3) Japanese.

(4) Korean.

(5) Italian.

(6) Spanish.

(7) German.

(8) French.

(9) Dutch.

(10) Portuguese.

(11) Chinese.

(12) Taiwanese.

(13) Finnish.

(14) Norwegian.

Ensures that all values are stored as 32-bit integers on all compilers.

### Description

The values in this enumeration are used to identify the language used for system messages sent by the servers.

### Notes

N/A

### Example

N/A

### See also

[NetLocalizationParams](#), [NetTypeSystemMessageCallback](#)

## NetMessageClass

Enumeration: Declares NetMessage class.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_nettypes.h	1.00	November 11, 2004

### Enumeration

```
typedef enum {
    MessageClassDME,
    MessageClassLobby,

    MessageClassApplication,
    MessageClassLobbyReport,

    MessageClassLobbyExt,

    MessageClassLobbyAuthentication,

    MaxMessageClasses,

    ExtraNetMessageClass = 0xfffff
} NetMessageClass;
```

(1) Identifies messages used internally by the DME.  
 (2) Identifies messages used by the Medius Lobby SDK.  
 (3) Identifies messages used by your game.  
 (4) Identifies messages used by the Medius Game Communications Library (MGCL).  
 (5) Identifies additional messages used by the Medius Lobby SDK.  
 (6) Identifies messages used during authentication. (Deprecated)  
 (7) Used as an array allocation size. Must always be the *last* valid value before ExtraNetMessageClass, not after.  
 Ensures that all values are stored as 32-bit integers on all compilers.

### Description

Message classes allow for orthogonal (non-interacting) registration of messages.

### Notes

N/A

### Example

N/A

### See also

[NetRegisterMessage\(\)](#)

## NetObjectLifespan

Enumeration: Declares NetObjects's lifespan.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	June 12, 2001

### Enumeration

```
typedef enum {
    ObjectLifespanOneUpdate,           (0) Not yet implemented
    ObjectLifespanTimeout,             (1) Not yet implemented
    ObjectLifespanClient,              (2) Persists until client's creator leaves
    ObjectLifespanSession,             (3) Persists until all clients leave
    ObjectLifespanPermanent,           (4) Persists until deleted by session master
    ExtraNetObjectLifespan = 0xfffff   Ensures that all values are stored as 32-bit integers
                                        on all compilers.
} NetObjectLifespan;
```

### Description

The values in this enumerated data type are used when defining how long an object will persist.

### Notes

N/A

### Example

N/A

### See also

[NetCreateObject\(\)](#), [NetRequestCreateRemoteNamedObject\(\)](#)

## NetObjectOwnershipType

Enumeration: Identifies the type or result of a shared NetObject ownership request.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	2.07	June 18, 2004

### Enumeration

```
typedef enum {
    NetObjectOwnershipNone = 0,           (0) No object ownership
    NetObjectOwnershipGranted = 1,        (1) Object ownership request granted
    NetObjectOwnershipDenied = 2,         (2) Object ownership request denied
    NetObjectOwnershipNotShared = 3,      (3) Object is not in a shared state
    NetObjectOwnershipShared = 4,         (4) Private ownership was release
    ExtraNetObjectOwnershipType = 0xffffffff
    Ensures that all values are stored as 32-bit integers
    on all compilers.
} NetObjectOwnershipType;
```

### Description

Identifies the type or result of a shared object ownership request.

### Notes

N/A

### Example

N/A

### See also

[NetTypeOwnershipUpdateData](#), [NetTypeOwnershipRequestCallback](#)

# NetOwnershipStatus

Enumeration: Defines NetObject ownership types.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_object.h	1.00	June 12, 2001

## Enumeration

```
typedef enum {
  OwnershipNone,
  OwnershipPrivate,
  OwnershipShared,
  OwnershipNotAvailable,
  ExtraNetOwnershipStatus = 0xffffffff
} NetOwnershipStatus;
```

(0) Not owned

(1) NetObject is currently privately owned by a given client. Any modifications to this NetObject by this client will automatically be propagated to all other clients.

(2) NetObject is currently in a shared state and ready for a client to request private ownership of this NetObject. If no client has private ownership of a given NetObject, then the client that "created" the NetObject will be responsible (i.e. the creator's modifications to the NetObject will automatically be propagated to all the other clients).

(3) Cannot be owned.

Ensures that all values are stored as 32-bit integers on all compilers.

## Description

The values in this enumeration are used for determining the ownership of a registered DME Object.

## Notes

N/A

## Example

N/A

## See also

[NetJoinInParams](#), [NetCreateObject\(\)](#), [NetRequestCreateRemoteNamedObject\(\)](#)

## NetSessionType

Enumeration: Available NetLANFind session types.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Enumeration

```
typedef enum {
    NetSessionTypeGame = 0,           (0) Peer to game session
    NetSessionTypePeer,              (1) Solitary peer session.
    NetSessionTypeIntegratedServer,  (2) Integrated TCP server sesison.
    ExtraNetSessionType = 0xffffffff Ensures that all values are stored as 32-bit integers
                                     on all compilers.
} NetSessionType;
```

### Description

Declares all available session types that may be found via the DME's NetLANFind features.

### Notes

N/A

### Example

N/A

### See also

[NetLANFindExchangeCallbackInArgs](#), [NetLANFindCallbackDataArgs](#), [NetLANFindInParams](#)

# NetStreamMediaAudioType

Enumeration: Declares the type of audio being used.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_nettypes.h	1.24	November 11, 2004

## Enumeration

```
typedef enum {  
    NetStreamMediaAudioTypeRAW = 0,           (0) RAW Unencoded audio data  
    NetStreamMediaAudioTypeCUSTOM = 1,        (1) Custom application audio data  
    NetStreamMediaAudioTypeGSM = 2,           (2) DME encoded GSM audio data  
    NetStreamMediaAudioTypeLPC = 3,           (3) DME encoded LPC audio data  
    NetStreamMediaAudioTypeLPC10 = 4,         (4) DME encoded LPC10 audio data  
    ExtraNetStreamMediaAudioType = 0xfffff    Ensures that all values are stored as 32-bit integers  
                                              on all compilers.  
} NetStreamMediaAudioType;
```

## Description

The values in this enumerated data type are used for determining which type of audio is being used in stream media.

## Notes

N/A

## Example

N/A

## See also

[NetStreamMediaEndRecording\(\)](#), [NetStreamMediaAudioRecordData](#), [NetStreamMediaAudioPlayData](#)



## NetStreamMediaGridType

Enumeration: Declares which mechanism stream media will be distributed.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_nettypes.h	1.24	November 11, 2004

### Enumeration

```
typedef enum {
    NetStreamMediaGridTypeRelay = 0,
    NetStreamMediaGridTypeDirect = 1,
    ExtraNetStreamMediaGridType = 0xfffff
} NetStreamMediaGridType;
```

(0) Audio data can be relayed through multiple peers in order to distribute the stream to all clients. The stream is sent to at most 4 other clients using this process.

(1) Audio data is sent directly to all other clients. Using this process the stream is sent to N-1 clients where N is the number of clients in the game.

Ensures that all values are stored as 32-bit integers on all compilers.

### Description

The values in this enumerated data type are used for determining which type of grid is being used in stream media.

### Notes

N/A

### Example

N/A

### See also

[NetStreamMediaParams](#)

# NetSystemStatus

Enumeration: Declares what status the NetSystem is in.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_nettypes.h	2.09	November 11, 2004

## Enumeration

```
typedef enum {  
    NetSystemTokenReady = 0x1,           (1) Token system is ready for use  
    ExtraNetSystemStatus = 0xffffffff    Ensures that all values are stored as 32-bit integers  
                                          on all compilers.  
} NetSystemStatus;
```

## Description

Enumerated data type for net system status indication, each bit for one type of state, total 32 states.

## Notes

N/A

## Example

N/A

## See also

[NetSystemStatusData](#), [NetTypeSystemStatusCallback](#)

## NetThresholdMethod

Enumeration: Declares which NetObject field change threshold method is being used.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	June 12, 2001

### Enumeration

```
typedef enum {
  NoThreshold,
  ThresholdEquality,
  ThresholdAbsoluteMagnitude,
  ThresholdRatioMagnitude,
  ThresholdAnchorDelta,
  ThresholdCallback,
  ExtraNetThresholdMethod = 0xfffff
} NetThresholdMethod;
```

(0) No Threshold.  
 (1) When the values changes.  
 (2) value changes a fixed amount  
 (3) value changes by a percentage  
 (4) not yet implemented  
 (5) use the specified callback  
 Ensures that all values are stored as 32-bit integers on all compilers.

### Description

The values in this enumeration are used when defining a [NetTypeBroadcastSchedule](#) for a [NetTypeField](#). The broadcast scheduler uses this enumeration to determine if the value of the field has changed significantly (given a threshold method). If the value has changed enough, then it is transmitted to all other clients.

### Notes

N/A

### Example

N/A

### See also

[NetTypeBroadcastSchedule](#), [NetRegisterObjectField](#)

# NetUpdateType

Enumeration: NetUpdate types.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_object.h	1.00	November 11, 2004

## Enumeration

```
typedef enum {  
    NetFullObjectUpdate = 0,           (0) Full NetObject update  
    NetFieldUpdate = 1,                (1) Field update of a NetObject  
    ExtraNetUpdateType = 0xffffffff  Ensures that all values are stored as 32-bit integers  
                                     on all compilers.  
} NetUpdateType;
```

## Description

Enumeration of all possible NetObject update types.

## Notes

N/A

## Example

N/A

## See also

[NetObjectFilterData](#)

## Chapter 3: Typedefs

This page intentionally left blank.

## HDME

Connection handle to the given connection.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_nettypes.h	2.07	November 11, 2004

### Syntax

```
typedef NetTypeConnectionInfo *HDME
```

### Description

Each time a new connection has been established, a new HDME will be exposed to the applicaiton. The application maintains a pointer to the HDME so that at later times it can be accessed when communicating on that connection.

### Notes

N/A

### Example

N/A

### See also

N/A

This page intentionally left blank.



## Chapter 4: Structures

This page intentionally left blank.

# NetAddress

Structure: Network address information.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_nettypes.h	1.18	November 11, 2004

## Structure

<code>typedef struct {</code>	
<code>    <a href="#">NetAddressType</a> <i>AddressType</i>;</code>	Defines the type of address stored in the address array below.
<code>    char <i>Address</i>[NET_MAX_NETADDRESS_LENGTH];</code>	Blob of address information that is formatted according the <i>AddressType</i> defined above.
<code>    unsigned int <i>Port</i>;</code>	Little endian 2-byte port representation associated with the address defined above or a 2-byte virtual port in the 2 high bytes and the 2-byte network port in the 2 low bytes.
<code>} NetAddress;</code>	

## Description

Defines the data structure used by the DME to specify connection addresses.

## Notes

N/A

## Example

N/A

## See also

[NetLANPeerDesc](#), [NetResolveAddrInParams](#), [NetSetNATServiceAddr\(\)](#), [NetPingNetAddress\(\)](#), [NetGetMyNetAddress\(\)](#), [NetAddressToStringAddress\(\)](#), [NetAddressList](#)

## NetAddressList

Structure: List of network address information.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_nettypes.h	1.18	November 11, 2004

### Structure

```
typedef struct {
    NetAddress
    aAddressList[NET_ADDRESS_LIST_COUNT];

} NetAddressList;
```

Array of network addresses that identify a machine on the network. (NET\_ADDRESS\_LIST\_COUNT is "2" as of this writing).  
See [NetAddress](#).

### Description

Defines the data structure used by the DME to specify a list of network addresses. Almost every address specified within the DME includes both an internal and external address. The SCEA NAT service is used to help clients determine their real external IP address and port number(s), such as what might be assigned by a broadband router in connection-sharing situations.

When peers are connecting to one another, this structure is used so that a connection attempt can be made on *both* the internal and external addresses. In certain situations, when players are connecting to the Medius Lobby Servers to find games online, but end up playing against each other on a local network (very typical in colleges and apartment "roommate" situations where one DSL or Cable modem is being shared), the peers will find each other, directly, via their broadband router or switch.

Otherwise, the machines need to know their proper external address so that the connection can be properly established even in situations where a broadband router is performing Network Address Translation (NAT) for peers behind the router. Without a NAT service, there would be no way to know how to connect to peers behind these NATs.

### Notes

N/A

### Example

N/A

### See also

[NetResolveAddrData](#), [NetIncomingClientInParams](#), [NetConnectionInfo](#)

## NetAudioDataCharacteristics

Structure: A DME type used for defining the characteristics of audio.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_nettypes.h	1.24	December 1, 2002

### Structure

```
typedef struct {
    int nChannelsIn;           Number of channels used for input.
    int nBitsPerSampleIn;     Number of bits per sample for input.
    int nSampleRateIn;        Sample rate of input in Hz.
    int nChannelsOut;         Number of channels used for output.
    int nBitsPerSampleOut;     Number of bits per sample for output.
    int nSampleRateOut;        Sample rate of output in Hz.
} NetAudioDataCharacteristics;
```

### Description

This information is used for defining the characteristics associated with audio data.

### Notes

N/A

### Example

N/A

### See also

[NetStreamMediaParams](#)

## NetBandwidthInfo

Structure: Client bandwidth usage information.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	dme.h	1.03	August 1, 2002

### Structure

```
typedef struct {
    NetClientMetric Sends;           Packets per second
    NetClientMetric SendBytes;       Bytes per second
    NetClientMetric Recvs;           Packets per second
    NetClientMetric RecvBytes;       Bytes per second
} NetBandwidthInfo;
```

### Description

Data type used to collect the given application's bandwidth number of sends and receives per second and the number of bytes used.

### Notes

N/A

### Example

N/A

### See also

[NetGetBandwidthInfo\(\)](#)

## NetBitMask

Structure: Generic DME bitmask.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_nettypes.h	2.09	November 11, 2004

### Structure

```
typedef struct {
```

```
    int bitmask[NET_MAX_BITMASK_ARRAY];
```

Array of 256 bits for use as an array of true/false values. These true/false values will be used by various functions to define a list of clients/tokens/objects/etc that should be operated upon.

```
    int base_id;
```

Defines the base index of the bitmask array. I.e. if *base\_id* is set to 256, then bit 0 is mapped to identifier 256 and bit 255 is mapped to identifier 511.

```
    int max_id;
```

Defines the max index that is used in the bitmask array. This field can be specified in order to minimize the number of loop iterations when processing the bitmask array. For instance, if only the first 32 of the 256 available bits are used, then *max\_id* should be set to 32.

```
} NetBitMask;
```

### Description

Defines the data structure used by the DME to specify a generic bitmask.

### Notes

N/A

### Example

N/A

### See also

[NetTokenListRequest\(\)](#), [NetTokenListRelease\(\)](#), [NetSetDefaultBitMask\(\)](#), [NetBitMaskIsSet\(\)](#), [NetBitMaskSet\(\)](#), [NetBitMaskUnSet\(\)](#), [NetClientList](#), [NetObjectFilterData](#)

### NetClientList

Structure: Bitmask of DME clients.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_nettypes.h	2.09	November 11, 2004

#### Structure

```
typedef struct {
    int TargetClient;

    NetBitMask ClientMask;

} NetClientList;
```

Target client definition. This field can be set in one of three ways:

NET\_SEND\_TO\_CLIENT\_MASK:  
Use the following client bitmask.

NET\_SEND\_TO\_ALL\_CLIENTS:  
Broadcast the message according to the client's broadcast flag setting.

0-(NET\_CLIENT\_LIMIT-1): Send to a single specified target.

Bitmask of destination clients. Note: This field is ignored if TargetClient is set to anything but NET\_SEND\_TO\_CLIENT\_MASK.

#### Description

This structure defines the information used by the DME to specify a bitmask of clients.

#### Notes

N/A

#### Example

N/A

#### See also

[NetGenerateClientList\(\)](#), [NetGenerateJoinedClientList\(\)](#), [NetSendMessageInParams](#), [NetSendMessageOutParams](#), [NetLatencyMetricsParams](#), [NetObjectFilterData](#)



## NetClientMetric

Structure: Client bandwidth information.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.03	August 1, 2001

### Structure

```
typedef struct {
    float AverageRate;           Arbitrary average rate information
    float MaxAverageRate;       Arbitrary max average rate.
} NetClientMetric;
```

### Description

Data type used to collect bandwidth metrics for the given application.

### Notes

Child structure for NetClientBandwidthInfo.

### Example

N/A

### See also

[NetBandwidthInfo](#)

## NetColorArray

Structure: DME type for data associated with a video image.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_nettypes.h	1.24	November 11, 2004

### Structure

```
typedef struct {
    int cbSize;           Total color array size.
    int lineSize;        Number of bytes taken by each image line.
    int xsize;           Horizontal pixel count.
    int ysize;           Vertical pixel count.
    char data[16];        Array of color data.
} NetColorArray;
```

### Description

This structure represents data associated with a video image taken from a camera.

### Notes

N/A

### Example

N/A

### See also

[NetStreamMediaVideoRecordData](#)

# NetCompletionData

Structure: DME type returned on non-blocking [NetJoin\(\)](#).

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_object.h	1.24	December 1, 2002

## Structure

```
typedef struct {  
  NetErrorCode Result;           NetErrorNone if successful  
  void *pUserData;               Pointer to UserData available when callback is  
                                triggered.  
} NetCompletionData;
```

## Description

This is a structure with the completion callback data. This information is returned to the caller in the completion callback and contains data to be used by the application.

## Notes

N/A

## Example

N/A

## See also

[NetTypeCompletionCallback](#)

## NetConnectInParams

Structure: Defines the input parameter for [NetConnect\(\)](#).

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_nettypes.h	1.18	November 11, 2004

### Structure

<b>typedef struct {</b>	
<b>int</b> <i>bDefaultSet</i> ;	Has this structure been filled with defaults (1 == true). This will be set by <a href="#">NetSetDefaultConnectParams()</a> and should not be modified by the caller
<b>int</b> <i>MaxClients</i> ;	Max clients to allow on connection
<a href="#">NetConnectionInfo</a> <i>ConnectionInfo</i> ;	Connection information
<a href="#">NetTypeConnectCallback</a> <i>pfLocalConnectCallback</i> ;	Called when the connection to the server is complete.
<b>void</b> <i>*pLocalConnectCallbackData</i> ;	Application defined pointer returned in local connect callback
<a href="#">NetTypeConnectCallback</a> <i>pfLocalDisconnectCallback</i> ;	Called when the connection to the server is lost.
<b>void</b> <i>*pLocalDisconnectCallbackData</i> ;	Application defined pointer returned in local disconnect callback
<a href="#">NetTypeClientConnectCallback</a> <i>pfRemoteClientConnectCallback</i> ;	Called when a remote client connects
<b>void</b> <i>*pRemoteClientConnectCallbackData</i> ;	Application defined pointer returned in remote connect callback
<a href="#">NetTypeClientConnectCallback</a> <i>pfRemoteClientDisconnectCallback</i> ;	Called when a remote client disconnects
<b>void</b> <i>*pRemoteClientDisconnectCallbackData</i> ;	Application defined pointer returned in remote disconnect callback
<a href="#">NetTypePeerToPeerHostChangeCallback</a> <i>pfPeerToPeerHostChangeCallback</i> ;	Called when peer-to-peer host changes
<b>void</b> <i>*pHostChangeCallbackData</i> ;	Application defined pointer returned in host change callback
<a href="#">NetTypeSystemStatusCallback</a> <i>pfSystemStatusCallback</i> ;	Called when one of the system status becomes ready
<b>void</b> <i>*pSystemStatusCallbackData</i> ;	Application defined pointer returned in system status callback
<a href="#">NetStreamMediaParams</a> <i>StreamMediaParams</i> ;	Options for using stream media with this connection
<b>int</b> <i>UserSpecified</i> ;	Information to be passed to all clients at connect time
<b>int</b> <i>AuxUDPBindPort</i> ;	If <a href="#">NetConnectionType</a> is set to <a href="#">NetConnectionTypeClientServerTCPAuxUDP</a> then this port number is bound
<a href="#">NetTokenParams</a> <i>TokenParams</i> ;	Enables DME NetTokens.
<b>int</b> <i>bUseTimeBase</i> ;	Set to 1 if use timebase. <a href="#">NetSetDefaultConnectParams</a> defaults this field to 1.
<b>unsigned int</b> <i>SendBufferSize</i> ;	Send buffer size to allocate for the connection. One buffer for TCP, Two for TCPAuxUDP, One or Two per peer for P2P

**unsigned int** *RecvBufferSize*;

Receive buffer size to allocate for the connection.  
One buffer for TCP, Two for TCPAuxUDP, One or  
Two per peer for P2P

**} NetConnectInParams;**

### Description

Defines the input parameter for [NetConnect\(\)](#).

### Notes

N/A

### Example

N/A

### See also

[NetSetDefaultConnectParams\(\)](#), [NetConnect\(\)](#)

## NetConnectionInfo

Structure: DME connection information.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_nettypes.h	1.18	November 11, 2004

### Structure

```
typedef struct {
```

```
    NetConnectionType Type;
```

Specifies the type of connection represented (client-server TCP, peer-to-peer UDP, client-server AuxUDP, etc.).

```
    NetAddressList AddressList;
```

Array of addresses available to be used to establish this connection. Both an internal and an external address are commonly specified except in certain situations such as when first connecting to the Medius Authentication Server, where only the external address is valid.

```
    int WorldID;
```

Every connection also requires a DME World ID. The DME can (and does) operate several unique worlds with the context of one connected socket. So, for example, when you connect to mygameserver.scea.com, port 50000, There may well be dozens of unique "worlds" running there. For the most part, this value is handled internally, or explicitly set to one (1) for connections to the Medius Authentication Server, Medius Lobby Server, peer-to-peer game hosts, etc.

```
    RSA_KEY ServerKey;
```

The server/host's public RSA encryption key. Again, this is typically handled internally, but is specified manually for your first connection to the universe (such as when connecting to the Medius Authentication Server or to the MUIS).

```
    char aSessionKey[NET_SESSION_KEY_LEN];
```

For a first connection to the universe, this should be zeroed. Subsequent connections, however, will commonly set this value internally to indicate the client's session key identifier on the server. This "key" allows the servers to reserve slots for clients, such as when calling MediusJoinChannel and/or MediusJoinGame. The Medius Universe Manager will send your session key to the server to which you are about to connect and ask that server to reserve a seat/slot for you. The client then has, at most, thirty (30) seconds to connect to that server and present this key.

```
    char aAccessKey[NET_ACCESS_KEY_LEN];
```

The size of this array, as of this writing, is 17 bytes. A key returned by completing login and requesting a server to play on via a Medius Server connection.

```
} NetConnectionInfo;
```

### Description

A complete connection information structure that is used by the DME to connect to servers, peer-to-peer hosts and peers.

### Notes

N/A

**Example**

N/A

**See also**

[NetConnectInParams](#)

NetConnectionStatus

Structure: Contains connection status information.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	dme.h	1.31	December 8, 2003

Structure

```

typedef struct {
    NetConnectStatus myConnectStatus;

    int nValidClientCount;

    int nConnectedClientCount;

} NetConnectionStatus;

```

Status of my connection. See [NetGetConnectStatus\(\)](#).

Number of valid clients. See [NetGetValidClientCount\(\)](#).

Total number of invalid and valid clients on the connection. Invalid clients are those that the application has not received a connect callback about.

Description

This structure contains information about the client's connection status.

Notes

N/A

Example

N/A

See also

[NetGetConnectionStatus\(\)](#)



## NetConnectOutParams

Structure: Output parameters for [NetConnect\(\)](#).

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	dme.h	1.18	June 18, 2004

### Structure

```
typedef struct {
```

```
    int blsSet;
```

```
    NetErrorCode ErrorCode;
```

```
    HDME ConnectionHandle;
```

```
} NetConnectOutParams;
```

Has this structure been filled by [NetConnect\(\)](#); (1 == true).

Result of call the same as returned by the function call itself.

(Return value) The connection identifier for this connection.

### Description

This type defines the output parameters for [NetConnect\(\)](#). Set when [NetConnect\(\)](#) returns.

### Notes

N/A

### Example

N/A

### See also

[NetConnect\(\)](#)

## NetData

Structure: DME buffer data information.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	June 18, 2004

### Structure

```
typedef struct {  
    int nSize;                Size of the data in bytes.  
    void *pData;              Pointer to buffer containing data.  
} NetData;
```

### Description

Structure used throughout the DME that contains a pointer and size to a given buffer.

### Notes

N/A

### Example

N/A

### See also

[NetLANFindExchangeCallbackInArgs](#), [NetLANFindExchangeCallbackOutArgs](#),  
[NetLANFindCallbackDataArgs](#), [NetLANFindInParams](#)

# NetDisconnectParams

Structure: Input parameters for [NetDisconnect\(\)](#).

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_nettypes.h	2.09	November 11, 2004

## Structure

```
typedef struct {  
    HDME ConnectionHandle;           DME connection handle that should be  
                                     disconnected  
  
    NetDisconnectReason Reason;      Reason for disconnecting (optional parameter)  
  
    NetTypeConnectCallback pfLocalDisconnectCallback; Application defined callback function to call when  
                                     the NetDisconnect\(\) process has completed.  
  
    void *pUserData;                Pointer to UserData available when callback is  
                                     triggered.  
}  
  
} NetDisconnectParams;
```

## Description

Structure used to pass params to [NetDisconnect\(\)](#).

## Notes

N/A

## Example

N/A

## See also

[NetSetDefaultDisconnectParams\(\)](#), [NetDisconnect\(\)](#)

## NetDmeVersion

Structure that contains the version in both binary and ASCII zero-terminated string.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Structure

```
typedef struct {  
    unsigned int nVersion;           Version as binary.  
    char szVersion[NET_VERSION_STRING_LENGTH]; Version as ASCII.  
} NetDmeVersion;
```

### Description

Structure that contains the version in both binary and ASCII zero-terminated string

### Notes

N/A

### Example

N/A

### See also

[NetLANPeerDesc](#)

## NetEnableLanMessagingInParams

Structure: Enable LAN messaging.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_lanmessage.h	2.09	November 11, 2004

### Structure

```
typedef struct {
    NET\_LAN\_TEXT\_MESSAGE\_CALLBACK
    pLanTextMessageCallback;
    void *pTextMessageCallbackUserData;

    NET\_LAN\_RAW\_MESSAGE\_CALLBACK
    pLanRawMessageCallback;
    void *pRawMessageCallbackUserData;
} NetEnableLanMessagingInParams;
```

Application defined callback function for processing text messages.

User data passed to callback when text message is received.

Application defined callback function for processing raw messages.

User data passed to raw data callback when raw message received.

### Description

Input structure to enable LAN message via [NetEnableLANMessaging\(\)](#).

### Notes

N/A

### Example

N/A

### See also

[NetEnableLANMessaging\(\)](#), [NetLANSetDefaultEnableMessagingInParams\(\)](#)

## NetErrorThresholdCallbackData

Structure: Data returned to application-defined error threshold callback.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.24	December 1, 2002

### Structure

```
typedef struct {
    int ObjectIndex;           Object to be updated.
    int FieldIndex;           Field to be updated.
    int FieldSize;           Size of the field to update.
    int FieldCount;          Array size (1 == not an array).
    void *pCurrentData;        Pointer to current field data.
    void *pLastUpdateData;     Pointer to the last data sent.
} NetErrorThresholdCallbackData;
```

### Description

This structure with the error threshold is the callback data. This information is returned to the caller in the error threshold callback for the object field and contains data to be used by the application

### Notes

N/A

### Example

N/A

### See also

[NetTypeErrorThresholdCallback](#)

## NetHostPeerToPeerInParams

Structure: Input parameter for [NetHostPeerToPeer\(\)](#).

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	dme.h	1.18	December 8, 2003

### Structure

```
typedef struct {
```

```
int bDefaultSet;
```

```
NetTypeConnectCallback pfLocalConnectCallback;
```

```
void *pLocalConnectCallbackData;
```

```
NetTypeConnectCallback pfLocalDisconnectCallback;
```

```
void *pLocalDisconnectCallbackData;
```

```
NetTypeClientConnectCallback
```

```
pfRemoteClientConnectCallback;
```

```
void *pRemoteClientConnectCallbackData;
```

```
NetTypeClientConnectCallback
```

```
pfRemoteClientDisconnectCallback;
```

```
void *pRemoteClientDisconnectCallbackData;
```

```
NetTypeSystemStatusCallback pfSystemStatusCallback;
```

```
void *pSystemStatusCallbackData;
```

```
unsigned int MaxClients;
```

```
NetStreamMediaParams StreamMediaParams;
```

```
int UserSpecified;
```

```
int bEnabDisconnectFwd;
```

```
NetTokenParams TokenParams;
```

```
int bUseTimeBase;
```

```
unsigned int SendBufferSize;
```

```
unsigned int RecvBufferSize;
```

```
} NetHostPeerToPeerInParams;
```

Has this structure been filled with defaults (1 == true) this will be set by [NetSetDefaultConnectParams\(\)](#) and should not be modified by the caller.

Called when the connection to own server is complete.

Application defined pointer returned in local connect callback.

Called when the connection to the server is lost.

Application defined pointer returned in local disconnect callback.

Called when a remote client connects.

Application defined pointer returned in local connect callback.

Called when a remote client disconnects.

Application defined pointer returned in local connect callback.

Called when one of the system status becomes ready.

Application defined pointer returned in system status callback.

Maximum clients allowed in this game.

Options for using stream media with this connection.

Information to be passed to all clients at connect time.

Use Disconnect forwarding.

Options for using token with this connection.

Set to 1 if use timebase.

Send buffer size to allocate for the connection. One buffer for TCP, two for TCPAuxUDP, one or two per peer for P2P.

Receive buffer size to allocate for the connection. One buffer for TCP, two for TCPAuxUDP, one or two per peer for P2P.

### Description

This type defines the input parameter for [NetHostPeerToPeer](#).

**Notes**

N/A

**Example**

N/A

**See also**

[NetSetDefaultHostPeerToPeerParams\(\)](#), [NetHostPeerToPeer\(\)](#)



# NetHostPeerToPeerOutParams

Structure: Output parameter for [NetHostPeerToPeer\(\)](#).

Link to file	Include file	Introduced	Last modified
<a href="#">librtbasePS2.a</a>	dme.h	1.18	June 18, 2004

## Structure

```
typedef struct {
    int blsSet;
    NetErrorCode ErrorCode;
    HDME ConnectionHandle;
} NetHostPeerToPeerOutParams;
```

Has this structure been filled by [NetHostPeerToPeer\(\)](#) (1 == true).  
Result of call the same as returned by the function call itself.  
(Return value) The connection identifier for this hosted connection.

## Description

This type defines the output parameter for NetHostPeerToPeer. Set when NetHostPeerToPeer returns.

## Notes

N/A

## Example

N/A

## See also

[NetHostPeerToPeer\(\)](#)

## NetIncomingClientInParams

Structure: Input parameter for [NetIncomingClient\(\)](#).

Link to file	Include file	Introduced	Last modified
<a href="#">librtbasePS2.a</a>	dme.h	2.09	November 11, 2004

### Structure

```
typedef struct {  
    int bDefaultSet;  
  
    NetAddressList IncomingAddressList;  
    RSA\_KEY pubKey;  
} NetIncomingClientInParams;
```

Has this structure been filled with defaults (1 == true), this will be set by [NetSetDefaultIncomingClientParams\(\)](#) and should not be modified by the caller.

Address list of incoming client.

The incoming client's public key.

### Description

Structure with input parameter information for [NetIncomingClient\(\)](#).

### Notes

N/A

### Example

N/A

### See also

[NetSetDefaultIncomingClientParams\(\)](#), [NetIncomingClient\(\)](#)

## NetIncomingClientOutParams

Structure: Output parameter for NetIncomingClient.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	dme.h	2.09	November 11, 2004

### Structure

```
typedef struct {
```

```
    int blsSet;
```

```
    NetErrorCode ErrorCode;
```

```
} NetIncomingClientOutParams;
```

Has [NetIncomingClient\(\)](#) been called? If so blsSet will be set to 1.

Error code set if any problems occurred during this call.

### Description

Structure with output parameter information for [NetIncomingClient\(\)](#). Set when [NetIncomingClient\(\)](#) returns.

### Notes

N/A

### Example

N/A

### See also

[NetIncomingClient\(\)](#)

## NetInitializeParams

Structure: Input parameters for [NetInitialize\(\)](#).

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.18	June 18, 2004

### Structure

```
typedef struct {
```

```
int bDefaultSet;
```

```
NetLocalizationParams Localization;
```

```
int ApplicationID;
```

```
unsigned char ApplicationName
```

```
[NET_MAX_APPLICATION_NAME_SIZE];
```

```
const RSA\_KEYPAIR *pLocalKeyPair;
```

```
const RSA\_KEYPAIR *pApplicationKeyPair;
```

```
NetConnectivityType ConnectivityType;
```

```
int UdpBindPort;
```

Has NetSetDefaultInitializeParams been set? If so bDefaultSet will be 1.

Localization parameters when the peer is in LAN mode.

SCE-RT assigned ApplicationID. Each title will have their own uniquely generated ApplicationID. ApplicationID == 0 is for general purpose. Calling [KM\\_GetSoftwareID\(\)](#) will return the SCE-RT assigned ApplicationID from the security library generated by SCE-RT. Use the security library SCERTSAMPLE\_\*.a(.lib) until your title has a SCE-RT generated security library.

Application name. Based on NetInitializeParams::LocalizationParams.

The key pair used for public/private key encryption between client and server. This is the machine specific RSA local key pair. The application should call KM\_GenerateRSAKeyPair() to generate it upon startup. This key is unique from machine to machine and is recommended that you generate a unique key every reboot.

Title specific software key. The application should call KM\_GetSoftwareKeyPair() to get the software key pair from the SCE-RT security library.

Connectivity type that will be used for this networking session. This setting allows the client code to take different optimal codepaths depending on network availability. This field defaults to NetConnectivityNone. This field must be set to a valid connectivity type otherwise NetInitialize will fail.

UDP bindport to be used for all P2P and StreamMedia communications. This field will be set to a random number between 6000-7000 by the call to [NetSetDefaultInitializeParams\(\)](#). These are the SCE-RT recommended settings for Internet gameplay. If the application wishes to run in a LAN environment then ConnectivityType above should be set to NetConnectivityLAN and UdpBindPort should be set to DEFAULT\_LAN\_UDP\_BINDPORT. If this field setting is not within MIN\_UDP\_BINDPORT and MAX\_UDP\_BINDPORT then [NetInitialize\(\)](#) will return NetErrorBadPort.

`int UPnPMemoryCeiling;`

Memory ceiling for UPnP queries. If this is set to zero then UPnP functionality will be disabled. Setting to 512 KB is typical. Until further notice, SCE-RT recommends that this field be set to zero.

`int bEnableLANBroadcastComms;`

Enable broadcast communications for LAN gameplay. This field is ignored unless ConnectivityType is set to NetConnectivityLAN.

[NetTypeSystemMessageCallback](#)

`pfSystemMessageCallback;`

Medius system message callback. Triggered when a Medius server sends a client a system message.

`void *pSystemMessageCallbackData;`

User defined data to be passed along with the system message callback.

`} NetInitializeInParams;`

### Description

Structure with input parameter information for [NetInitialize\(\)](#).

### Notes

N/A

### Example

N/A

### See also

[NetSetDefaultInitializeParams\(\)](#), [NetInitialize\(\)](#)

## NetInitializeOutParams

Structure: Output parameters for [NetInitialize\(\)](#).

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	dme.h	1.18	March 13, 2002

### Structure

```
typedef struct {  
    int blsSet;                                Has NetInitialize\(\) been called.  
    NetErrorCode ErrorCode;                   Error code set if any problems occurred during this  
                                              call.  
    char szVersion[NET_VERSION_STRING_LENGTH]; DME client version.  
} NetInitializeOutParams;
```

### Description

This type is used for storing output parameters for [NetInitialize\(\)](#). Set when [NetInitialize\(\)](#) returns.

### Notes

N/A

### Example

N/A

### See also

[NetInitialize\(\)](#)

## NetJoinInParams

Structure: Input parameters for [NetJoin\(\)](#).

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_object.h	1.24	June 18, 2004

### Structure

```
typedef struct {
```

```
    int bDefaultSet;
```

```
    HDME ConnectionHandle;
```

```
    char szClientName  
    [NET_MAX_CLIENT_NAME_LENGTH];
```

```
    int NetObjectCount;
```

```
    int DataStreamCount;
```

```
    NetOwnershipStatus SessionMasterStatus;
```

```
    NetTypeCompletionCallback pfLocalJoinCallback;
```

```
    void *pJoinCallbackData;
```

```
    NetTypeRemoteClientEventCallback
```

```
    pfRemoteClientEventCallback;
```

```
    void *pRemoteClientEventCallbackData;
```

```
    NetTypeSMChangeCallback pfSMChangeCallback;
```

```
    void *pSMChangeCallbackData;
```

```
    NetTypeOwnershipUpdateCallback
```

```
    pfOwnershipUpdateCallback;
```

```
    void *pOwnershipUpdateCallbackData;
```

```
    NetTypeOwnershipRequestCallback
```

```
    pfOwnershipRequestCallback;
```

```
    void *pOwnershipRequestCallbackData;
```

```
} NetJoinInParams;
```

Has this structure been filled with defaults (1 == true). This will be set by [NetSetDefaultJoinParams\(\)](#) and should not be modified by the caller

Connection handle of the given connection.

Client name. Note: Client name should be copied into this structure

Number of objects that will be used

Number of data streams that will be used

Session master selection process

Pointer to callback for join complete

Application defined pointer returned in join callback

Pointer to callback for another client joining or leaving

Application defined pointer returned in remote joins/leaves callback

Pointer to callback for session master change

Application defined pointer returned in session master change callback

Pointer to callback for ownership updates

Application defined pointer returned in ownership update callback

Pointer to callback for handling object ownership requests

Application defined pointer returned in ownership request callback

### Description

Structure with input parameter information for [NetJoin\(\)](#)

### Notes

N/A

### Example

N/A

### See also

[NetSetDefaultJoinParams\(\)](#), [NetJoin\(\)](#)

## NetJoinOutParams

Structure: Output parameters set by [NetJoin\(\)](#).

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	rt_object.h	1.24	December 1, 2002

### Structure

```
typedef struct {
```

```
    int blsSet;
```

```
    NetErrorCode ErrorCode;
```

```
} NetJoinOutParams;
```

Has this structure been filled by [NetJoin\(\)](#) (1 == true)

Error code set if any problems occurred during this call.

### Description

Structure with output parameter information for [NetJoin\(\)](#).

### Notes

N/A

### Example

N/A

### See also

[NetJoin\(\)](#)



# NetLANFindCallbackDataArgs

Structure: NetLANFindCallback information.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_lanfind.h	2.09	November 11, 2004

## Structure

```
typedef struct {  
    short nNumClients;           Current number of peers in the game at the time of  
                                the request.  
  
    short nMaxNumClients;       Maximum number of peers allowed in the game.  
  
    NetSessionType SessionType; Session type of the game.  
  
    NetLANPeerDesc PeerDesc;    Peer description information about the host of the  
                                game.  
  
    NetData Details;           Arbitrary data returned by the host of the game.  
  
    void *pUserData;           Pointer to UserData available when callback is  
                                triggered.  
  
} NetLANFindCallbackDataArgs;
```

## Description

This structure contains game specific information.

## Notes

N/A

## Example

N/A

## See also

[NetLANFindCallback](#)

# NetLANFindExchangeCallbackInArgs

Structure: Input parameters for NetLANFindExchangeCallback.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_lanfind.h	2.09	November 11, 2004

## Structure

```
typedef struct {  
    HDME ConnectionHandle;           Connection handle of the given connection.  
    NetSessionType SessionType;      The session type of the requesting client.  
    NetLANPeerDesc PeerDesc;         Descriptive information about the requesting peer.  
    NetData Details;                Arbitrary data that the requesting peer may have sent.  
  
    void *pUserData;                Pointer to UserData available when callback is triggered.  
}  
NetLANFindExchangeCallbackInArgs;
```

## Description

This structure is passed to a host when a LANFind() request is received by a host. The information contains environment of the client sending the request.

## Notes

N/A

## Example

N/A

## See also

[NetLANFindExchangeCallback](#)

# NetLANFindExchangeCallbackOutArgs

Output parameters for NetLANFindExchangeCallback.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_lanfind.h	2.09	November 11, 2004

## Structure

```
typedef struct {
    int bRespondToSender;
    NetData Details;
} NetLANFindExchangeCallbackOutArgs;
```

If FALSE, LANFind will not respond to the sender.  
(Useful to act as if invisible.)

Max: MAX\_LANFIND\_DETAILS\_SIZE

## Description

This structure is passed to a host when a LANFind request is received by a host. The host can fill in the information which will be replied back to the calling client.

## Notes

N/A

## Example

N/A

## See also

[NetLANFindExchangeCallback](#)

NetLANFindInParams

Structure: Input parameters for [NetLANFind\(\)](#).

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_lanfind.h	2.09	November 11, 2004

Structure

typedef struct {	
int bDefaultSet;	Call <a href="#">NetSetDefaultLANFindParams()</a> to set this field to true.
<a href="#">NetSessionType</a> SessionType;	NetSessionTypeGame - finds all peer-to-peer hosts on the LAN NetSessionTypePeer - finds all peers on the LAN NetSessionTypeIntegratedServer - finds all Servers (Client/Server games) on the LAN NET_LANFIND_FILTER_xxx Max: MAX_LANFIND_DETAILS_SIZE
unsigned int Filter;	UDP port to find games on.
<a href="#">NetData</a> Details;	Callback if lan game found.
unsigned int UDPPort;	Pointer to UserData available when callback is triggered.
<a href="#">NetLANFindCallback</a> pfnLANFindCallback;	
void *pUserData;	
} NetLANFindInParams;	

Description

This structure contains input parameters for [NetFindGame\(\)](#).

Notes

N/A

Example

N/A

See also

[NetLANFind\(\)](#), [NetSetDefaultLANFindParams\(\)](#)

# NetLANPeerDesc

Structure: LAN peer information.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	dme.h	2.09	November 11, 2004

## Structure

<b>typedef struct {</b>	
<b>NetDmeVersion</b> <i>DmeVersion</i> ;	DME Client version used by this peer.
<b>EnumNetPlatformID</b> <i>NetPlatformID</i> ;	Platform type used by this peer (eg. PS2 or PSP).
<b>int</b> <i>ApplicationID</i> ;	Application ID assigned to this peer.
<b>NetAddress</b> <i>PeerAddress</i> ;	Local LAN address information currently assigned to this peer.
<b>NetLocalizationParams</b> <i>Localization</i> ;	Language localization information regarding this peer.
<b>unsigned char</b> <i>ApplicationName</i> [NET_MAX_APPLICATION_NAME_SIZE];	Short application name currently in use.
<b>unsigned char</b> <i>UserName</i> [MAX_USER_NAME];	Short username used by this peer.
<b>} NetLANPeerDesc;</b>	

## Description

Contains peer information to be used by the DME's [NetLANFind\(\)](#) features.

## Notes

N/A

## Example

N/A

## See also

[NetLANFindExchangeCallbackInArgs](#), [NetLANFindCallbackDataArgs](#), [NetLanTextMessageCallbackParams](#), [NetLanRawMessageCallbackParams](#)

NetLanRawMessageCallbackParams

Structure: Parameters for NET\_LAN\_RAW\_MESSAGE\_CALLBACK.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_lanmessage.h	2.09	November 11, 2001

Structure

```

typedef struct {
    NetLANPeerDesc PeerDescription;
    int nMsgSize;
    const void *pData;
    void *pUserData;
} NetLanRawMessageCallbackParams;

```

Peer description of the person sending the raw message.  
Size in bytes of the raw message.  
Pointer to buffer containing raw message.  
Pointer to UserData available when callback is triggered.

Description

Structure used for passing parameters in the NET\_LAN\_RAW\_MESSAGE\_CALLBACK.

Notes

N/A

Example

N/A

See also

[NET\\_LAN\\_RAW\\_MESSAGE\\_CALLBACK](#)

## NetLANSendRawMessageInParams

Structure: Input parameters for [NetLANSendRawMessage\(\)](#).

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	<a href="#">rt_lanmessage.h</a>	2.09	November 11, 2004

### Structure

```
typedef struct {
    NetAddress Address;           Destination address.
    int nMsgSize;                 Size of the binary message.
    const void *pMsg;             Pointer to the message.
} NetLANSendRawMessageInParams;
```

### Description

This structure contains input parameters for [NetLANSendRawMessage\(\)](#).

### Notes

N/A

### Example

N/A

### See also

[NetLANSetDefaultSendRawMessageInParams\(\)](#), [NetLANSendRawMessage\(\)](#)

## NetLANSendTextMessageInParams

Structure: Input parameters for NetLanSendTextMessage().

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_lanmessage.h	2.09	November 11, 2004

### Structure

```
typedef struct {
    NetAddress Address;           Network Address of machine to whom raw
                                message is sent.
    int nTextMsgSize;           Size of text message.
    const char *pTextMsg;       Pointer to buffer containing text message.
} NetLANSendTextMessageInParams;
```

### Description

This structure contains input parameters for NetLanSendTextMessage().

### Notes

N/A

### Example

N/A

### See also

[NetLANSetDefaultSendTextMessageInParams\(\)](#), [NetLANSendTextMessage\(\)](#)



# NetLanTextMessageCallbackParams

Structure: Paramater passed to NET\_LAN\_TEXT\_MESSAGE\_CALLBACK.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_lanmessage.h	2.09	November 11, 2004

## Structure

```
typedef struct {
    NetLANPeerDesc PeerDescription;
    int nMsgSize;
    const char *pTextMsg;
    void *pUserData;
} NetLanTextMessageCallbackParams;
```

Peer description of client sending message.

Size in bytes of Message.

Pointer to buffer containing message.

Pointer to UserData available when callback is triggered.

## Description

NetLanTextMessageCallbackParams is a structure used for passing parameters with NET\_LAN\_TEXT\_MESSAGE\_CALLBACK.

## Notes

N/A

## Example

N/A

## See also

[NET\\_LAN\\_TEXT\\_MESSAGE\\_CALLBACK](#)

# NetLatencyMetricsDataArgs

Strcuture: Information returned to the caller in the NetTypeLatencyMetricsCallback.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_nettypes.h	2.09	November 11, 2004

## Structure

```
typedef struct {
    HDME ConnectionHandle;           DME connection handle that kicks off this callback
    int TargetClientIndex;           Client of interest for the latency metrics.
    NetLatencyMetricsInfo LatencyMetricsInfo;  Client latency metrics
    NetErrorCode ErrorCode;         Error return code
    void *pUserData;               Pointer to UserData available when callback is
                                   triggered.
} NetLatencyMetricsDataArgs;
```

## Description

This information is returned to the caller in the NetTypeLatencyMetricsCallback.

## Notes

N/A

## Example

N/A

## See also

[NetTypeLatencyMetricsCallback](#)

## NetLatencyMetricsInfo

Structure: Overall client latency metrics.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	rt_nettypes.h	2.09	November 11, 2004

### Structure

```
typedef struct {
    unsigned int LatencyMin;           Shortest recorded latency.
    unsigned int LatencyMax;          Longest recorded latency.
    unsigned int LatencyAvg;          Average recorded latency.
} NetLatencyMetricsInfo;
```

### Description

As a field of [NetLatencyMetricsDataArgs](#), NetLatencyMetricsInfo provides overall client latency metrics. For a client-server game, the latency is defined as the round-trip time between that client and the server. For a peer-to-peer game, the latency is defined as the round-trip time between that peer and the querying peer.

### Notes

N/A

### Example

N/A

### See also

[NetLatencyMetricsDataArgs](#)

# NetLatencyMetricsParams

Structure: Input parameter for [NetGetLatencyMetrics\(\)](#).

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_nettypes.h	2.09	November 11, 2004

## Structure

```
typedef struct {  
    HDME ConnectionHandle;  
    NetClientList TargetClientList;  
  
    NetTypeLatencyMetricsCallback  
    pfLatencyMetricsCallback;  
    void *pUserData;  
  
} NetLatencyMetricsParams;
```

Connection handle of the given connection.

List of clients that are of interest for their latency metrics. The list defaults TargetClient = a single specified target. If TargetClient = NET\_ALL\_CLIENTS, Dme will return data on all clients/peers, in peer-to-peer case, the latency to the requesting peer itself will be the latency to the host, and host will have zero latency to itself.

Application defined callback function to call when the NetGetLatencyMetrics process has completed.

Pointer to UserData available when callback is triggered.

## Description

Structure used to pass parameters to the NetLatencyMetrics function.

## Notes

N/A

## Example

N/A

## See also

[NetSetDefaultLatencyMetricsParams\(\)](#), [NetGetLatencyMetrics\(\)](#)

## NetLocalizationParams

Structure: Localization information.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.31	December 8, 2003

### Structure

```
typedef struct {
    NetCharacterEncodingType CharacterEncodingType;    Set Character Encoding.
                                                         (ISO-8859-1) Latin chacters.
                                                         (UTF-8) Unicode characters.
    NetLanguageType LanguageType;                    Language specified.
} NetLocalizationParams;
```

### Description

Field of [NetInitializeInParams](#) that declares the clients current Character Encoding and Language information.

### Notes

N/A

### Example

N/A

### See also

[NetLANPeerDesc\(\)](#), [NetInitializeInParams\(\)](#), [NetGetLocalizationSettings\(\)](#)

## NetMemoryCallbackParams

Structure: Memory callback information.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Structure

```
typedef struct {
    NetMallocCallback pfMallocCallback;           Pointer to malloc override.
    NetReallocCallback pfReallocCallback;         Pointer to realloc override.
    NetFreeCallback pfFreeCallback;               Pointer to free override.
} NetMemoryCallbackParams;
```

### Description

Structure that contains function pointers to memory allocations and deallocation routines.

### Notes

N/A

### Example

N/A

### See also

[NetSetDefaultMemoryCallbackParams\(\)](#), [NetRegisterMemoryCallbacks\(\)](#)

# NetObjectFilterData

Structure: Structured parameter list for a NetTypeObjectFilterCallback.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_object.h	2.09	November 11, 2004

## Structure

<b>typedef struct {</b>	
<b>NetClientList</b> <i>ClientList</i> ;	List of clients that should receive this object/field update. The list defaults TargetClient = NET_SEND_TO_ALL_CLIENTS, and ClientMask will contain the list of all NetJoined clients. The application is free to modify the ClientList according to their needs.
<b>int</b> <i>ObjectIndex</i> ;	Index of the object that this update is relevant to.
<b>NetBitMask</b> <i>FieldsUpdated</i> ;	Mask that denotes the object fields that are being transmitted in this update. NOTE: Changing this field will not affect the update message that will be transmitted across the wire, hence the constness of the parameter. If control of individual field updates are desired, then this is best achieved by using the ErrorThresholdCallback functionality of the DME.
<b>NetUpdateType</b> <i>UpdateType</i> ;	Netupdate type in this call back
<b>void</b> * <i>pUserData</i> ;	Pointer to UserData available when callback is triggered.
<b>} NetObjectFilterData;</b>	

## Description

This structure allows the user to modify the object filtering with a custom callback.

## Notes

Only the ClientList member can be modified in the callback.

## Example

N/A

## See also

[NetTypeObjectFilterCallback](#)

### NetPeerToPeerHostChangeData

Structure: A DME type returned on the peer-to-peer host change callback.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_nettypes.h	1.24	November 11, 2004

#### Structure

```
typedef struct {  
    HDME ConnectionHandle;           Connection handle of the given connection.  
    int HostClientIndex;           Client index of the new host.  
    void *pUserData;           Pointer to UserData available when callback is  
                               triggered.  
} NetPeerToPeerHostChangeData;
```

#### Description

This information is returned to the caller in the peer-to-peer host change callback is triggered and contains data to be used by the application.

#### Notes

The peer-to-peer host change callback [NetTypePeerToPeerHostChangeCallback](#) is set in the [NetConnectInParams](#) structure.

#### Example

N/A

#### See also

[NetTypePeerToPeerHostChangeCallback](#)



# NetRegisterObjectFilterInParams

Structure: Input parameters for [NetRegisterObjectFilter\(\)](#).

Link to file	Include file	Introduced	Last modified
<a href="#">librtbasePS2.a</a>	rt_object.h	2.09	November 11, 2004

## Structure

```
typedef struct {  
    NetTypeObjectFilterCallback ObjectFilterCallBack;    Filter callback for restricting/modifying object/field  
                                                         update destination lists  
  
    void *pUserData;    Pointer to UserData available when callback is  
                       triggered.  
  
} NetRegisterObjectFilterInParams;
```

## Description

This function registers an object filter function. The client will execute this function when the object has changed to determine whether the remote client should receive the update for object's new state.

## Notes

N/A

## Example

N/A

## See also

[NetSetDefaultRegisterObjectFilterParams\(\)](#), [NetRegisterObjectFilter\(\)](#)

**NetRegisterObjectFilterOutParams**

Structure: Output parameters for [NetRegisterObjectFilter\(\)](#).

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_object.h	2.09	November 11, 2004

**Structure**

```

typedef struct {
    int ObjectFilterType;
    NetErrorCode ErrorCode;
} NetRegisterObjectFilterOutParams;

```

Will be filled out with the filter type index that is associated with this callback  
 Error code returned by the function call.

**Description**

Parameter structure for output parameters to [NetRegisterObjectFilter\(\)](#).

**Notes**

N/A

**Example**

N/A

**See also**

[NetRegisterObjectFilter\(\)](#)

# NetRemoteClientEventData

Structure: Remote client event data.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_object.h	1.24	December 1, 2002

## Structure

```
typedef struct {
    NetClientEventType EventType;           Event that triggered the callback
    HDME ConnectionHandle;                 Connection handle of the given connection.
    int ClientIndex;                       Client index of remote client that caused the event.
    void *pUserData;                       Pointer to UserData available when callback is
                                           triggered.
} NetRemoteClientEventData;
```

## Description

This structure contains the remote client event callback data. This information is returned to the caller in the remote client event callback and contains data to be used by the application.

## Notes

N/A

## Example

N/A

## See also

[NetTypeRemoteClientEventCallback](#)

**NetResolveAddrData**

Structure: Network Address Resolution (NAT) information.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	dme.h	2.09	November 11, 2004

**Structure**

```

typedef struct {
    int blsSet;
    NetErrorCode ErrorCode;
    NetAddressList AddressList;
} NetResolveAddrData;

```

Set to true if this structure been filled by [NetResolveAddr\(\)](#).  
Result of the resolution.  
Address list for this client.

**Description**

Structure with address resolution data. This information is returned to the caller in the address resolution callback.

**Notes**

N/A

**Example**

N/A

**See also**

[NetTypeResolveAddrCallback](#)

# NetResolveAddrInParams

Structure: Input parameter for [NetResolveAddr\(\)](#).

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	dme.h	2.09	November 11, 2004

## Structure

```
typedef struct {  
    int bDefaultSet;  
  
    NetTypeResolveAddrCallback pfResolveAddrCallback;  
    NetAddress NatServiceAddress;  
} NetResolveAddrInParams;
```

Set to true when this structure has been filled with defaults. Defaults are set by [NetSetDefaultResolveAddrParams\(\)](#).  
Called when the resolution is complete.  
NAT service address.

## Description

Structure with input parameters for NetResolveAddr.

## Notes

N/A

## Example

N/A

## See also

[NetSetDefaultResolveAddrParams\(\)](#), [NetResolveAddr\(\)](#)

## NetResolveAddrOutParams

Structure: Output parameter for [NetResolveAddr\(\)](#).

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Structure

```
typedef struct {
    int blsSet;
    NetErrorCode ErrorCode;
} NetResolveAddrOutParams;
```

Set to true if this structure has been filled by [NetResolveAddr\(\)](#).  
Result of call.

### Description

Structure with output parameter information for [NetResolveAddr\(\)](#). Set when [NetResolveAddr\(\)](#) returns.

### Notes

N/A

### Example

N/A

### See also

[NetResolveAddr\(\)](#)

## NetRGBArray

Structure: Represents the RGB data of a video image.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_nettypes.h	1.24	November 11, 2004

### Structure

```
typedef struct {
    int cbSize;
    int lineSize;
    int xsize;
    int ysize;
    unsigned int data[16];
} NetRGBArray;
```

Size of the structure (can vary in size depending upon the data).

The number of bytes taken by each image line.

Image size in pixels.

Image size in pixels.

Raw video data.

### Description

This structure represents the RGB data of a video image.

### Notes

N/A

### Example

N/A

### See also

[NetStreamMediaVideoPlayData](#)

NetSendMessageInParams

Structure: Input parameters for [NetSendAppMessage\(\)](#).

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_nettypes.h	2.09	November 11, 2004

Structure

```

typedef struct {
    HDME ConnectionHandle;           Connection handle of the given connection.
    char TransportFlags;             Transport flags for the message
                                     (NET_DELIVERY_CRITICAL, etc).
    int MessageType;                Message type returned from NetRegisterMessage\(\).
    NetClientList DestClient;        Specifies the intended destination(s)
    int MessageLength;              Length of message data in bytes.
    unsigned char *MessageData;     Pointer to message data body.
} NetSendMessageInParams;
    
```

Description

Input parameters for [NetSendAppMessage\(\)](#).

Notes

N/A

Example

N/A

See also

[NetSendAppMessage\(\)](#)



## NetSendMessageOutParams

Structure: Output parameter for [NetSendAppMessage\(\)](#).

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_nettypes.h	2.09	November 11, 2004

### Structure

```
typedef struct {
    NetErrorCode ErrorCode;
    NetClientList QueuedClient;
```

Result of this call.

If [NetSendAppMessage\(\)](#) returns NetErrorNone then the following client list will be a duplicate of the one passed in the NetSendAppMessageInParams. If there was an error sending the message to any of the clients, this client list will denote for which clients the message was successfully queued for sending.

```
} NetSendMessageOutParams;
```

### Description

Structure containing output parameters for [NetSendAppMessage\(\)](#).

### Notes

N/A

### Example

N/A

### See also

[NetSendAppMessage\(\)](#)

## NetSMChangeData

Structure: Session Master migration information.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.24	June 18, 2004

### Structure

```
typedef struct {
    HDME ConnectionHandle;           Connection handle of the given connection.
    int SMClientIndex;              Client index of the new Session Master.
    void *pUserData;               Pointer to UserData available when callback is
                                  triggered.
} NetSMChangeData;
```

### Description

This structure contains Session Master migration information. If the game supports Session Master migration, then if the old Session Master has exited the game either gracefully or ungracefully then the DME network engine will automatically arbitrate which client will become the new Session Master based on the remaining connected clients.

### Notes

N/A

### Example

N/A

### See also

[NetTypeSMChangeCallback](#), [NetJoinInParams](#)

# NetStreamMediaAudioPlayData

Structure: A DME type used for the stream media play callback.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_nettypes.h	1.24	November 11, 2004

## Structure

```
typedef struct {
    NetStreamMediaAudioType AudioType;
    int ClientIndex;

    const unsigned char *pBuffer;
    unsigned int BufSize;
    unsigned int BytesRead;

    void *pUserData;
} NetStreamMediaAudioPlayData;
```

## Description

Structure with the stream media audio play callback. This information is returned to the caller in the stream media audio play callback and contains data to be used by the application.

## Notes

N/A

## Example

N/A

## See also

[NetTypeStreamMediaAudioPlayCallback](#)

# NetStreamMediaAudioRecordData

Structure: A DME type used for the stream media record callback.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_nettypes.h	1.24	November 11, 2004

## Structure

```
typedef struct {
    NetStreamMediaAudioType AudioType;
    unsigned char *pBuffer;
    unsigned int BufSize;
    unsigned int BytesStored;
    char TransportFlags;
    void *pUserData;
} NetStreamMediaAudioRecordData;
```

Must be set to audio type that was used to record audio.

Location to copy recorded data.

Number of bytes than can be stored in buffer.

Must be set to number of bytes stored in buffer (cannot be greater than BufSize).

NET\_DELIVERY\_CRITICAL, etc

Pointer to UserData available when callback is triggered.

## Description

Structure with the stream media audio record callback information. This information is returned to the caller in the stream media audio record callback and must be filled out by the application.

## Notes

The TransportFlags parameter is only used if AudioType is set to NetStreamMediaAudioTypeRAW or NetStreamMediaAudioTypeCUSTOM.

## Example

N/A

## See also

[NetTypeStreamMediaAudioRecordCallback](#)

# NetStreamMediaChannelInfo

Structure: A DME type for stream media information.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_media.h	1.24	June 18, 2004

## Structure

```
typedef struct {
    HDME ConnectionHandle;           Connection handle of the given connection.
    unsigned int ChannelNum;         Channel number.
    unsigned int ClientCount;        Number of clients currently joined to this channel.
} NetStreamMediaChannelInfo;
```

## Description

This structure contains stream media channel information.

## Notes

N/A

## Example

N/A

## See also

[NetStreamMediaGetChannelInfo\(\)](#)

## NetStreamMediaChannelStateData

Structure: A DME type for stream media state.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_media.h	1.24	December 1, 2002

### Structure

```
typedef struct {
    int bCanRecord;
} NetStreamMediaChannelStateData;
```

Set to true when we can start recording on the current channel.

### Description

This structure contains channel state data.

### Notes

N/A

### Example

N/A

### See also

[NetStreamMediaGetCurrentChannelState\(\)](#)

# NetStreamMediaClientInfo

Structure: A DME type for stream media state for a client.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_media.h	1.24	June 18, 2004

## Structure

```
typedef struct {  
    HDME ConnectionHandle;           Connection handle of the given connection.  
    int ClientIndex;                Index of client.  
    int bJoinedToChannel;           Set to true if the client is joined to any stream  
                                   media channel.  
    unsigned int ChannelNum;        Channel number if client is joined.  
} NetStreamMediaClientInfo;
```

## Description

This structure contains stream media client information.

## Notes

N/A

## Example

N/A

## See also

[NetStreamMediaGetClientInfo\(\)](#)

## NetStreamMediaCustomVideoPlayData

Structure: Stream media custom video play information.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_nettypes.h	1.24	June 18, 2004

### Structure

```
typedef struct {
    int ClientIndex;
    unsigned char *pBuffer;
    unsigned int nBytesAvailable;
    unsigned int nBytesProcessed;
    void *pUserData;
} NetStreamMediaCustomVideoPlayData;
```

Client index of client from whom this video originated

Pointer to buffer with video data

Number of bytes in data buffer

Number of bytes processed from buffer

Pointer to UserData available when callback is triggered.

### Description

This structure contains the stream media custom video play callback. This information is returned to the caller in the stream media video play callback and contains data to be used by the application.

### Notes

N/A

### Example

N/A

### See also

[NetTypeStreamMediaCustomVideoPlayCallback](#)



## NetStreamMediaCustomVideoRecordData

Structure: Stream media custom video record information.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_nettypes.h	1.24	June 18, 2004

### Structure

```
typedef struct {
    unsigned char *pBuffer;           Location to store video data
    unsigned int nBufferSize;         Size of buffer for video data
    unsigned int nBytesStored;         Number of bytes copied into buffer
    char TransportFlags;               NET_DELIVERY_CRITICAL, etc
    void *pUserData;                  Pointer to UserData available when callback is
                                     triggered.
} NetStreamMediaCustomVideoRecordData;
```

### Description

This structure contains the stream media custom video record callback. This information is returned to the caller in the stream custom media video record callback and must be filled out by the application.

### Notes

N/A

### Example

N/A

### See also

[NetTypeStreamMediaCustomVideoRecordCallback](#)

## NetStreamMedialgnoreData

Structure: A DME type for ignored stream media.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_media.h	1.31	June 18, 2004

### Structure

```
typedef struct {
    int bDefaultsSet;
    HDME ConnectionHandle;
    int ClientIndex;
    int bIgnore;
} NetStreamMedialgnoreData;
```

Set to true if filled in by [NetStreamMediaSetDefaultIgnoreParams\(\)](#).  
 Connection handle of the given connection.  
 Index of client.  
 1 to ignore, 0 to stop ignoring.

### Description

This structure contains information for ignoring stream media from a given user.

### Notes

N/A

### Example

N/A

### See also

[NetStreamMediaSetDefaultIgnoreParams\(\)](#), [NetStreamMediaSetIgnoreState\(\)](#)

## NetStreamMediaParams

Structure: A DME type used for enabling stream media when clients connect.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_nettypes.h	1.24	November 11, 2004

### Structure

<b>typedef struct {</b>	
<b>int</b> <i>bDefaultSet</i> ;	Set to true (by <a href="#">NetSetDefaultStreamMediaParams()</a> ) if this structure has been filled with default values.
<b>int</b> <i>bUseStreamMedia</i> ;	Set to true to enable stream media (on given connection).
<b>unsigned int</b> <i>NumChannels</i> ;	Number of stream media channels to be used (for example, one for each team).
<b>unsigned int</b> <i>MaxIncomingAudioStreams</i> ;	Max number of audio streams that can be received at one time (default 1).
<b>unsigned int</b> <i>RecordNoDataTimeout</i> ;	Amount of time to wait before releasing ownership of an audio stream automatically (milliseconds).
<b>unsigned int</b> <i>SendBufferSize</i> ;	Per peer send buffer size to allocate. Note: Only used for Client/Server style connections. Buffers are shared for peer-to-peer games.
<b>unsigned int</b> <i>RecvBufferSize</i> ;	Per peer receive buffer size to allocate. Note: Only used for Client/Server style connections. Buffers are shared for peer-to-peer games.
 <a href="#">NetStreamMediaGridType</a> <i>GridType</i> ;	Type of grid to be used (Relay or Direct).
<a href="#">NetAudioDataCharacteristics</a> <i>AudioDataCharacteristics</i> ;	Information about the audio being streamed.
<a href="#">NetTypeStreamMediaAudioRecordCallback</a> <i>pfAudioRecordCallback</i> ;	Function called that requests recorded audio data from application.
<b>void</b> <i>*pAudioRecordCallbackData</i> ;	Application defined pointer returned in audio record callback.
 <a href="#">NetTypeStreamMediaAudioPlayCallback</a> <i>pfAudioPlayCallback</i> ;	Function called that feeds audio data to application.
<b>void</b> <i>*pAudioPlayCallbackData</i> ;	Application defined pointer returned in audio play callback.
 <a href="#">NetVideoDataCharacteristics</a> <i>VideoDataCharacteristics</i> ;	Information about the video being streamed.
<a href="#">NetTypeStreamMediaVideoRecordCallback</a> <i>pfVideoRecordCallback</i> ;	Function called that requests recorded video data from application.
<b>void</b> <i>*pVideoRecordCallbackData</i> ;	Application defined pointer returned in video record callback.
 <a href="#">NetTypeStreamMediaVideoPlayCallback</a> <i>pfVideoPlayCallback</i> ;	Function called that feeds video data to application.
<b>void</b> <i>*pVideoPlayCallbackData</i> ;	Application defined pointer returned in video play callback.
 <a href="#">NetTypeStreamMediaCustomVideoRecordCallback</a> <i>pfCustomVideoRecordCallback</i> ;	Function called that requests custom video data from application.
<b>void</b> <i>*pCustomVideoRecordCallbackData</i> ;	Application defined pointer returned in video record callback.
 <a href="#">NetTypeStreamMediaCustomVideoPlayCallback</a> <i>pfCustomVideoPlayCallback</i> ;	Function called that feeds custom video data to the application.

```
void *pCustomVideoPlayCallbackData;
```

Application defined pointer returned in video record callback.

```
} NetStreamMediaParams;
```

### Description

This structure contains information for enabling stream media when clients connect on a given connection.

### Notes

When setting the video record/play callbacks, set either the custom callbacks or the regular video record/play callbacks. If both custom and non-custom callbacks are set, the DME client will prefer the non-custom callbacks over the custom callbacks, and the custom callbacks will never be called.

### Example

N/A

### See also

[NetHostPeerToPeerInParams](#), [NetConnectInParams](#), [NetSetDefaultStreamMediaParams\(\)](#)

# NetStreamMediaVideoPlayData

Structure: Stream media video play information.

Link to file	Include file	Introduced	Last modified
<a href="#">librtbasePS2.a</a>	<a href="#">rt_nettypes.h</a>	1.24	November 11, 2004

## Structure

```
typedef struct {
    int ClientIndex;
    const NetRGBArray *pRGBArray;
    int bDataProcessed;
    void *pUserData;
} NetStreamMediaVideoPlayData;
```

Client index of client from whom this video originated

Pointer to image data

This is set to true if application processed (i.e., used) the video data. Note: This MUST be set by the application

Pointer to UserData available when callback is triggered.

## Description

This structure contains the stream media video play callback. This information is returned to the caller in the stream media video play callback and contains data to be used by the application.

## Notes

N/A

## Example

N/A

## See also

[NetTypeStreamMediaVideoPlayCallback](#)

# NetStreamMediaVideoRecordData

Structure: Stream media video record information.

Link to file	Include file	Introduced	Last modified
<a href="#">librtbasePS2.a</a>	<a href="#">rt_nettypes.h</a>	1.24	November 11, 2004

## Structure

```
typedef struct {
    NetColorArray *pY;           Location to store LUMA data.
    NetColorArray *pCr;          Location to store chroma RED data.
    NetColorArray *pCb;          Location to store chroma BLUE data.
    int bDataStored;             This is set to true if the application stored video
                                data to be streamed. Note: This MUST be set by
                                the application
    void *pUserData;             Pointer to UserData available when callback is
                                triggered.
} NetStreamMediaVideoRecordData;
```

## Description

This structure contains the stream media video record callback. This information is returned to the caller in the stream media video record callback and must be filled out by the application.

## Notes

N/A

## Example

N/A

## See also

[NetTypeStreamMediaVideoRecordCallback](#)

# NetSystemStatusData

Structure: System status information.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_nettypes.h	2.09	November 11, 2004

## Structure

```
typedef struct {  
    HDME ConnectionHandle;           Connection handle of the given connection.  
    NetSystemStatus Status;          Current system status.  
    NetErrorCode ErrorCode;          Result of call.  
    void *pUserData;                 Pointer to UserData available when callback is  
                                     triggered.  
} NetSystemStatusData;
```

## Description

This structure contains status data. This information is returned to the caller in the status callback.

## Notes

N/A

## Example

N/A

## See also

[NetTypeSystemStatusCallback](#)

## NetTokenOwnershipNotifyData

Structure: Token ownership notification information.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_nettypes.h	2.09	November 11, 2004

### Structure

```
typedef struct {
    HDME ConnectionHandle;           Connection handle of the given connection.
    int TokenID;                     Token ID.
    int OwnerClientIndex;            Owner Client ID.
    NetErrorCode ErrorCode;          Result of call.
    void *pUserData;                Pointer to UserData available when callback is
                                   triggered.
} NetTokenOwnershipNotifyData;
```

### Description

This structure contains token data. This information is returned to the caller in the token ownership notify callback.

### Notes

N/A

### Example

N/A

### See also

[NetTypeTokenOwnershipNotifyCallback](#)



# NetTokenParams

Structure: Enables net tokens on a connection by connection basis.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_nettypes.h	2.09	November 11, 2004

## Structure

```
typedef struct {  
    NetTypeTokenOwnershipNotifyCallback      Called when token ownership changes  
    pfTokenOwnershipNotifyCallback;  
    void *pTokenOwnershipNotifyCallbackData; Application defined pointer returned in token  
                                           ownership change callback  
    int bUseToken;                          Set to true to enable net tokens when a connection  
                                           is made.  
} NetTokenParams;
```

## Description

This structure contains parameters for NetToken.

## Notes

Application must set bUseToken to 1 at [NetConnect\(\)](#) to use NetToken API

## Example

N/A

## See also

[NetHostPeerToPeerInParams](#), [NetConnectInParams](#)

## NetTypeBroadcastSchedule

Structure: Defines how a network object's field is updated.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_object.h	1.00	December 8, 2003

### Structure

```
typedef struct {
    unsigned int MinUpdateInterval;
    NetThresholdMethod ErrorThresholdType;

    float ErrorThresholdMagnitude;
    NetTypeErrorThresholdCallback pfThresholdCallback;

    char TransportFlags;
} NetTypeBroadcastSchedule;
```

Maximum rate limit for sending updates.

Which NetObject field change threshold method is being used.

Minimum delta for broadcast.

Error threshold callback function to be executed. (Only used if ErrorThresholdType == ThresholdCallback). This allows the client application greater control when a network object's field is to be propagated across the network.

NET\_DELIVERY\_CRITICAL, etc.

### Description

This structure is used to determine when an object field update is sent.

### Notes

N/A

### Example

N/A

### See also

[NetTypeField](#), [NetObjectField\(\)](#)

# NetTypeClient

Structure: Describes a joined network client.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_object.h	1.00	June 12, 2001

## Structure

```
typedef struct {
    unsigned int ConnectTime;           Time of connection.
    unsigned int ClientObjectIndex;     Network object of client (avatar).
    int NetObjectBufferStart;           Index into net object array.
    int NetObjectBufferCount;           Number of objects.
    unsigned int NetDataStreamStart;    Index into NetDataStreamList.
    unsigned int NetDataStreamCount;    Number of streams.
    char Name[NET_MAX_CLIENT_NAME_LENGTH]; Client name.
} NetTypeClient;
```

## Description

This structure describes a joined network client. A client may be active (creating NetObjects) or passive (creating no NetObjects).

## Notes

N/A

## Example

N/A

## See also

[NetGetClient\(\)](#)

## NetTypeClientConnectCallbackData

Structure: Data passed to the NetTypeClientConnectCallback.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_nettypes.h	1.31	November 11, 2004

### Structure

```
typedef struct {
    HDME ConnectionHandle;           Connection handle of the given connection.
    int ClientIndex;                 Index of remote client connecting or disconnecting.
    const char *ClientIpString;      IP address string of remote client.
    NetClientStatus ClientStatus;   The client's current status.
    int UserSpecified;              Data specified by the incoming client at
                                   NetConnect() time (the UserSpecified field of
                                   NetConnectInParams is treated as a blob of data).

    void *pUserData;                Pointer to UserData available when callback is
                                   triggered.
} NetTypeClientConnectCallbackData;
```

### Description

The data type used by the NetTypeClientConnectCallback prototype.

### Notes

N/A

### Example

N/A

### See also

[NetTypeClientConnectCallback](#)

# NetTypeConnectCallbackData

Structure: Connect callback information.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_nettypes.h	1.31	June 18, 2004

## Structure

```
typedef struct {
    HDME ConnectionHandle;
    NetConnectStatus ConnectStatus;

    NetConnectFailureReason FailureReason;
    NetErrorCode ErrorCode;
    void *pUserData;
} NetTypeConnectCallbackData;
```

Connection handle of the given connection.

Status of connection (ConnectStatusOpen or ConnectStatusFailed).

Reason of failure if ConnectStatusFailed.

Result of call.

Pointer to UserData available when callback is triggered.

## Description

The datatype is used by the NetTypeConnectCallback prototype.

## Notes

N/A

## Example

N/A

## See also

[NetTypeConnectCallback](#)

## NetTypeDataStream

Structure: Data stream information used for streaming audio/video.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	June 12, 2001

### Structure

<b>typedef struct {</b>	
<b>unsigned char</b> <i>Status</i> ;	Status is either free or active.
<b>unsigned char</b> <i>CircularBuffer</i> ;	If true, then a circular buffer is being used.
<b>unsigned char</b> <i>RemoteBuffer</i> ;	If true, then create a buffer on remote client.
<b>unsigned char</b> <i>StreamType</i> ;	User defined (audio/video etc.)
<b>unsigned char</b> <i>BufferComplete</i> ;	If true, the buffer is complete.
<b>unsigned char</b> <i>SendToAll</i> ;	If true, then this data stream is sent to all clients.
<b>int</b> <i>TargetClientIndex</i> ;	Client index for which this data stream is to be sent to (assuming SendToAll is set to false).
 <b>int</b> <i>OwnerClientIndex</i> ;	 The client sending the data should set this to their ClientIndex.
 <b>char</b> * <i>BufferStart</i> ;	 Pointer to start of the stream data.
<b>char</b> * <i>BufferEnd</i> ;	Pointer to the end of the stream data.
<b>char</b> * <i>ReadPtr</i> ;	Pointer to buffer to read from.
<b>char</b> * <i>WritePtr</i> ;	Pointer to buffer to which to write.
<b>int</b> <i>DataRate</i> ;	The send number of bytes per second.
<b>unsigned short</b> <i>MinPacketSize</i> ;	(For aggregation) Minimum packet size.
<b>unsigned short</b> <i>MaxPacketSize</i> ;	(for granularity) Maximum packet size.
<b>unsigned int</b> <i>TimeOfLastUpdate</i> ;	Time of last update (for broadcast scheduling).
<b>} NetTypeDataStream;</b>	

### Description

This structure is for streaming audio/video data or sending large buffers (polygon meshes, texture maps etc.) in a bandwidth controlled way. Non-buffered data streams must be processed during callbacks.

### Notes

N/A

### Example

N/A

### See also

[NetGetDataStream\(\)](#)

## NetTypeDoubleVector2

Structure: A DME-defined type.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	June 12, 2001

### Structure

```
typedef struct {
    double x;           X component of the vector.
    double y;           Y component of the vector.
} NetTypeDoubleVector2;
```

### Description

Structure defining a double precision two-dimensional vector.

### Notes

These compound data structures are directly supported by the broadcast scheduler.

### Example

N/A

### See also

N/A

## NetTypeDoubleVector3

Structure: A DME-defined type.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	June 12, 2001

### Structure

```
typedef struct {  
    double x;           X component of the vector.  
    double y;           Y component of the vector.  
    double z;           Z component of the vector.  
} NetTypeDoubleVector3;
```

### Description

Structure defining a double precision three-dimensional vector.

### Notes

These compound data structures are directly supported by the broadcast scheduler.

### Example

N/A

### See also

N/A



# NetTypeField

Structure: Atomic data unit for network object updates.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_object.h	1.00	June 12, 2001

## Structure

```
typedef struct {
    int Offset;
    int ElementSize;
    int ElementCount;
    int Type;
    NetTypeBroadcastSchedule UpdateSchedule;
} NetTypeField;
```

The offset, in bytes, of the field from the base of structure.

Size of field in bytes.

The number of elements in array. Set this field to "1" if an array is not being used.

The field type (i.e. char, short, float etc).

A structure containing the broadcast schedule for the field (i.e. if and when this field's information should be propagated out onto the network).

## Description

This structure defines atomic data unit for network object updates. An array of NetTypeFields defines a NetTypeStructure.

## Notes

N/A

## Example

N/A

## See also

[NetTypeStructure](#)

## NetTypeFloatVector2

Structure: A DME-defined type.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	June 12, 2001

### Structure

```
typedef struct {
    float x;                X component of the vector.
    float y;                Y component of the vector.
} NetTypeFloatVector2;
```

### Description

Structure defining a single precision two-dimensional vector.

### Notes

These compound data structures are directly supported by the broadcast scheduler.

### Example

N/A

### See also

N/A

## NetTypeFloatVector3

Structure: A DME-defined type.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	June 12, 2001

### Structure

```
typedef struct {
    float x;           X component of the vector.
    float y;           Y component of the vector.
    float z;           Z component of the vector.
} NetTypeFloatVector3;
```

### Description

Structure defining a single precision three dimensional vector.

### Notes

These compound data structures are directly supported by the broadcast scheduler.

### Example

N/A

### See also

N/A

## NetTypeIntVector2

Structure: A DME-defined type.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	June 12, 2001

### Structure

```
typedef struct {
    int x;                X component of the vector.
    int y;                Y component of the vector.
} NetTypeIntVector2;
```

### Description

Structure defining a two-dimensional integer vector.

### Notes

These compound data structures are directly supported by the broadcast scheduler.

### Example

N/A

### See also

N/A

## NetTypeIntVector3

Structure: A DME-defined type.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	June 12, 2001

### Structure

```
typedef struct {
    int x;           X component of the vector.
    int y;           Y component of the vector.
    int z;           Z component of the vector.
} NetTypeIntVector3;
```

### Description

Structure defining a three dimensional integer vector.

### Notes

These compound data structures are directly supported by the broadcast scheduler.

### Example

N/A

### See also

N/A

## NetTypeLookupParams

Structure: Used to issue a DNS request.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Structure

```
typedef struct {
    char szHostName[NET_MAX_HOSTNAME_LENGTH];      Hostname to resolve.
    char szServerIP[NET_MAX_NETADDRESS_LENGTH];    IP address of DNS server to use for query.
    NetTypeLookupCallback pfLookupResponse;        Pointer to function.
} NetTypeLookupParams;
```

### Description

Structure that contains information for issues a DNS lookup.

### Notes

N/A

### Example

N/A

### See also

[NetSetDefaultLookupParams\(\)](#), [NetGetHostByName\(\)](#)

# NetTypeLookupResponse

Structure: Contains DNS lookup information.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	dme.h	2.09	November 11, 2004

## Structure

```
typedef struct {  
    char aszIPAddresses  
    [NET_MAX_RESPONSES][NET_MAX_NETADDRESS_LENGTH];  
    unsigned int nIPAddresses;  
  
    NetErrorCode ErrorCode;  
  
} NetTypeLookupResponse;
```

Network address information.

Number of valid addresses in the array.

Error code for the response.

## Description

This structure contains DNS lookup information in response to a call to [NetGetHostByName\(\)](#).

## Notes

N/A

## Example

N/A

## See also

[NetTypeLookupCallback](#)

## NetTypeObject

Structure: Defines a network object (NetObject).

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	June 12, 2001

### Structure

<b>typedef struct {</b>	
<b>unsigned char</b> <i>StructureIndex</i> ;	Application-defined (object type).
<b>unsigned char</b> <i>FilterType</i> ;	Index for filter callback.
<b>unsigned char</b> <i>LifespanType</i> ;	Lifespan type timeout/client/session/permanent.
<b>unsigned char</b> <i>LatencyCritical</i> ;	Latency critical enabled true/false.
<b>int</b> <i>OwnerClientIndex</i> ;	Specifies the current owner of this object (-1 = shared).
<b>int</b> <i>CreatorClientIndex</i> ;	Specifies which client created this object.
<b>char</b> <i>Name</i> [NET_MAX_OBJECT_NAME_LENGTH];	Specifies the 'name' of this object.
<b>char</b> * <i>CurrentObjectData</i> ;	Actual current state data.
<b>char</b> * <i>LastGlobalObjectDataUpdate</i> ;	Used by the broadcast scheduler for deltas (when to propagate data on the network).
<b>unsigned int</b> <i>HiFieldChangeSet</i> ;	The high-order bits of field change set.
<b>unsigned int</b> <i>LoFieldChangeSet</i> ;	The low-order bits of field change set.
<b>unsigned int</b> <i>TimeOfExpiration</i> ;	Declares the time when this object's lifespan will expire.
<b>unsigned int</b> <i>MaxUpdateInterval</i> ;	Minimum update rate (heartbeat).
<b>unsigned int</b> <i>TimeOfLastGlobalUpdate</i> ;	Used by the broadcast scheduler.
<b>unsigned int</b> * <i>TimeOfLastClientUpdate</i> ;	Reserved by DME.
<b>unsigned int</b> * <i>TimeOfLastClientFieldUpdate</i> ;	Reserved by DME.
<b>void</b> * <i>LocalUserData</i> ;	Can be used to associate a game object with a network object or non-propagated data.
<b>} NetTypeObject;</b>	

### Description

This structure is the primary data unit for the representation and propagation of world state data.

### Notes

N/A

### Example

N/A

### See also

[NetGetObject\(\)](#)



## NetTypeOwnershipRequestData

Structure: Data filled out by a NetTypeOwnershipRequestCallback.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	2.07	June 14, 2004

### Structure

```
typedef struct {
    HDME ConnectionHandle;           Connection handle of the given connection.
    int ClientIndex;                 Client requesting ownership.
    int ObjectIndex;                 Index of object requested.
    void *pUserData;                 Pointer to UserData available when callback is
                                    triggered.
} NetTypeOwnershipRequestData;
```

### Description

A NetTypeOwnershipRequestCallback fills out this data structure type.

### Notes

N/A

### Example

N/A

### See also

[NetTypeOwnershipRequestCallback](#)

# NetTypeOwnershipUpdateData

Structure: Data filled out by a NetTypeOwnershipUpdateCallback.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_object.h	2.07	June 18, 2004

## Structure

```
typedef struct {
    HDME ConnectionHandle;           Connection handle of the given connection.
    int ClientIndex;                 New/current owner of object.
    int ObjectIndex;                 Relevant object index.
    NetObjectOwnershipType state;     Ownership state change.
    void *pUserData;                Pointer to UserData available when callback is
                                    triggered.
} NetTypeOwnershipUpdateData;
```

## Description

A NetTypeOwnershipUpdateCallback fills out this data structure type.

## Notes

N/A

## Example

N/A

## See also

[NetTypeOwnershipUpdateCallback](#)

## NetTypeShortVector2

Structure: A DME-defined type.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	June 12, 2001

### Structure

```
typedef struct {
    short x;                X component of the vector.
    short y;                Y component of the vector.
} NetTypeShortVector2;
```

### Description

Structure defining a two-dimensional vector of shorts.

### Notes

These compound data structures are directly supported by the broadcast scheduler.

### Example

N/A

### See also

N/A

## NetTypeShortVector3

Structure: A DME-defined type.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	June 12, 2001

### Structure

```
typedef struct {
    short x;           X component of the vector.
    short y;           Y component of the vector.
    short z;           Z component of the vector.
} NetTypeShortVector3;
```

### Description

Structure defining a three-dimensional vector of shorts.

### Notes

These compound data structures are directly supported by the broadcast scheduler.

### Example

N/A

### See also

N/A

# NetTypeStructure

Structure: Defines a net structure type that can be used by a network object (NetObject).

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_object.h	1.00	June 12, 2001

## Structure

```
typedef struct {
    int TotalSize;
    int FieldCount;
    char Name[NET_MAX_STRUCT_NAME_LENGTH];
    NetTypeField *ChildFieldType
    [NET_MAX_FIELDS_PER_STRUCTURE];
    int ChildFieldOffset
    [NET_MAX_FIELDS_PER_STRUCTURE];
} NetTypeStructure;
```

## Description

Defines a structure that can be used with a network object as one if its fields. This structure will consist of an array of fields each with their own broadcast scheduler. This structure with its list of NetTypeFields can be nested recursively to create compound structures. This helps a programmer register network objects when they may not know up-front changing requirements of a given data-object.

## Notes

N/A

## Example

N/A

## See also

N/A

# NetTypeSystemMessageData

Structure: System Message DME type.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	dme.h	1.31	June 18, 2004

## Structure

```
typedef struct {
    HDME ConnectionHandle;
    unsigned char Severity;
    NetCharacterEncodingType eEncodingType;
    NetLanguageType eLanguageType;
    unsigned char bEndOfMessage;

    unsigned short nMessageLength;
    const unsigned char *pMessage;
    void *pUserData;
} NetTypeSystemMessageData;
```

Connection handle of the given connection.

Message severity. The server defines this field.

Character Encoding Type.

Language Type.

Set to 1 (true) when the last message has been sent.

Length of the current System Message.

Pointer to the System Message.

Pointer to UserData available when callback is triggered.

## Description

Data structure type filled out by a NetTypeSystemMessageCallback.

## Notes

N/A

## Example

N/A

## See also

[NetTypeSystemMessageCallback](#)

# NetUpdateConnErrors

Structure: Information about errors within [NetUpdate\(\)](#).

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	dme.h	1.18	June 18, 2004

## Structure

```
typedef struct {
    unsigned int SizeofNetUpdateErrors;
    NetUpdateError aErrors[NET_MAX_CONNECTIONS];
    NetErrorCode UDPError;
} NetUpdateConnErrors;
```

Users of [NetGetNetUpdateErrors](#) must set this field to the size of [NetUpdateErrors](#).

Error list for per connection index basis. It is set to [NetErrorNone](#) if there was no error.

It is set for any UDP message error.

## Description

This type is used when retrieving the error that occurred on a particular connection within the context of returning from [NetUpdate\(\)](#).

## Notes

N/A

## Example

N/A

## See also

[NetGetNetUpdateErrors\(\)](#)

NetUpdateError

Structure: [NetUpdate\(\)](#) error information.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	dme.h	2.07	June 18, 2004

Structure

```

typedef struct {
    HDME ConnectionHandle;           Connection handle of the given connection.
    NetErrorCode NetError;           Error for this connection is set to NetErrorNone if
                                     there is no error.
} NetUpdateError;

```

Description

Structure that is filled in by [NetGetNetUpdateErrors\(\)](#).

Notes

N/A

Example

N/A

See also

[NetUpdateConnErrors](#)



## NetVideoDataCharacteristics

Structure: A DME type for video data characteristics.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_nettypes.h	1.24	November 11, 2004

### Structure

```
typedef struct {  
    int XSize;           Number of pixels in X dimension of image.  
    int YSize;           Number of pixels in Y dimension of image.  
} NetVideoDataCharacteristics;
```

### Description

This structure contains characteristics of streamed video.

### Notes

N/A

### Example

N/A

### See also

[NetStreamMediaParams](#)

## RSA\_KEY

RSA public key encryption key.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtcryptPS2.a	rt_crypttypes.h	1.00	June 12, 2001

### Structure

```
typedef struct {  
    unsigned int key[RSA_SIZE_DWORD];  
} RSA_KEY;
```

Key used in RSA public key encryption.

### Description

RSA public key encryption key.

### Notes

N/A

### Example

N/A

### See also

KM\_GetMediusPublicKey(), KM\_GetSoftwareKeyPair(), [KM\\_GetSoftwareKey\(\)](#), KM\_GenerateRSAKeyPair()

# RSA\_KEYPAIR

RSA public key encryption key pair.

Link to file	Include file	Introduced	Last modified
librcryptPS2.a	rt_crypttypes.h	1.00	June 12, 2001

## Structure

```
typedef struct {  
    RSA_KEY publicKey;           Public key used in RSA public key encryption.  
    RSA_KEY privateKey;         Private key used in RSA public key encryption.  
} RSA_KEYPAIR;
```

## Description

RSA public key encryption key pair.

## Notes

Used internally.

## Example

N/A

## See also

N/A

This page intentionally left blank.

## Chapter 5: Callback Functions

This page intentionally left blank.

## NET\_LAN\_RAW\_MESSAGE\_CALLBACK

Callback: LAN raw message received callback.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_lanmessage.h	2.09	November 11, 2004

### Syntax

```
typedef void (*NET_LAN_RAW_MESSAGE_CALLBACK)(
    NetLanRawMessageCallbackParams *pParams);
```

Pointer to LAN raw message information.

### Description

This typedef is for a callback that occurs when a LAN raw message is received.

### Notes

N/A

### Return value

None.

### Example

N/A

### See also

[NetLanRawMessageCallbackParams](#), [NetEnableLanMessagingInParams](#)

## NET\_LAN\_TEXT\_MESSAGE\_CALLBACK

This typedef is for a callback that occurs when a LAN text message is received.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_lanmessage.h	2.09	November 11, 2004

### Syntax

```
typedef void (*NET_LAN_TEXT_MESSAGE_CALLBACK)(  
    NetLanTextMessageCallbackParams *pParams);
```

Pointer to LAN text message information.

### Description

This typedef is for a callback that occurs when a LAN text message is received.

### Notes

N/A

### Return value

None.

### Example

N/A

### See also

[NetLanTextMessageCallbackParams](#), [NetEnableLanMessagingInParams](#)



## NetFreeCallback

Used to define a custom version of free.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	June 12, 2001

### Syntax

```
typedef void (*NetFreeCallback)(
    void *BlockPtr);
```

Pointer to block to be freed.

### Description

This prototype is used by an application to define a custom version of the Standard C library function free. This user defined version of free will be used for releasing dynamically allocated memory in the DME.

### Notes

N/A

### Return value

None.

### Example

N/A

### See also

[NetMemoryCallbackParams](#)

## NetLANFindCallback

Callback: Prototype for a LAN Find callback.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_lanfind.h	2.09	November 11, 2004

### Syntax

```
typedef void (*NetLANFindCallback)(
    const NetLANFindCallbackDataArgs *pArgs);
```

Pointer to LANFind information.

### Description

This prototype is used by an application to define a callback that is called by the return with the LANFind information.

### Notes

N/A

### Return value

None.

### Example

N/A

### See also

[NetLANFindCallbackDataArgs](#), [NetLANFindInParams](#)

## NetLANFindExchangeCallback

Callback: Prototype for a LAN Find Exchange callback.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_lanfind.h	2.09	November 11, 2004

### Syntax

```
typedef void (*NetLANFindExchangeCallback)(
    const NetLANFindExchangeCallbackInArgs *pInArgs,    LANFind exchange input information.
    NetLANFindExchangeCallbackOutArgs *pOutArgs);       LANFind exchange output information.
```

### Description

This prototype is used by an application to define a callback that is called by the DME to exchange LANFind data.

### Notes

N/A

### Return value

None.

### Example

N/A

### See also

N/A

## NetMallocCallback

Used to define a custom version of malloc.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	June 12, 2001

### Syntax

```
typedef void * (*NetMallocCallback)(
    size_t size);
```

Size of memory block being allocated

### Description

This prototype is used by an application to define a custom version of the Standard C library function malloc. This user defined version of malloc will be used for memory allocation by the DME.

### Notes

N/A

### Return value

The callback should return a pointer to the newly-allocated memory.

### Example

N/A

### See also

N/A

## NetReallocCallback

Used to define a custom version of realloc.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

```
typedef void * (*NetReallocCallback)(
    void *BlockPtr,           Pointer to block being reallocated
    size_t size);             Size of memory block to reallocate
```

### Description

This prototype is used by an application to define a custom version of the Standard C library function realloc. This user-defined version of realloc will be used for memory allocation by the DME.

### Notes

N/A

### Return value

The callback should return a pointer to the newly-allocated memory.

### Example

N/A

### See also

N/A

## NetTypeClientConnectCallback

Callback: Prototype for a client connect callback.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_nettypes.h	1.00	December 8, 2003

### Syntax

```
typedef void (*NetTypeClientConnectCallback)(  
    NetTypeClientConnectCallbackData *pData);
```

Pointer to client connect data.

### Description

This prototype is used by an application to define callback functions that are called when a remote client connects and disconnects.

### Notes

N/A

### Return value

None.

### Example

N/A

### See also

[NetInitialize](#).

## NetTypeCompletionCallback

Callback: Prototype for [NetJoin\(\)](#) completion callback.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.03	December 1, 2002

### Syntax

```
typedef void (*NetTypeCompletionCallback)(
    NetCompletionData *pCompletionData);
```

Result of non-blocking call.

### Description

This callback is issued on completion of a non-blocking call. If the function was succesful, the callback is passed on to a value of NetErrorNone. If the function failed, the callback is passed on to a value of NetErrorTimedOut.

### Notes

N/A

### Return value

None.

### Example

N/A

### See also

[NetJoinInParams](#).

## NetTypeConnectCallback

Callback: Non-blocking client callback issued from [NetConnect\(\)](#).

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_nettypes.h	1.03	November 11, 2004

### Syntax

```
typedef void (*NetTypeConnectCallback)(
    NetTypeConnectCallbackData *pData);
```

Result of non-blocking call.

### Description

This prototype is used by an application to define callback functions that are called when a pending connection is completed. This completion can be caused by either a success or failure to make a connection.

### Notes

N/A

### Return value

None.

### Example

N/A

### See also

[NetConnect](#).



## NetTypeDataStreamEndCallback

Callback: Prototype for data stream end callback.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2	rt_object.h	1.00	September 12, 2001

### Syntax

```
typedef void (*NetTypeDataStreamEndCallback)(
    int ClientIndex,           Sender's client index
    int Channel);             Data stream channel index
```

### Description

This prototype is used by an application to define a function that will be executed when a remote client has performed a NetEndDataStream on a channel. By defining such a function, a client can know when it has received the end of the data on a particular channel.

### Notes

N/A

### Return value

None.

### Example

N/A

### See also

[NetRegisterDataStream](#).

## NetTypeDataStreamFilterCallback

Callback: Prototype for a data stream filter callback.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	June 12, 2001

### Syntax

```
typedef int (*NetTypeDataStreamFilterCallback)(
    int TargetClientIndex,           Client to receive the object update
    int SourceClientIndex,          Client that sent data stream information.
    int StreamChannel);             Data stream being sent out
```

### Description

This prototype is used by an application to define custom filter functions for data streams. This callback function is called for every receiving client for every data stream update that is issued by broadcast scheduler.

### Notes

N/A

### Return value

If the target client should receive the update, the filter function must return 1. If the target client should not receive the update, the filter function must return 0.

### Example

N/A

### See also

N/A

## NetTypeDataStreamUpdateCallback

Callback: Prototype for a data stream update callback.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2	rt_object.h	1.00	September 12, 2001

### Syntax

```
typedef void (*NetTypeDataStreamUpdateCallback)(
    int ClientIndex,           Sender's client index
    int Channel,              Data stream channel index
    char *NewData,            Pointer to incoming data for this update
    int ByteCount);           Size of incoming data in bytes
```

### Description

This prototype is used by an application to define a function that will be executed when streaming data is received from a remote client.

### Notes

Callbacks are segregated by data stream type.

### Return value

None.

### Example

N/A

### See also

[NetRegisterDataStream](#).

## NetTypeErrorThresholdCallback

Callback: Prototype for an error threshold callback.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	2.09	November 11, 2004

### Syntax

```
typedef int (*NetTypeErrorThresholdCallback)(
    NetErrorThresholdCallbackData *pThresholdData);
```

Pointer to threshold data.

### Description

This prototype is used by an application to define a function that will be executed to determine if a particular field update should be sent over the network.

### Notes

N/A

### Return value

Return 1 if an update should be sent and 0 if not.

### Example

N/A

### See also

N/A

## NetTypeLatencyMetricsCallback

Callback: prototype for a latency metrics callback.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_nettypes.h	2.09	November 11, 2004

### Syntax

```
typedef void (*NetTypeLatencyMetricsCallback)(
    const NetLatencyMetricsDataArgs *pArgs);
```

Pointer to latency metrics data.

### Description

This callback is issued after NetGetLatencyMetrics has returned a value or after timeout.

### Notes

N/A

### Return value

None.

### Example

N/A

### See also

N/A

## NetTypeLookupCallback

This prototype is used by an application to define callback functions that are called when a [NetGetHostByName\(\)](#) call completes.

Link to file	Include file	Introduced	Last modified
<a href="#">librtbasePS2.a</a>	dme.h	2.09	November 11, 2004

### Syntax

```
typedef void (*NetTypeLookupCallback)(  
    NetTypeLookupResponse *pLookupResponse);
```

Pointer to lookup response.

### Description

This prototype is used by an application to define callback functions that are called when a [NetGetHostByName\(\)](#) call completes.

### Notes

N/A

### Return value

None.

### Example

N/A

### See also

N/A

## NetTypeMessageParser

User-defined callback function invoked by the DME when a message is received.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	June 18, 2004

### Syntax

```
typedef int (*NetTypeMessageParser)(
    HDME ConnectionHandle,           Connection message was received on
    int WorldID,                     World the message was sent from
    int ClientIndex,                 Client who sent the message
    void *MessageData);              Message data to be parsed
```

### Description

This user-defined callback function is invoked by the DME when a message is received.

### Notes

Messages can be of dynamic length as long as handler returns correct size.

### Return value

Very important: The application developer must have their user-defined callback functions return the size of the message in bytes. This is so that the message parser internally is incremented properly. Return the result of sizeof() of the structure you used to send and receive a message with.

### Example

N/A

### See also

N/A

## NetTypeObjectCallback

Callback: NetObject creation and deletion callback.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.10	October 31, 2001

### Syntax

```
typedef void (*NetTypeObjectCallback)(  
    int ClientIndex,                                Index of client sending the creation/deletion (object  
                                                       owner)  
    int ObjectIndex);                                Index of object being created/deleted
```

### Description

This prototype is used by an application to define two functions that will be executed when an object is created or deleted.

### Notes

N/A

### Return value

None.

### Example

N/A

### See also

[NetInitialize](#).



## NetTypeObjectFilterCallback

Callback: NetObject filter callback.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	November 11, 2004

### Syntax

```
typedef void (*NetTypeObjectFilterCallback)(
    NetObjectFilterData *pObjectFilterData);
```

Pointer to object filter data.

### Description

This prototype is used by an application to define custom filter functions. This callback function is called for every receiving client for every object and field update issued by the broadcast scheduler.

### Notes

N/A

### Return value

None

### Example

N/A

### See also

N/A

## NetTypeObjectUpdateCallback

Callback: NetObject update callback.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	October 31, 2001

### Syntax

```
typedef void (*NetTypeObjectUpdateCallback)(
    int ClientIndex,           Client sending the update (object owner).
    int ObjectIndex,          Object being updated.
    int FieldIndex);          Field that has been updated
                              (NET_FULL_OBJECT_UPDATE -> all fields).
```

### Description

This prototype is used by an application to define a function that will be executed when an object is updated.

### Notes

N/A

### Return value

None.

### Example

N/A

### See also

[NetRegisterRemoteObjectCallback](#).

## NetTypeOwnershipRequestCallback

Callback: Ownership request callback.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	<a href="#">rt_object.h</a>	2.07	June 18, 2004

### Syntax

```
typedef NetObjectType
(*NetTypeOwnershipRequestCallback)(
    NetTypeOwnershipRequestData          Pointer to ownership request data.
    *pOwnershipRequestData);
```

### Description

This prototype is used by an application to define a function to be called for processing an object ownership request.

### Notes

N/A

### Return value

None

### Example

N/A

### See also

[NetTypeOwnershipRequestData](#), [NetJoinInParams](#)

## NetTypeOwnershipUpdateCallback

Callback: NetObject ownership update callback.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	2.07	June 18, 2004

### Syntax

```
typedef void (*NetTypeOwnershipUpdateCallback)(
    NetTypeOwnershipUpdateData          Pointer to NetObject ownership update data.
    *pOwnershipUpdateData);
```

### Description

This prototype is used by an application to define a function to be called for notification of an object ownership update.

### Notes

N/A

### Return value

None.

### Example

N/A

### See also

[NetTypeOwnershipUpdateData](#), [NetJoinInParams](#)

## NetTypePeerToPeerHostChangeCallback

Callback: Peer-to-peer host change callback.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_nettypes.h	1.00	November 11, 2004

### Syntax

```
typedef void
(*NetTypePeerToPeerHostChangeCallback)(
    NetPeerToPeerHostChangeData *pHostChangeData);
```

Pointer to peer-to-peer host change data.

### Description

This prototype is used by an application to define a function to be called when the host of a peer-to-peer game changes.

### Notes

N/A

### Return value

None.

### Example

N/A

### See also

[NetPeerToPeerHostChangeData](#), [NetConnectInParams](#)

## NetTypePingCallback

Callback: NetPing information callback.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.03	December 10, 2001

### Syntax

```
typedef void (*NetTypePingCallback)(
    NetErrorCode CompletionResult,           Indicates if the ping was successful.
    char DestIP[NET_MAX_NETADDRESS_LENGTH], Destination IP address of ping.
    unsigned int Latency);                  Round trip latency to ping target.
```

### Description

Callback issued after a NetPing has returned a value, or after a timeout.

### Notes

N/A

### Return value

None.

### Example

N/A

### See also

[NetPing\(\)](#), [NetPingIP\(\)](#), [NetPingNetAddress\(\)](#)

## NetTypeRemoteClientEventCallback

Callback: Remove client NetJoin or NetLeave callback.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.24	December 1, 2002

### Syntax

```
typedef void (*NetTypeRemoteClientEventCallback)(
    NetRemoteClientEventData *pRemoteClientEventData);
```

Pointer to remote client event data.

### Description

This prototype is used by an application to define callback functions that are called when a remote client either joins (via [NetJoin\(\)](#)) a world/game or leaves (via [NetLeave\(\)](#)) a world/game.

### Notes

Players may be connected to a world/game; however, additional DME state is represented in the NetJoin and NetLeave calls (for example, calling NetLeave disassociates the client from a world/game but the client is still in a connected state. A player must call NetJoin to start creating/receiving NetObjects. If a player has not called NetJoin, they can still send NetMessages).

### Return value

None.

### Example

N/A

### See also

[NetRemoteClientEventData](#), [NetJoinInParams](#)

## NetTypeResolveAddrCallback

Callback: Address resolution callback.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

```
typedef void (*NetTypeResolveAddrCallback)(
    NetResolveAddrData *pResolveData);          (Return value) Pointer to NAT resolution data
```

### Description

This prototype is used by an application to define callback functions that are called when a pending address resolution completes.

### Notes

N/A

### Return value

None.

### Example

N/A

### See also

[NetResolveAddrData](#), [NetResolveAddrInParams](#)



## NetTypeSMChangeCallback

Callback: Session Master change/migration callback.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.24	December 2, 2002

### Syntax

```
typedef void (*NetTypeSMChangeCallback)(
    NetSMChangeData *pSMChangeData);
```

Session Master change/migration data.

### Description

This prototype is used by an application to define a function to be called when the session master changes.

### Notes

N/A

### Return value

None.

### Example

N/A

### See also

[NetSMChangeData](#), [NetJoinInParams](#)

## NetTypeStreamMediaAudioPlayCallback

Callback: Stream media audio play.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_nettypes.h	1.24	November 11, 2004

### Syntax

```
typedef void
(*NetTypeStreamMediaAudioPlayCallback)(
    NetStreamMediaAudioPlayData *pAudioPlayData);
```

Stream media audio play data.

### Description

This prototype is used by an application to define callback functions that are called when audio data is available to the application via the streaming media.

### Notes

N/A

### Return value

None.

### Example

N/A

### See also

[NetStreamMediaAudioPlayData](#), [NetStreamMediaParams](#)

## NetTypeStreamMediaAudioRecordCallback

Callback: Stream media audio record callback.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_nettypes.h	1.24	November 11, 2004

### Syntax

```
typedef void
(*NetTypeStreamMediaAudioRecordCallback)(
    NetStreamMediaAudioRecordData *pAudioRecordData);
```

Stream media audio record data.

### Description

This prototype is used by an application to define callback functions that are called when audio data is to be sent via the streaming media.

### Notes

N/A

### Return value

None.

### Example

N/A

### See also

[NetStreamMediaAudioRecordData](#), [NetStreamMediaParams](#)

## NetStreamMediaCustomVideoPlayCallback

Callback: Stream media custom video play callback.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_nettypes.h	1.24	November 11, 2004

### Syntax

```
typedef void
(*NetStreamMediaCustomVideoPlayCallback)(
    NetStreamMediaCustomVideoPlayData          Pointer to stream media custom video play data.
    *pCustomVideoPlayData);
```

### Description

This prototype is used by an application to define callback functions that are called when video data is available to the application via the streaming media.

### Notes

N/A

### Return value

None.

### Example

N/A

### See also

[NetStreamMediaCustomVideoPlayData](#), [NetStreamMediaParams](#)

## NetStreamMediaCustomVideoRecordCallback

Callback: Stream media custom video record callback.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_nettypes.h	1.24	November 11, 2004

### Syntax

```
typedef void
(*NetTypeStreamMediaCustomVideoRecordCallback)(
    NetStreamMediaCustomVideoRecordData          Pointer to stream media custom record data.
    *pCustomVideoRecordData);
```

### Description

This prototype is used by an application to define callback functions that are called when video data is to be sent via the streaming media.

### Notes

N/A

### Return value

None.

### Example

N/A

### See also

[NetStreamMediaCustomVideoRecordData](#), [NetStreamMediaParams](#)

## NetTypeStreamMediaVideoPlayCallback

Callback: Stream media video play callback.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_nettypes.h	1.24	November 11, 2004

### Syntax

```
typedef void (*NetTypeStreamMediaVideoPlayCallback)(
    NetStreamMediaVideoPlayData *pVideoPlayData);
```

Pointer to stream media video play data.

### Description

This prototype is used by an application to define callback functions that are called when video data is available to the application via the streaming media.

### Notes

N/A

### Return value

None.

### Example

N/A

### See also

[NetStreamMediaVideoPlayData](#), [NetStreamMediaParams](#)

## NetTypeStreamMediaVideoRecordCallback

Callback: Stream media video record callback.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_nettypes.h	1.24	November 11, 2004

### Syntax

```
typedef void
(*NetTypeStreamMediaVideoRecordCallback)(
    NetStreamMediaVideoRecordData *pVideoRecordData);
```

Pointer to stream media video record data.

### Description

This prototype is used by an application to define callback functions that are called when video data is to be sent via the streaming media.

### Notes

N/A

### Return value

None.

### Example

N/A

### See also

[NetStreamMediaVideoRecordData](#), [NetStreamMediaParams](#)

## NetTypeSystemMessageCallback

Callback: Triggered upon receipt of system message.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.31	December 8, 2003

### Syntax

```
typedef void (*NetTypeSystemMessageCallback)(
    NetTypeSystemMessageData *pSystemMessageData);
```

Structure containing a DME System Message DME, and related fields.

### Description

This prototype is used by an application to define a function to be called upon reception of a server system message.

### Notes

N/A

### Return value

None.

### Example

N/A

### See also

[NetTypeSystemMessageData](#), [NetInitializeInParams](#)



## NetTypeSystemStatusCallback

Callback: Triggered when a system status message becomes available.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_nettypes.h	2.09	November 11, 2004

### Syntax

```
typedef void (*NetTypeSystemStatusCallback)(
    NetSystemStatusData *pStatusData);
```

Pointer to system status data.

### Description

System status callback prototype to be defined by the application so they may be notified of various DME system status updates.

### Notes

N/A

### Return value

None.

### Example

N/A

### See also

[NetSystemStatusData](#), [NetConnectInParams](#)

## NetTypeTokenOwnershipNotifyCallback

Callback: DME Token ownership notification callback.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_nettypes.h	2.09	November 11, 2004

### Syntax

```
typedef void (*NetTypeTokenOwnershipNotifyCallback)(  
    NetTokenOwnershipNotifyData *pTokenData);
```

Pointer to token ownership notification data.

### Description

This callback is called by the DME when token ownership is released, granted.

### Notes

N/A

### Return value

None.

### Example

N/A

### See also

[NetTokenOwnershipNotifyData](#), [NetTokenParams](#)

## Chapter 6: Functions

This page intentionally left blank.

## KM\_GetSoftwareID

Retrieve ApplicationID from security key library.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtcryptPS2.a	rt_crypt.h	1.00	June 12, 2001

### Syntax

```
unsigned int KM_GetSoftwareID();
```

### Description

This function retrieves the unique ApplicationID from the security key library that has been linked in the application.

### Notes

Each outside developer should be provided with a different `rt_softkey_***` ! library file. It is compiled using different key pairs.

### Return value

Returns the ApplicationID hardcoded in the security key library.

### Example

N/A

### See also

N/A

**NetAddressToStringAddress**

Function that converts [NetAddress](#) to a string address.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	dme.h	1.00	November 11, 2004

**Syntax**

<b>NetErrorCode</b> <b>NetAddressToStringAddress</b> (	
<b>NetAddress</b> *pNetAddress,	Binary Net Address
<b>NetAddressType</b> *pType,	Enumeration of address types.
char *pAddressBuf,	Pointer to the destination buffer containing the address.
unsigned int nAddressBufSize,	Size of the buffer in bytes.
unsigned short *pPort,	Address in which the port will be placed.
unsigned short *pVirtualPort);	Address in which the virtual port will be assigned.

**Description**

Extracts the data passed in the [NetAddress](#) structure to the various parameters passed in. If a field is not desired, then it may be set to NULL or zero.

**Notes**

N/A

**Return value**

NetErrorNone: If successful.

**Example**

N/A

**See also**

[NetAddress](#), [NetAddressType](#)

## NetAddToDataStream

Function that adds data to the stream buffer.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	May 31, 2001

### Syntax

```
NetErrorCode NetAddToDataStream(
    int Channel,
    char *SendData,
    int ByteCount);
```

Data stream channel to to which the data will be added.

Pointer to the buffer containing the data to be added to the data stream.

Number of bytes pointed to by SendData

### Description

Function that adds data to the stream buffer.

### Notes

ByteCount must not exceed the BufferSize. This function is used when creating a Data Stream via NetOpenDataStream.

### Return value

**NetErrorNone** If successful.

**NetErrorBadDataStreamChannel** If invalid channel.

### Example

N/A

### See also

[NetOpenDataStream](#)

## NetAssignLocalServer

Function that tells the DME that this connection is hosting a server.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a *	dme.h	1.11	June 18, 2004

### Syntax

```
NetErrorCode NetAssignLocalServer(  
    HDME LocalServerConnectionHandle,           Connection handle of the local server.  
    int LocalServerPort);                        Port of the local server.
```

### Description

This function tells the DME that this connection is hosting a server. This allows a DME client to internally assign a DME server and DME port number. These values are used when a remote client issues a call to NetRequestServersOnSubnet(). This function only needs to be called in the context of an "integrated server" game. This call happens automatically in a peer-to-peer game.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

N/A



## NetBitMaskIsSet

Function that sets b\_set to 1 or 0.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

```
NetErrorCode NetBitMaskIsSet(
    const NetBitMask *pBitMask,           Bitmask to be checked
    const int mask_index,                 Index in the bitmask to ckeck
    int *b_set);                          Return the value -1 if set, and return the value 0 if
                                         not set.
```

### Description

Takes a pointer to a [NetBitMask](#) structure and an index within the bitmask range, and sets b\_set to 1 if the bit is set or 0 if not.

### Notes

The mask\_index parameter must be contained within the range of base\_id and max\_id. Thus: pBitMask->base\_id <= mask\_index < pBitMask->max\_id

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetBitMask](#), [NetBitMaskIsSet\(\)](#), [NetBitMaskUnSet\(\)](#)

**NetBitMaskSet**

Function that sets an index within a bitmask range.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	dme.h	2.09	November 11, 2004

**Syntax**

```

NetErrorCode NetBitMaskSet(
    NetBitMask *pBitMask,           Bitmask to set.
    const int mask_index);          Bits to mask on a bitwise operation.

```

**Description**

Takes a pointer to a [NetBitMask](#) structure and sets an index within the bitmask range.

**Notes**

The mask\_index parameter must be contained within the range of base\_id and max\_id. Thus: pBitMask->base\_id <= mask\_index < pBitMask->max\_id

**Return value**

NetErrorNone: If successful.

**Example**

N/A

**See also**

[NetBitMask](#), [NetBitMaskIsSet\(\)](#), [NetBitMaskUnSet\(\)](#)

## NetBitMaskUnSet

Function that unsets a [NetBitMask](#).

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

```
NetErrorCode NetBitMaskUnSet(
  NetBitMask *pBitMask,           NetBitMask to unset.
  const int mask_index);          Bits to mask on a bitwise operation.
```

### Description

Takes a pointer to a [NetBitMask](#) structure and zeros an index within the bitmask range.

### Notes

The mask\_index parameter must be contained within the range of base\_id and max\_id. Thus: pBitMask->base\_id <= mask\_index < pBitMask->max\_id

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetBitMask](#), [NetBitMaskIsSet\(\)](#), [NetBitMaskUnSet\(\)](#)

## NetClose

Function that shuts down a network connection and frees the associated data structures.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	June 12, 2001

### Syntax

```
NetErrorCode NetClose(  
void );
```

### Description

Shutdown a network connection and free the associated data structures. An application should call [NetDisconnect\(\)](#) before calling [NetClose\(\)](#). If [NetInitialize\(\)](#) is called again, NetMessages and NetObjects do not need to be reregistered.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetInitialize\(\)](#)

## NetConnect

This function connects the client application to either the specified RTIME server or peer-to-peer host.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	dme.h	1.00	March 13, 2002

### Syntax

```
NetErrorCode NetConnect(
    const NetConnectInParams *pInParams,           Pointer to input parameters.
    NetConnectOutParams *pOutParams);           Pointer to output parameters.
```

### Description

This function connects the client to the specified SCE-RT server or peer-to-peer host (based on the pInParams->ConnectionInfo field). If the ConnectCallback argument is NULL, then this function will block.

### Notes

[NetConnectOutParams](#): A client can have multiple connections to various servers. Each connection is uniquely identified by the DME via the ConnectionHandle.

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetConnectInParams](#), [NetConnectOutParams](#), [NetSetDefaultConnectParams\(\)](#), [NetTypeConnectCallback](#), [NetGetConnectionStatus\(\)](#), [NetDisconnect\(\)](#)

NetCreateObject

Function that creates a network object.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_object.h	1.00	September 12, 2001

Syntax

<b>NetErrorCode</b> NetCreateObject( int *ObjectIndex,  <b>NetOwnershipStatus</b> Ownership,  <b>NetObjectLifespan</b> LifespanType, unsigned int LifeSpan,  int FilterType,   unsigned int MaxUpdateInterval,  int LatencyCritical, void *ObjectData, char *ObjectName, int NetStructureIndex);	<p>The object identifier of the newly created object. This is an integer greater than or equal to zero identifying the created object.</p> <p>The ownership status of the object: OwnershipShared, OwnershipPrivate, or OwnershipNone.</p> <p>The persistence of the object.</p> <p>Not implemented. The number of milliseconds for which the object should persist.</p> <p>The filter type for the object as returned by NetRegisterObjectFilter. If the object does not need to be filtered, use NET_OBJECT_NOT_FILTERED.</p> <p>The number of milliseconds between full object updates Set this value to zero to disable.</p> <p>True or False</p> <p>Copy of the initial data</p> <p>Generic name for the object</p> <p>This value is returned by NetRegisterStructure.</p>
--	---

Description

Function that creates a network object.

Notes

N/A

Return value

<b>NetErrorNone</b>	If successful.
<b>NetErrorNoFreeObject</b>	Maximum number of Network objects exceeded.

Example

N/A

See also

NetOwnershipStatus, NetObjectLifespan, NetFreeObject(), NetGetObject()

## NetDataStreamBytesFree

This function retrieves the number of free data stream bytes.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	May 31, 2001

### Syntax

**NetErrorCode** NetDataStreamBytesFree(

int \*ReturnedBytesFree,

The number of bytes available based on the DataStreamChannel. The value must be less than the BufferSize set in NetOpenDataStream.

int DataStreamChannel);

Channel number for the Data Stream

### Description

This function returns the number of bytes available based upon the DataStreamChannel. The value must be less than the BufferSize set in NetOpenDataStream.

### Notes

N/A

### Return value

**NetErrorNone**

If successful.

**NetErrorBadIndex**

If invalid channel.

### Example

N/A

### See also

N/A

## NetDisableLanMessaging

Function that disables LAN messaging.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_lanmessage.h	2.09	November 11, 2004

### Syntax

```
NetErrorCode NetDisableLanMessaging();
```

### Description

This function disables LAN messaging.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

N/A



## NetDisconnect

Function that disconnects from the specified connection.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	dme.h	1.00	May 31, 2001

### Syntax

```
NetErrorCode NetDisconnect(  
  NetDisconnectParams *pDisconnectParams);
```

Pointer to the disconnect information.

### Description

Disconnect from the specified connection.

### Notes

N/A

### Return value

NetErrorNode: If successful.

### Example

N/A

### See also

[NetDisconnectParams](#), [NetSetDefaultDisconnectParams\(\)](#), [NetConnect\(\)](#)

## NetEnableLANMessaging

This function enables LAN messaging.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_lanmessage.h	2.09	November 11, 2004

### Syntax

```
NetErrorCode NetEnableLANMessaging(  
  NetEnableLanMessagingInParams *pParams);
```

Pointer to LAN messaging enable parameters.

### Description

This function enables LAN messaging.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetEnableLanMessagingInParams](#), [NetLANSetDefaultEnableMessagingInParams\(\)](#)

## NetEndDataStream

Function that ends a DataStream.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	May 31, 2001

### Syntax

```
NetErrorCode NetEndDataStream(  
    int Channel);
```

Channel to be closed.

### Description

Close the DataStream associated with the Channel. Data stream will not close until all pending broadcasts are completed. This call is non-blocking.

### Notes

N/A

### Return value

<b>NetErrorNone</b>	If successful.
<b>NetErrorBadDataStreamChannel</b>	If the channel is invalid.

### Example

N/A

### See also

[NetTypeDataStreamEndCallback\(\)](#)

## NetFreeAllObjects

Function that releases and frees all network objects.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	May 31, 2001

### Syntax

```
NetErrorCode NetFreeAllObjects(  
    void );
```

### Description

This function deletes all of the network objects that are owned by this client.

### Notes

A client will have NetObject ownership if they created a NetObject and the NetObject is in a shared state (not owned by another client). A NetObject is deleted if the NetObject was created by another client, but currently is owned by a client that calls [NetFreeAllObjects\(\)](#).

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetFreeObject\(\)](#), [NetCreateObject\(\)](#), [NetGetObject\(\)](#)

## NetFreeObject

This function deletes the network object associated with the given ObjectIndex if this client currently has NetObject ownership.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	May 31, 2001

### Syntax

**NetErrorCode** NetFreeObject(

int *ObjectIndex*);

NetObject index to delete.

### Description

Function that releases and frees the given network object (NetObject).

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetFreeAllObjects\(\)](#), [NetCreateObject\(\)](#), [NetGetObject\(\)](#)

## NetGenerateClientList

Function that generates a client list based upon the players who are connected.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

```
NetErrorCode NetGenerateClientList(
    HDME ConnectionHandle,           Connection handle of the given connection.
    NetClientList *pClientList);      The returned client list.
```

### Description

This function fills out a [NetBitMask](#) structure that maps to the current clients who are connected to the game.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetClientList](#), [NetBitMask](#), [NetGenerateJoinClientList\(\)](#)

## NetGenerateJoinedClientList

Function that generates a client list based on players who are joined to the game.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

```
NetErrorCode NetGenerateJoinedClientList(
HDME ConnectionHandle,           Connection handle of the given connection.
NetClientList *pClientList);    The returned client list.
```

### Description

Fills out a [NetBitMask](#) structure that maps to the current clients NetJoined to the game.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetClientList](#), [NetBitMask](#), [NetGenerateClientList\(\)](#)

## NetGetAverageDelayToClient

Function that returns the one-way latency to a client in milliseconds.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.18	March 13, 2002

### Syntax

**NetErrorCode** NetGetAverageDelayToClient(

**unsigned int** \*ReturnedDelay,

    Latency to client in milliseconds.

**int** ClientIndex);

    Client for which to get the metric.

### Description

This function returns the number of milliseconds it takes to send data to a particular client. The DME internally keeps track of these values. If an application wants to determine its own latency, use call [NetPing\(\)](#).

### Notes

The client must be joined (via [NetJoin\(\)](#)).

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

N/A



## NetGetBandwidthInfo

This function retrieves information about the given bandwidth usage.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.18	March 13, 2002

### Syntax

```
NetErrorCode NetGetBandwidthInfo(  
    NetBandwidthInfo *BandwidthInfo);
```

Returned bandwidth information.

### Description

This function retrieves information about the given bandwidth usage.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetBandwidthInfo](#)

## NetGetBufferStatus

Function that retrieves the buffer status for a given connection.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.03	June 18, 2004

### Syntax

**NetErrorCode** NetGetBufferStatus(

**HDME** *ConnectionHandle*,

**unsigned int** \*UsedSend,

**unsigned int** \*UsedRecv,

**unsigned int** \*UsedSendPer,

**unsigned int** \*UsedRecvPer);

Connection handle of the given connection.

Pointer to amount used in send buffer

Pointer to amount used in receiving buffer

Pointer to the percentage of send buffer used

Pointer to the percentage of receiving buffer used

### Description

This function retrieves buffer status information for a given connection. Information includes the amount of buffer space currently being used in both the send and receive buffers.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetGetBandwidthInfo\(\)](#)

## NetGetBuildTimeStamp

Function that returns the build time stamp of this DME client.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	September 12, 2001

### Syntax

```
NetErrorCode NetGetBuildTimeStamp(  
    char ReturnedTimeStamp                                Returned build time stamp (length of 32).  
    [NET_TIMESTAMP_STRING_LENGTH]);
```

### Description

Get this DME client's build time stamp. The time stamp indicates when the DME Library was actually built. This function returns a pointer to a string containing the build version of the DME API library. The string is in the form: "Time of build 10:59:15 Dec 16 2003".

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetGetClientVersion\(\)](#), [NetGetServerVersion\(\)](#)

NetGetClient

This function retrieves client information.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_object.h	1.00	December 10, 2001

Syntax

NetErrorCode	NetGetClient(	
NetTypeClient	**ClientPtr,	Returned client information.
HDME	ConnectionHandle,	Connection handle of the given connection.
int	ClientIndex);	Client ID for which to get information.

Description

This function retrieves information for the specified client based on the ConnectionHandle and ClientIndex. It can only be called after calling NetJoin().

Notes

N/A

Return value

NetErrorNone	If successful.
NetErrorBadIndex	If the ClientIndex is invalid.

Example

N/A

See also

NetTypeClient, NetGetClientStatus()

## NetGetClientIpAddress

This function retrieves the IP address for the specified client.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	June 18, 2004

### Syntax

```
NetErrorCode NetGetClientIpAddress(
    char IpAddress[NET_MAX_NETADDRESS_LENGTH],    Returned IP address.
    HDME ConnectionHandle,                        Connection handle of the given connection.
    int ClientIndex);                             Client ID for which to get address information.
```

### Description

This function retrieves the IP address of the specified client based upon the ConnectionHandle and ClientIndex.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

N/A

## NetGetClientStatus

This function retrieves status information for a client.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.11	December 10, 2001

### Syntax

```
NetErrorCode NetGetClientStatus(
NetClientStatus *ReturnedClientStatus,           Returned client status information.
HDME ConnectionHandle,                           Connection handle of the given connection.
int ClientIndex);                                Client ID for which to get client status information.
```

### Description

This function retrieves information for the specified client based on the ConnectionHandle and ClientIndex.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetClientStatus](#)

## NetGetClientVersion

Function that gets the library version number for this DME client. \*.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.10	September 12, 2001

### Syntax

**NetErrorCode** NetGetClientVersion(

**char** *ReturnedVersion*

[NET\_VERSION\_STRING\_LENGTH]);

The returned DME client library version number.  
(The length is 16 characters.)

### Description

Get the library version number for this DME client. The string is in the form: v.vv.vvvv. This indicates the major build number, the minor build number, and the build number. A change in the build number indicates a low level code optimization or bug fix; no changes at the application level are required. Changes in the second digit of the minor build (e.g., 1.03.0001 to 1.04.0001) indicate that additional functionality has been added, but changes to the application software are only required if the new functionality is used. A change in the first digit of the minor build number (e.g., 1.03.0001 to 1.10.0001) indicates that either a change to the application is required, or that there have been changes that affect compatibility with other components (e.g., the backend services).

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetGetBuildTimeStamp\(\)](#)

## NetGetConnectionStatus

Function that gets the status of a specified connection.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.31	December 8, 2003

### Syntax

```
NetErrorCode NetGetConnectionStatus(  
  NetConnectionStatus *ConnectionStatus,           Pointer to the returned connection status.  
  HDME ConnectionHandle);                          Connection handle of the given connection.
```

### Description

This function fills out a [NetConnectionStatus](#) structure with information about the connection.

### Notes

[NetGetConnectStatus\(\)](#) and [NetGetConnectionStatus\(\)](#) both return [NetConnectStatus](#) information. [NetGetConnectionStatus\(\)](#) just returns more information than [NetGetConnectStatus\(\)](#).

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetConnectionStatus](#), [NetGetConnectStatus\(\)](#)



## NetGetConnectStatus

This function gets the status of a specified connection.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	September 12, 2001

### Syntax

```
NetErrorCode NetGetConnectStatus(  

NetConnectStatus *ReturnedStatus,           Pointer to the returned connect status information.  

HDME ConnectionHandle);                     Connection handle of the given connection.
```

### Description

This function will fill out a NetConnectStatus structure with various information regarding the connection.

### Notes

[NetGetConnectStatus\(\)](#) and [NetGetConnectionStatus\(\)](#) both return NetConnectStatus information. [NetGetConnectionStatus\(\)](#) just returns more information than [NetGetConnectStatus\(\)](#).

### Return value

ReturnedStatus NetErrorNone: If successful.

### Example

N/A

### See also

[NetConnectStatus](#), [NetGetConnectionStatus\(\)](#)

## NetGetDataStream

This function gets data stream channel information.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	May 31, 2001

### Syntax

```
NetErrorCode NetGetDataStream(
NetTypeDataStream **DataStreamPtr,           Pointer to the returned data stream information.
int DataStreamChannel);                      Channel for which to get data stream.
```

### Description

This function retrieves information for the specified data stream channel.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetTypeDataStream](#)

## NetGetHostByName

This function issues a DNS request.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

```
NetErrorCode NetGetHostByName(  
  NetTypeLookupParams *pLookupParams);
```

Pointer to the returned DNS lookup information.

### Description

This function issues a DNS request and returns the response via the specified callback.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetTypeLookupParams](#), [NetTypeLookupResponse](#), [NetTypeLookupCallback](#)

**NetGetLatencyMetrics**

This function gets latency metrics about a client.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	dme.h	2.09	November 11, 2004

**Syntax**

**NetErrorCode**
**NetGetLatencyMetrics**(  
**NetLatencyMetricsParams** \*pParams);

Pointer to the returned latency metric information requested.

**Description**

This function gets latency metrics about a client.

**Notes**

N/A

**Return value**

NetErrorNone: If successful.

**Example**

N/A

**See also**

[NetLatencyMetricsParams](#), [NetTypeLatencyMetricsCallback](#)

## NetGetLocalTime

This function returns the time elapsed since the program began.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	May 31, 2001

### Syntax

**NetErrorCode** NetGetLocalTime(

    unsigned int \*LocalTime);

The returned local time.

### Description

This function returns the time elapsed since the program began. The resolution is in milliseconds. Use [NetGetTime\(\)](#) for a Global time base.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetGetTime\(\)](#)

## NetGetMyClientIndex

This function gets my own client index.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	June 12, 2001

### Syntax

**NetErrorCode** NetGetMyClientIndex(

int \*ReturnedClientIndex,

Returned client index.

**HDME** ConnectionHandle);

Connection handle of the given connection.

### Description

This function returns the Client index for this client based upon the ConnectionHandle.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

N/A

## NetGetMyIpAddress

This function gets my client's local machine IP address.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	June 12, 2001

### Syntax

```
NetErrorCode NetGetMyIpAddress(  
    char ReturnedIpAddress                The returned IP address.  
    [NET_MAX_NETADDRESS_LENGTH]);
```

### Description

This function returns the IP address of the local machine.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetGetMyNetAddress\(\)](#)

## NetGetMyNetAddress

This function gets my clients local machine DME [NetAddress](#).

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	June 12, 2001

### Syntax

```
NetErrorCode NetGetMyNetAddress(
  NetAddress *pNetAddress,           Pointer to the NetAddress to return.
  HDME ConnectionHandle);           Connection handle of the given connection.
```

### Description

This function returns the Netaddress of the local machine. Setting ConnectionHandle to NET\_NO\_CONNECTION will not get the virtual port.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetAddress](#), [NetGetMyIpAddress\(\)](#)



## NetGetNetUpdateErrors

This function gets [NetUpdate\(\)](#) error information.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	dme.h	1.18	March 13, 2002

### Syntax

```
NetErrorCode NetGetNetUpdateErrors(
    NetUpdateConnErrors *pNetUpdateErrorStruct);
```

Pointer to the returned [NetUpdateConnErrors](#) information.

### Description

This function fills out a [NetUpdateConnErrors](#) structure with extended error information on a per connection index basis. If multiple errors occurred on a particular connection, then only the last error is returned. This function fills out error information about errors from NetUpdate or NetTick.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetUpdateConnErrors](#)

## NetGetObject

This function gets NetObject information.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	June 12, 2001

### Syntax

**NetErrorCode** NetGetObject(  
**NetTypeObject** \*\*ObjectPtr,  
  
int ObjectIndex);

Pointer to a pointer to the returned NetObject information.  
  
The Object ID of the NetObject for which information is to be returned.

### Description

This function retrieves information for the specified network object.

### Notes

N/A

### Return value

NetErrorNone

If successful.

NetErrorBadIndex

If invalid ObjectIndex.

### Example

N/A

### See also

N/A

## NetGetPeerToPeerHostClientIndex

This function gets ClientIndex of the peer-to-peer host..

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.18	June 18, 2004

### Syntax

**NetErrorCode** NetGetPeerToPeerHostClientIndex(

**HDMF** *ConnectionHandle*,

**int** *\*HostClientIndex*);

Connection handle of the given connection.

Pointer to the returned ClientIndex of the peer-to-peer host.

### Description

This function returns the client index of the peer-to-peer host.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetGetMyClientIndex\(\)](#), [NetGetSessionMasterClientIndex\(\)](#)

## NetGetServerVersion

This function gets the DME game server version.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.10	June 18, 2004

### Syntax

```
NetErrorCode NetGetServerVersion(
    char ReturnedVersion                                The returned server version string (of size 16).
    [NET_VERSION_STRING_LENGTH],
    HDME ConnectionHandle);                             Connection handle of the given connection.
```

### Description

This function returns the DME game server version in the case of a client-server game. The returned string is in the form: v.vv.vvvv. This indicates the major build number, the minor build number, and the build number.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetGetClientVersion\(\)](#)

## NetGetSessionMasterClientIndex

This function gets the ClientIndex of the Session Master.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	June 18, 2004

### Syntax

**NetErrorCode** NetGetSessionMasterClientIndex(

int \*SessionMasterClientIndex,

Pointer to the returned client index of the session master.

HDME ConnectionHandle);

Connection handle of the given connection.

### Description

This function returns the client index associated with the session master. It must be NetJoined to succeed.

### Notes

N/A

### Return value

NetErrorNone: If successful

### Example

N/A

### See also

[NetGetPeerToPeerHostClientIndex\(\)](#), [NetGetMyClientIndex\(\)](#)

## NetGetTime

This function gets the DME global time base as managed by the Server/Host.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	May 31, 2001

### Syntax

```
NetErrorCode NetGetTime(  
    HDME ConnectionHandle,           Connection handle of the given connection.  
    unsigned int *GlobalTime);       Pointer to the returned global timebase.
```

### Description

This function returns the global time as seen by the Server/Host. This function returns a positive integer that indicates the current time. The resolution is in milliseconds.

Every time a new game world is created, a global time base is created with it. Then, every client that connects to this game world is synced with this value. Leveraging a global timebase is very important for gaming techniques that involve things like physics in a distributed network.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

N/A

## NetGetValidClientCount

This function gets the client count on a given connection.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	June 18, 2004

### Syntax

**NetErrorCode** NetGetValidClientCount(

int \*ReturnedClientCount,

Pointer to the returned client count on the given connection.

HDME ConnectionHandle);

Connection handle of the given connection.

### Description

This function returns the number of clients connected based on the given ConnectionHandle.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

N/A

## NetHostPeerToPeer

This function sets this client to be in a hosting state for peer-to-peer games.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.18	March 13, 2002

### Syntax

```
NetErrorCode NetHostPeerToPeer(
    const NetHostPeerToPeerInParams *pInParams,      Pointer to input parameters.
    NetHostPeerToPeerOutParams *pOutParams);        Pointer to output parameters.
```

### Description

This function puts this client into a hosting state for peer-to-peer games. As soon as this function's callback returns with success, other clients will then be able to start attempting to [NetConnect\(\)](#) to this client.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetHostPeerToPeerInParams](#), [NetHostPeerToPeerOutParams](#), [NetSetDefaultHostPeerToPeerParams\(\)](#), [NetTypeConnectCallback](#)



## NetIncomingClient

This function notifies the peer-to-peer host that a client will connect.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

```
NetErrorCode NetIncomingClient(
    const NetIncomingClientInParams *pInParams,      Pointer to input parameters.
    NetIncomingClientOutParams *pOutParams);        Pointer to output parameters.
```

### Description

This function notifies the peer-to-peer host that a client will connect.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetIncomingClientInParams](#), [NetIncomingClientOutParams](#), [NetSetDefaultIncomingClientParams\(\)](#)

NetInitialize

This function initializes the DME API.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	dme.h	1.00	June 12, 2001

Syntax

```

NetErrorCode NetInitialize(
    const NetInitializeInParams *pInParams,           Pointer to input parameters
    NetInitializeOutParams *pOutParams);              Pointer to output parameters

```

Description

This function initializes the DME API.

Notes

NetClose() is called to uninitialize the DME API.

Return value

NetErrorNone: If successful.

Example

N/A

See also

NetInitializeInParams, NetInitializeOutParams, NetSetDefaultInitializeParams(), NetUseDme(), NetClose(), NetSetSendAggregationInterval()

# NetJoin

This function joins a game world.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_object.h	1.00	December 1, 2002

## Syntax

```
NetErrorCode NetJoin(
    const NetJoinInParams *pInParams,           Pointer to input parameters.
    NetJoinOutParams *pOutParams);              Pointer to output parameters.
```

## Description

Sets the client to be in a joined state with the game world with which they are connected. NetObjects and Session Master information are only available after a [NetJoin\(\)](#) succeeds. Call [NetLeave\(\)](#) to un-join the game world and free the NetObjects.

## Notes

[NetConnect\(\)](#) is called to connect to a DME game world. If the connect is successful, then NetMessages can be sent. However, to start working with NetObjects you must call [NetJoin\(\)](#). Sometimes, when a client has just connected to a world, they may be at a staging GUI screen and are not ready for NetObject world data to be sent to them. When the user is ready to join the game (from the staging screen), then they usually call [NetJoin\(\)](#) to have their NetObjects created, to have remote player NetObjects created, and to transition to the game.

## Return value

NetErrorNone: If successful.

## Example

N/A

## See also

[NetJoinInParams](#), [NetJoinOutParams](#), [NetSetDefaultJoinParams\(\)](#), [NetLeave\(\)](#), [NetSetMyClientReceiveBroadcast\(\)](#)

## NetLANFind

This function searches for games on the LAN.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	<a href="#">rt_lanfind.h</a>	2.09	November 11, 2004

### Syntax

```
NetErrorCode NetLANFind(  
  NetLANFindInParams *pParams);
```

Pointer to input parameters.

### Description

This function searches the LAN for games.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetLANFindInParams](#), [NetSetDefaultLANFindParams\(\)](#), [NetLANFindCallback](#), [NetLANFindCancel\(\)](#)

## NetLANFindCancel

This function cancels a search for LAN games.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_lanfind.h	2.09	November 11, 2004

### Syntax

```
NetErrorCode NetLANFindCancel();
```

### Description

This function clears the callback set in NetLANFind. This results in suppressing any callbacks for LANFind results. This is typically called after a client connects to a game.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetLANFind\(\)](#)

## NetLANFindEnableExchange

This function controls how receivers of LANFind messages respond to requests.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_lanfind.h	2.09	November 11, 2004

### Syntax

<b>NetErrorCode</b> NetLANFindEnableExchange( int <i>bEnable</i> ,	TRUE - Allows client to respond. FALSE - Client never responds.
<b>NetLANFindExchangeCallback</b> <i>pCB</i> ,	If <i>pCB</i> is set to NULL, then DME always responds to the client; otherwise, <i>pCB</i> is called to respond to the client.
void * <i>pUserData</i> );	Pointer to <i>UserData</i> that is available when the callback is triggered. . *

### Description

This function enables the receiver of the LANFind to enable, disable, or control which clients can see them.

### Notes

If *bEnable* is set to TRUE, then the client can respond. Specifying a *pCB* allows a developer to override responses. Setting *pCB* to NULL causes the DME to automatically respond to a client. If *bEnable* is set to FALSE, then the client is virtually invisible to all other clients.

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetLANFindExchangeCallback](#)

## NetLANSendRawMessage

This function sends a message of raw data.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	<a href="#">rt_lanmessage.h</a>	2.09	November 11, 2004

### Syntax

```
NetErrorCode NetLANSendRawMessage(  
  NetLANSendRawMessageInParams *pParams);
```

Pointer to input parameters.

### Description

This function sends a LAN message of raw data to another client.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetLANSendRawMessageInParams](#), [NetLANSetDefaultSendRawMessageInParams\(\)](#),  
[NetLANSendTextMessage\(\)](#), [NET\\_LAN\\_RAW\\_MESSAGE\\_CALLBACK](#)

## NetLANSendMessage

This function sends a LAN message of text data.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_lanmessage.h	2.09	November 11, 2004

### Syntax

```
NetErrorCode NetLANSendMessage(  
  NetLANSendMessageInParams *pParams);
```

Pointer to input parameters.

### Description

This function sends a LAN message of text data to another client.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetLANSendMessageInParams](#), [NetLANSetDefaultSendMessageInParams\(\)](#),  
[NetLANSendMessage\(\)](#), [NetLANSendRawMessage\(\)](#), [NET\\_LAN\\_TEXT\\_MESSAGE\\_CALLBACK](#)



## NetLANSetDefaultEnableMessagingInParams

This function enables LAN messaging.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	<a href="#">rt_lanmessage.h</a>	2.09	November 11, 2004

### Syntax

[NetErrorCode](#)

**NetLANSetDefaultEnableMessagingInParams**(

[NetEnableLanMessagingInParams](#) \*pParams);      Pointer to input parameters.

### Description

This function enables LAN messaging.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetEnableLanMessagingInParams](#), [NetLANSendRawMessage\(\)](#), [NetLANSendTextMessage\(\)](#)

**NetLANSetDefaultSendRawMessageInParams**

This function sets default values for [NetLANSendRawMessageInParams](#).

Link to file	Include file	Introduced	Last modified
<a href="#">librtbasePS2.a</a>	<a href="#">rt_lanmessage.h</a>	2.09	November 11, 2004

**Syntax**

```

NetErrorCode
NetLANSetDefaultSendRawMessageInParams(
    NetLANSendRawMessageInParams *pParams);           Pointer to input parameters.
    
```

**Description**

This function will initialize the structure to 0, and then set any SCE-RT required default values to [NetLANSendRawMessageInParams](#).

**Notes**

N/A

**Return value**

NetErrorNone: If successful.

**Example**

N/A

**See also**

[NetLANSendRawMessageInParams](#), [NetLANSendRawMessage\(\)](#)

## NetLANSetDefaultSendTextMessageInParams

This function sets default values for [NetLANSendTextMessageInParams](#).

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	<a href="#">rt_lanmessage.h</a>	2.09	November 11, 2004

### Syntax

#### NetErrorCode

**NetLANSetDefaultSendTextMessageInParams**(

[NetLANSendTextMessageInParams](#) \*pParams);      Pointer to input parameters.

### Description

This function initializes the structure to 0, and then sets any SCE-RT required default values to [NetLANSendTextMessageInParams](#).

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetLANSendTextMessageInParams](#), [NetLANSendTextMessage\(\)](#)

## NetLANSetUserName

This function assigns a user name for use on the LAN.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

```
NetErrorCode NetLANSetUserName(  
    const unsigned char *pUserName,           Pointer to the user name to assign.  
    int nSize);                               Size in bytes of the user name.
```

### Description

This function assigns a user name for this client. This user name is sent to other peers when a NetLANFind is issued. The user name also is returned to the caller of the LANFind.

### Notes

The argument specified is in [NetInitializeParams::Localization](#) format. The length must not exceed MAX\_USER\_NAME.

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

N/A

## NetLeave

This function unjoins a DME game world.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	June 18, 2004

### Syntax

```
NetErrorCode NetLeave(  
    HDMF ConnectionHandle);
```

Connection handle of the given connection.

### Description

Unjoins a DME game world. The client will still be connected to a DME game world and will still be able to send NetMessages; however, all NetObjects will be freed.

### Notes

[NetLeave\(\)](#) is often used when a client wants to leave a game before it is complete. They call [NetLeave\(\)](#) and transition to a summary page to review the current statistics of that game. Then, the client could transition to an equipment selection screen and rejoin the game, or could exit and disconnect from the game and return to the lobby.

### Return value

**NetErrorNone** If successful.  
**NetErrorBadMode** If not NetJoined.

### Example

N/A

### See also

[NetJoin\(\)](#)

**NetObjectField**

This function registers a field. Fields are used in the definition of a structure. The DME API allows an application to define its own structures that will be used to transport application specific information among clients.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_object.h	1.00	June 12, 2001

**Syntax**

```

NetErrorCode NetObjectField(
    int *FieldIndex,
    int FieldOffset,
    int ElementSize,
    int DataType,
    int ElementCount,
    NetTypeBroadcastSchedule *UpdateSchedule);

```

This function returns an integer identifying the registered field.  
 Byte offset of the structure element.  
 Size of the structure element.  
 Type of data represented in structure element.  
 Number of consecutive elements (i.e. array size).  
 Field specific broadcast schedule (when to propagate data to the network).

**Description**

This function registers a field. Fields are used in the definition of a structure. The DME API allows an application to define its own structures that will be used to transport application specific information among clients. In registering these fields, within the structures, the application has the ability to define the individual fields within the structure. The definition of individual fields results in a reduction of network traffic, because the DME API only needs to send fields that have been updated.

**Notes**

The macro is a more convenient method of registering object fields.

**Return value**

NetErrorNone: If successful.

**Example**

N/A

**See also**

[NetRegisterObjectField](#) macro.

## NetOpenDataStream

This function opens a data stream channel.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	June 18, 2004

### Syntax

**NetErrorCode** NetOpenDataStream(

<i>int *DataStreamChannel,</i>	The returned data stream channel just opened.
<i>int BufferSize,</i>	Size in bytes
<i>int DataRate,</i>	Bytes per second
<i>int StreamType,</i>	User defined (audio/video etc.)
<i>int RemoteBuffer,</i>	True or false based on whether a remote buffer is to be used.
<i>int TargetClientIndex,</i>	NET_SEND_TO_ALL_CLIENTS -> global broadcast
<i>int CircularBuffer,</i>	True or false based on if a circular buffer is to be used.
<i>unsigned int MinPacketSize,</i>	Minimum packet size
<i>unsigned int MaxPacketSize);</i>	Maximum packet size.

### Description

This function opens a data stream channel. You must specify the number of streams you want when you call NetJoin.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetEndDataStream\(\)](#)

NetPing

This function returns the packet latency to a target client or server in milliseconds.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	dme.h	1.03	June 18, 2004

Syntax

NetErrorCode
NetPing(
HDME
ConnectionHandle,
int
Target,
NetTypePingCallback
PingCallback);

Connection handle of the given connection.  
Target client index to send a ping to or NET\_SEND\_TO\_SERVER.  
Callback triggered when a response is received.

Description

This function returns the packet latency to a target client or server in milliseconds through callback that is provided. returns an error code of NetErrorTimedOut if no response is received.

Notes

N/A

Return value

NetErrorNone: If successful.

Example

N/A

See also

NetTypePingCallback, NetPingIP(), NetPingNetAddress()



## NetPingIP

This function returns the packet latency to a target IP address in milliseconds.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.03	August 15, 2001

### Syntax

```
NetErrorCode NetPingIP(  
    char *PingIpAddress,           Pointer to an IP address to which to send a ping.  
    NetTypePingCallback PingCallback);    Callback triggered when a response is received.
```

### Description

This function returns the packet latency to an IP address in milliseconds through a callback that is provided. It returns an error code of NetErrorTimedOut if no response is received.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

N/A

## NetPingNetAddress

This function returns the packet latency to a target DME [NetAddress](#) in milliseconds.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

```
NetErrorCode NetPingNetAddress(  
  NetAddress *pNetAddress,           Pointer to the DME NetAddress to which to send a  
                                       ping.  
  NetTypePingCallback PingCallback); Callback triggered when a response is received.
```

### Description

This function returns the packet latency to an [NetAddress](#) address in milliseconds through a callback that is provided. It returns an error code of NetErrorTimedOut if no response is received.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetAddress](#), [NetTypePingCallback](#), [NetPing\(\)](#), [NetPingIP\(\)](#)

# NetRegisterApplicationMessage

This function registers a DME application message.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	dme.h	1.00	June 12, 2001

## Syntax

**NetErrorCode** NetRegisterApplicationMessage(  
     **unsigned int** \*MessageType,  
  
     **NetTypeMessageParser** ThisMessageParser);

Pointer to the returned index used to send this type of message.  
  
 Pointer to the function called when a net message of this type is received.

## Description

This function registers a new network message with the DME. If successful, the returned MessageType is a unique identifier (ID). To send messages, call [NetSendApplicationMessage\(\)](#) with MessageType set to the ID returned (and saved off) from this function. When a network message is received, the callback ThisMessageParser is called based on this ID. . \* A network message only needs to be registered once during the lifetime of an online session. If [NetClose\(\)](#) is called, then the network message must be re-registered after calling [NetInitialize\(\)](#).

## Notes

This is the same as [NetRegisterMessage\(\)](#) with the message class automatically set to MessageClassApplication. It is recommended that all titles use [NetRegisterApplicationMessage\(\)](#) instead of [NetRegisterMessage\(\)](#).

## Return value

NetErrorNone: If successful.

## Example

N/A

## See also

[NetTypeMessageParser](#), [NetSendApplicationMessage\(\)](#), [NetSendAppMessage\(\)](#)

## NetRegisterDataStream

This function registers a data stream.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_object.h	1.00	September 12, 2001

### Syntax

<b>NetErrorCode</b> <b>NetRegisterDataStream</b> ( <b>int</b> *DataStreamType,  <b>NetTypeDataStreamUpdateCallback</b> DataStreamUpdateCallback, <b>NetTypeDataStreamEndCallback</b> DataStreamEndCallback, <b>NetTypeDataStreamFilterCallback</b> DataStreamFilterCallBack);	Pointer to the returned data stream unique identifier.  Called when data is received for a given data stream.  Called when the data stream is finished sending data.  Called to determine which clients should be sent the data stream.
---	---

### Description

This function registers a data stream update callback and filter handlers. An update callback is issued on remote data stream creation and for each subsequent update. An end is issued when remote data stream is closed. Filter function callback is called for each local update sent for each target client. If DataStreamFilterCallBack is NULL, then the data stream is not filtered.

### Notes

DataStreamUpdateCallback and DataStreamEndCallback can be NULL. However, if the data stream is non-buffered, then all remote updates must be handled.

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetTypeDataStreamUpdateCallback](#), [NetTypeDataStreamEndCallback](#), [NetTypeDataStreamFilterCallBack](#)

# NetRegisterMemoryCallbacks

This function registers callback functions for application controlled memory management.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	June 12, 2001

## Syntax

**NetErrorCode** NetRegisterMemoryCallbacks(  
**NetMemoryCallbackParams** \*pCallbackParams);      Pointer to memory callback information.

## Description

This function registers callback functions for application controlled memory management. This allows the client application's memory manager to manage all malloc(), realloc(), and free() calls within the SCE-RT libraries.

## Notes

N/A

## Return value

NetErrorNone: If successful.

## Example

N/A

## See also

[NetMemoryCallbackParams](#), [NetSetDefaultMemoryCallbackParams\(\)](#)

NetRegisterMessage

This function registers a new network message with the DME.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	dme.h	1.00	September 12, 2001

Syntax

<b>NetErrorCode</b> NetRegisterMessage( <b>unsigned int</b> *MessageType,  <b>NetMessageClass</b> MessageClass, <b>NetTypeMessageParser</b> ThisMessageParser);	Pointer to the returned index used to send messages of this type.  Class of message. Pointer to function called when a net message of this type is received.
---	---

Description

This function registers a new network message with the DME. If successful, the returned MessageType is a unique identifier (ID). To send messages, call [NetSendMessage\(\)](#) with MessageType set to the ID returned (and saved off) from this function. When a network message is received, the callback ThisMessageParser is called based on this ID.

A network message only needs to be registered once during the lifetime of an online session. If [NetClose\(\)](#) is called, then network message must be re-registered after calling [NetInitialize\(\)](#).

Notes

N/A

Return value

NetErrorNone: If successful.

Example

N/A

See also

[NetMessageClass](#), [NetTypeMessageParser](#), [NetRegisterApplicationMessage\(\)](#), [NetSendMessage\(\)](#)

## NetRegisterObjectFilter

This function registers a network object (NetObject) filter function.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	November 11, 2004

### Syntax

```
NetErrorCode NetRegisterObjectFilter(
NetRegisterObjectFilterInParams *pObjFilterInParams,    Pointer to input parameters.
NetRegisterObjectFilterOutParams *pObjFilterOutParams);  Pointer to output parameters.
```

### Description

This function registers a network object (NetObject) filter function. The client executes this function whenever the given network object's state has changed. The filter function is used to determine whether remote clients should receive updates with the new state information.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetRegisterObjectFilterInParams](#), [NetRegisterObjectFilterOutParams](#),  
[NetSetDefaultRegisterObjectFilterParams\(\)](#), [NetCreateObject\(\)](#), [NetRequestCreateRemoteNamedObject\(\)](#)

## NetRegisterObjectStart

This function starts the registration process of a network object.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	June 12, 2001

### Syntax

```
NetErrorCode NetRegisterObjectStart(
    void );
```

### Description

This function starts the registration process of a network object. It must precede calls to `NetRegisterObjectField`. Calling `NetRegisterStructure()` will end the registration process. It should be called after all of the fields of a network object have been declared. `NetRegisterRemoteObjectCallback` is used to register callbacks for the given network object.

### Notes

N/A

### Return value

`NetErrorNone`: If successful.

### Example

N/A

### See also

`NetRegisterObjectField`, `NetObjectField()`, `NetRegisterStructure()`, `NetRegisterRemoteObjectCallback`, `NetCreateObject()`



# NetRegisterRemoteObjectCallback

This function registers remote net object update callbacks.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_object.h	1.00	October 31, 2001

## Syntax

```

NetErrorCode NetRegisterRemoteObjectCallback(
    int NetStructureIndex,
    NetTypeObjectCallback ObjectCreationCallback,
    NetTypeObjectUpdateCallback ObjectUpdateCallback,
    NetTypeObjectCallback ObjectDeletionCallback);

```

Object type that is returned.

Register callbacks for this network object that has this NetStructureIndex (returned by [NetRegisterStructure\(\)](#)).

Callback triggered when a remote network object is created.

Callback triggered when a remote network object is deleted.

## Description

This function registers callback functions for handling remote network object updates. You can register a set of callbacks for each NetStructureIndex.

## Notes

Any callback function pointer that is NULL implies that the application does not need to process the event.

## Return value

NetErrorNone: If successful.

## Example

N/A

## See also

[NetRegisterObjectStart\(\)](#), [NetRegisterObjectField](#), [NetObjectField\(\)](#), [NetRegisterStructure\(\)](#), [NetCreateObject\(\)](#)

NetRegisterStructure

This function finishes the registration of a structure used by network objects.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_object.h	1.00	June 12, 2001

Syntax

<b>NetErrorCode</b> NetRegisterStructure(	
int *StructureIndex,	Pointer to the returned unique identifier that represents this registered network object..
const char *StructureName,	Associates a string name to the network object.
int StructureSize);	Total size of this structure (in bytes).

Description

This function registers a structure used by network objects. It must follow [NetRegisterObjectStart\(\)](#) and call to [NetRegisterObjectField\(\)](#). Structures are used in the definition of a NetObject. First, fields within the structure should be registered individually with the NetRegisterObjectField. After all of the appropriate fields within the structure are registered, call NetRegisterStructure.

Notes

N/A

Return value

NetErrorNone: If successful.

Example

N/A

See also

[NetRegisterObjectField](#), [NetCreateObject](#), [NetRegisterObjectStart\(\)](#), [NetRegisterObjectField](#), [NetObjectField\(\)](#), [NetRegisterRemoteObjectCallback](#), [NetCreateObject\(\)](#)

## NetReleaseObjectPrivateOwnership

This function releases private ownership of a network object.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	May 31, 2001

### Syntax

**NetErrorCode** NetReleaseObjectPrivateOwnership(

int *ObjectIndex*);

Network object to be released from private ownership.

### Description

This function releases private ownership of an object. The client who created the object sets the ownership of the object to be shared.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetRequestObjectPrivateOwnership\(\)](#)

NetRequestCreateRemoteNamedObject

This function requests that a remote client create a named network object.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_object.h	1.00	May 31, 2001

Syntax

<b>NetErrorCode</b> NetRequestCreateRemoteNamedObject( int *ObjectIndex,	Pointer to the returned unique identifier that represents this newly created network object (create callbacks on remote clients will be called and will have the same ObjectIndex).
<b>NetOwnershipStatus</b> Ownership,	NetObject ownership status: None, Private, Shared, etc.
<b>NetObjectLifespan</b> LifespanType, unsigned int LifeSpan,	NetObject life span type: Session, Permanent, etc. LifeSpan in millisecs.
int FilterMethod,	Filter index obtained from the NetRegisterObjectFilter.
unsigned int MaxUpdateInterval,	Maximum update interval in millisecs.
int LatencyCritical,	True or false
void *ObjectData,	Local network object data
char *ObjectName,	Network object name
int NetStructureIndex,	The identifier of the network object's structure (type to create).
int CreatorClientIndex);	Client index of the remote client that is being requested to create this network object.

Description

This function requests a remote client to create a named network object. If an object with the same name already exists, this function returns it as an identifier.

Notes

N/A

Return value

NetErrorNone: If successful.

Example

N/A

See also

NetOwnershipStatus, NetObjectLifespan, NetCreateObject()

# NetRequestObjectPrivateOwnership

This function requests private ownership of a network object.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	December 10, 2001

## Syntax

**NetErrorCode** NetRequestObjectPrivateOwnership(  
     int *ObjectIndex*);

Network object that is being requested for private ownership.

## Description

This function requests private ownership of a network object. The request goes to the client that created the given network object. If the creator sees that the network object is currently in a "shared" state, then the creator allows the requesting client to have private ownership. If the creator sees that the given network object is already privately owned by another client, then it notifies the requesting client that the network object is not currently available for private ownership.

## Notes

N/A

## Return value

NetErrorNone: If successful.

## Example

N/A

## See also

[NetReleaseObjectPrivateOwnership\(\)](#)

**NetResolveAddr**

This function resolves both the internal and external address of this client. Returns Network Address Resolution (NAT) information.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	dme.h	1.00	June 12, 2001

**Syntax**

```

NetErrorCode NetResolveAddr(
    const NetResolveAddrInParams *pInParams,           Pointer to input parameters
    NetResolveAddrOutParams *pOutParams);              Pointer to output parameters
    
```

**Description**

This function resolves both the internal and external address of this client. It returns Network Address Resolution (NAT) information.

**Notes**

N/A

**Return value**

NetErrorNone: If successful.

**Example**

N/A

**See also**

[NetResolveAddrInParams](#), [NetResolveAddrOutParams](#), [NetSetDefaultResolveAddrParams\(\)](#), [NetTypeResolveAddrCallback](#)

# NetSendApplicationMessage

This function sends an application defined network message.

Link to file	Include file	Introduced	Last modified
<a href="#">librtbasePS2.a</a>	<a href="#">dme.h</a>	1.00	June 12, 2001

## Syntax

<b>NetErrorCode</b> <b>NetSendApplicationMessage</b> (	
<b>char</b> <i>TransportFlags</i> ,	NET_DELIVERY_CRITICAL, NET_LATENCY_CRITICAL, etc.
<b>HDME</b> <i>ConnectionHandle</i> ,	Connection handle of the given connection.
<b>int</b> <i>TargetClientIndex</i> ,	Client index of client to receive message (or a special target flag - broadcast to all clients).
<b>int</b> <i>MessageType</i> ,	This is the registered message type from <a href="#">NetRegisterApplicationMessage()</a> .
<b>int</b> <i>MessageLength</i> ,	Size of the message in bytes.
<b>unsigned char</b> * <i>MessageData</i> );	Pointer to the body of the message.

## Description

This function sends a network message (NetMessage) to a specified target ClientIndex. Many types of transport flags can be set (delivery critical flag, latency critical flag, message in-order flag, etc). You can call this function any time after the network message has been registered with the DME client library. A client must be connected successfully to either a DME game server or a peer-to-peer host for this send to succeed. It does not matter whether a client has called [NetJoin\(\)](#) or [NetLeave\(\)](#); as long as they are connected, they will be able to send messages.

## Notes

When sending network messages, it is recommended that all titles register NetMessages using the [NetRegisterApplicationMessage\(\)](#) function, and then send messages using the [NetSendAppMessage\(\)](#) function.

Soon, [NetSendApplicationMessage\(\)](#) will be deprecated in favor of [NetSendAppMessage\(\)](#).

## Return value

NetErrorNone: If successful.

## Example

N/A

## See also

[NetRegisterApplicationMessage\(\)](#), [NetSendAppMessage\(\)](#)

NetSendMessage

This function sends an application defined network message.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	dme.h	2.09	November 11, 2004

Syntax

```

NetErrorCode NetSendMessage(
    NetSendMessageInParams *pAppMessageInParams,    Pointer to input parameters.
    NetSendMessageOutParams *pAppMessageOutParams);  Pointer to output parameters.

```

Description

This function sends an application defined network message according to the parameters defined by the [NetSendMessageInParams](#) structure. This function sends a network message (NetMessage) to the specified target ClientIndex or list of ClientIndex entries (in the [NetSendMessageInParams](#) structure). Many types of transport flags can be set (delivery critical flag, latency critical flag, message in-order flag, etc.). You can call this function any time after the network message has been registered with the DME client library. After a client successfully connects to a DME game server, a peer-to-peer host Join(), or [NetLeave\(\)](#), it is able to send messages as long as it remains connected.

Notes

When sending network messages, it is recommended that all titles register NetMessages with the [NetRegisterApplicationMessage\(\)](#) function, and send messages with the [NetSendMessage\(\)](#) function. [NetSendApplicationMessage\(\)](#) will soon be deprecated in favor of using [NetSendMessage\(\)](#).

Return value

NetErrorNone: If successful.

Example

N/A

See also

[NetRegisterApplicationMessage\(\)](#), [NetSendMessageInParams](#), [NetSendMessageOutParams](#), [NetSetDefaultAppMessageParams\(\)](#)



## NetSendFieldUpdates

This function forces a send of field updates for a given network object.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	2.09	November 11, 2004

### Syntax

```
NetErrorCode NetSendFieldUpdates(
    int ObjectIndex);
```

Network object to update.

### Description

This function forces a send of field updates for a given network object. When the application calls [NetSetFieldChanged\(\)](#) (which marks particular fields in a network object to update themselves regardless of its error threshold function or its broadcast schedule), the application can force the send with the `NetSendFieldUpdate()` function. The `NetSendFieldUpdate()` function is a lighter weight version of a [NetUpdate\(\)](#) call. It can be helpful during debugging sessions when one is trying to track individual field changes and their behavior.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetSetFieldChanged\(\)](#)

## NetSendMessage

This function sends a message to target clients or to the server.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	dme.h	1.00	June 12, 2001

### Syntax

```
NetErrorCode NetSendMessage(  
    char TransportFlags,                                NET_DELIVERY_CRITICAL,  
                                                    NET_LATENCY_CRITICAL, etc.  
    HDME ConnectionHandle,                             Connection handle of the given connection.  
    int TargetClientIndex,                             Client index of client to receive message (or a  
                                                    special target flag - broadcast to all clients).  
    int MessageClass,                                  Class of message (NetMessageClass).  
    int MessageType,                                  From NetRegisterApplicationMessage(). This is the  
                                                    registered message type.  
    int MessageLength,                                 Size of message in bytes.  
    unsigned char *MessageData);                       Pointer to message data.
```

### Description

This function sends a network message (NetMessage) to a specified target. Many types of transport flags can be set (delivery critical flag, latency critical flag, message in-order flag, etc.). You can call this function at any time after the network message has been registered with the DME client library. A client must be connected to either a DME game server or a peer-to-peer host in order for this send to succeed. It does not matter if a client has called [NetJoin\(\)](#) or [NetLeave\(\)](#); as long as it is connected it is able to send messages.

### Notes

When sending network messages, it is recommended that all titles register NetMessages with the [NetRegisterApplicationMessage\(\)](#) function, and to send messages with the [NetSendAppMessage\(\)](#) function.

[NetSendMessage\(\)](#) is typically reserved for SCE-RT client library functionality (i.e., Medius and MGCL client libraries), while [NetSendAppMessage\(\)](#) is used by the title application.

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetRegisterMessage\(\)](#), [NetSendAppMessage\(\)](#)

## NetSendMyClientUpdate

This function sends my client information.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	June 18, 2004

### Syntax

```
NetErrorCode NetSendMyClientUpdate(
HDME ConnectionHandle,           Connection handle of the given connection.
int TargetClientIndex);         Client index of client to receive message (or a
                                   special target flag - broadcast to all clients).
```

### Description

This function will send a full update for my client to the TargetClientIndex associated with the ConnectionHandle.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetTypeClient](#), [NetGetClient\(\)](#)

**NetSendObjectFullUpdate**

This function transmits every registered field in the object's structure based on the ObjectIndex to theTarget specified by TargetClientIndex.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_object.h	1.00	May 31, 2001

**Syntax**

```

NetErrorCode NetSendObjectFullUpdate(
    int TargetClientIndex,                                Client index of client to receive message (or a
                                                         special target flag
                                                         broadcast to all clients).

    int ObjectIndex);                                    Network object to send full update for.
    
```

**Description**

This function transmits every registered field in the object's structure based on the ObjectIndex to theTarget specified by TargetClientIndex.

**Notes**

If you have an ObjectFilter Callback registered with the DME, it will be invoked for determining whether to send the update.

**Return value**

NetErrorNone: If successful.

**Example**

N/A

**See also**

N/A

## NetSetDefaultAppMessageParams

This function sets default values for [NetSendMessageInParams](#).

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	dme.h	2.09	November 11, 2004

### Syntax

```
NetErrorCode NetSetDefaultAppMessageParams(  
  NetSendMessageInParams *pAppMessageInParams);
```

Pointer to the input parameter for which to set defaults.

### Description

This function sets the default values (initializes) the NetSendMessageInParams structure that is used with a [NetSendAppMessage\(\)](#) call.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetSendMessageInParams](#), [NetSendAppMessage\(\)](#)

## NetSetDefaultBitMask

This function sets the default DME defined bitmask.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

**NetErrorCode** NetSetDefaultBitMask(  
**NetBitMask** \*pBitMask);

Pointer to the returned bitmask with default values set.

### Description

This function will zero out the given bitmask structure and sets max\_id to NET\_CLIENT\_LIMIT.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetBitMask](#), [NetBitMaskIsSet\(\)](#), [NetBitMaskSet\(\)](#), [NetBitMaskUnSet\(\)](#)

## NetSetDefaultConnectParams

This function sets default values for [NetConnectInParams](#).

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	dme.h	1.18	March 13, 2002

### Syntax

```
NetErrorCode NetSetDefaultConnectParams(  
  NetConnectInParams *pParams);
```

Pointer to input parameters for which to set defaults.

### Description

This function initializes the input parameters that are used in [NetConnect\(\)](#) with default values.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetConnectInParams](#), [NetConnect\(\)](#)

## NetSetDefaultDisconnectParams

This function sets default values for [NetDisconnectParams](#).

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

```
NetErrorCode NetSetDefaultDisconnectParams(  
  NetDisconnectParams *pDisconnectParams);
```

Pointer to the input parameters for which to set defaults.

### Description

This function initializes the input parameters that are used in [NetDisconnect\(\)](#) with default values.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetDisconnectParams](#), [NetDisconnect\(\)](#)



## NetSetDefaultHostPeerToPeerParams

This function sets default values for [NetHostPeerToPeerInParams](#).

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	dme.h	1.18	March 13, 2002

### Syntax

```
NetErrorCode NetSetDefaultHostPeerToPeerParams(  
  NetHostPeerToPeerInParams *pParams);
```

Pointer to input parameters for which to set defaults.

### Description

This function initializes the input parameters that are used in [NetHostPeerToPeer\(\)](#) with default values.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetHostPeerToPeerInParams](#), [NetHostPeerToPeer\(\)](#)

## NetSetDefaultIncomingClientParams

This function sets default values for [NetIncomingClientInParams](#).

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	dme.h	2.09	November 11, 2004

### Syntax

```
NetErrorCode NetSetDefaultIncomingClientParams(  
  NetIncomingClientInParams *pParams);
```

Pointer to input parameters for which to set defaults.

### Description

This function initializes the input parameters that are used in [NetIncomingClient\(\)](#) with default values.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

N/A

## NetSetDefaultInitializeParams

This function sets default values for [NetInitializeInParams](#).

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	dme.h	1.18	March 13, 2002

### Syntax

```
NetErrorCode NetSetDefaultInitializeParams(  
  NetInitializeInParams *pParams);
```

Pointer to input parameters for which to set defaults.

### Description

This function initializes the input parameters that are used in [NetInitialize\(\)](#) with default values.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetInitializeInParams](#), [NetInitialize\(\)](#)

NetSetDefaultJoinParams

This function sets default values for [NetJoinInParams](#).

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_object.h	1.24	December 1, 20023.

Syntax

**NetErrorCode** NetSetDefaultJoinParams(  
[NetJoinInParams](#) \*pParams);

Pointer to input parameters for which defaults are to be set.

Description

This function initializes the input parameters that are used in [NetJoin\(\)](#) with default values.

Notes

N/A

Return value

NetErrorNone: If successful.

Example

N/A

See also

[NetJoinInParams](#), [NetJoin\(\)](#)

## NetSetDefaultLANFindParams

This function sets default values for [NetLANFindInParams](#).

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	<a href="#">rt_lanfind.h</a>	2.09	November 11, 2004

### Syntax

```
NetErrorCode NetSetDefaultLANFindParams(  
  NetLANFindInParams *pParams);
```

Pointer to input parameters for which to set defaults.

### Description

This function initializes the input parameters that are used in [NetLANFind\(\)](#) with default values.

Default values are to set: 1) Locate NetSessionTypeGame peers. 2) Locate peers of the same ApplicationID  
3) Locate peers of the same PlatformID.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetLANFindInParams](#), [NetLANFind\(\)](#)

**NetSetDefaultLatencyMetricsParams**

This function sets default values for [NetLatencyMetricsParams](#).

Link to file	Include file	Introduced	Last modified
<a href="#">librtbasePS2.a</a>	dme.h	2.09	November 11, 2004

**Syntax**

**NetErrorCode**
**NetSetDefaultLatencyMetricsParams**(  
[NetLatencyMetricsParams](#) \*pParams);

Pointer to input parameter for which to set defaults.

**Description**

This function initializes the input parameters that are used in [NetGetLatencyMetrics\(\)](#) with default values.

**Notes**

N/A

**Return value**

NetErrorNone: If successful.

**Example**

N/A

**See also**

[NetLatencyMetricsParams](#), [NetGetLatencyMetrics\(\)](#)

## NetSetDefaultLookupParams

This function sets default values for [NetTypeLookupParams](#).

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

```
NetErrorCode NetSetDefaultLookupParams(  
  NetTypeLookupParams *pLookupParams);
```

Pointer to input parameter for which to set defaults.

### Description

This function initializes the input parameters that are used in [NetGetHostByName\(\)](#) with default values.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetTypeLookupParams](#), [NetGetHostByName\(\)](#)

**NetSetDefaultMemoryCallbackParams**

This function sets default values for [NetMemoryCallbackParams](#).

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	dme.h	1.00	November 11, 2004

**Syntax**

**NetErrorCode**
**NetSetDefaultMemoryCallbackParams**(
  
**NetMemoryCallbackParams** \*pCallbackParams);

Pointer to input parameters for which to set defaults.

**Description**

This function initializes the input parameters that are used in [NetRegisterMemoryCallbacks\(\)](#) with default values.

**Notes**

N/A

**Return value**

NetErrorNone: If successful.

**Example**

N/A

**See also**

[NetMemoryCallbackParams](#), [NetRegisterMemoryCallbacks\(\)](#)



## NetSetDefaultRegisterObjectFilterParams

This function sets default values for [NetRegisterObjectFilterInParams](#).

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	November 11, 2004

### Syntax

[NetErrorCode](#)

**NetSetDefaultRegisterObjectFilterParams**(

[NetRegisterObjectFilterInParams](#) \*pObjFilterInParams);      Pointer to input parameters for which to set defaults.

### Description

This function initializes the input parameters that are used in [NetRegisterObjectFilter\(\)](#) with default values.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetRegisterObjectFilterInParams](#), [NetRegisterObjectFilter\(\)](#)

## NetSetDefaultResolveAddrParams

This function Sets default values for [NetResolveAddrInParams](#).

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	dme.h	1.00	November 11, 2004

### Syntax

```
NetErrorCode NetSetDefaultResolveAddrParams(  
  NetResolveAddrInParams *pParams);
```

Pointer to input parameters for which to set defaults.

### Description

This function initializes the input parameters that are used in [NetResolveAddr\(\)](#) with default values.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetResolveAddrInParams](#), [NetResolveAddr\(\)](#)

## NetSetDefaultStreamMediaParams

This function sets default values for [NetStreamMediaParams](#).

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	<a href="#">rt_media.h</a>	1.24	November 11, 2004

### Syntax

```
NetErrorCode NetSetDefaultStreamMediaParams(  
  NetStreamMediaParams *pParams);
```

Pointer to input parameters for which to set defaults.

### Description

This function initializes the [NetStreamMediaParams](#) field of either [NetConnectInParams](#) or [NetHostPeerToPeerInParams](#) depending if [NetConnect\(\)](#) is to be called or [NetHostPeerToPeer\(\)](#) is being called.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetStreamMediaParams](#), [NetConnectInParams](#), [NetHostPeerToPeerInParams](#)

## NetSetFieldChanged

This function sets a flag indicating that a network object field has changed.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.32	December 8, 2003

### Syntax

**NetErrorCode** NetSetFieldChanged(

int *ObjectIndex*,

Object to update

int *FieldIndex*);

Field to mark for update

### Description

This function marks a field as changed in a particular network object to force an update to be propagated to all other clients. This function marks the field in a certain way so that it bypasses the network object's error threshold callback and ignores the field's broadcast schedule for that frame.

When a field is marked as changed (whether or not the field actually did change) it is sent to the other peers during the next [NetUpdate\(\)](#) call. If the client application needs to send the field change immediately, then the client should call [NetSendFieldUpdates\(\)](#), which is just a light weight version of [NetUpdate\(\)](#), which just sends network object field updated or changed information.

### Notes

Most applications just rely on network object field error threshold callbacks and broadcast schedules; however, this function (when used with [NetSetFieldChanged\(\)](#)) is provided as a convenience; nonetheless, it is a helpful tool to change network object field data immediately and to have it sent to all remote clients.

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

None.

## NetSetMyClientObject

This function sets a network object as the client's primary network object.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_object.h	1.00	May 31, 2001

### Syntax

```
NetErrorCode NetSetMyClientObject(
    int ObjectIndex);
```

The object ID of the network object that one wants to tightly associate with this client.

### Description

This function sets a particular network object that a client created as its primary object (Avatar). Clients can only have one network object flagged as their Avatar at a time.

Calling [NetGetClient\(\)](#) returns a [NetTypeClient](#) structure. This structure has a field called "ClientObjectIndex", from which the network object ID is set by a client's call to [NetSetMyClientObject\(\)](#). (This is really the only logic this function provides).

[NetSetMyClientObject\(\)](#) does not need to be called for every client application. However, it does provide an easy and quick way to associate which network object best represents a given client if the client application needs to have that information readily available.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetTypeClient](#), [NetGetClient\(\)](#)

## NetSetMyClientReceiveBroadcast

This function enables the reception per ConnectionHandle of client data sent via NET\_SEND\_TO\_ALL\_CLIENTS by other clients.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.00	June 18, 2004

### Syntax

**NetErrorCode** NetSetMyClientReceiveBroadcast(

**HDMF** *ConnectionHandle*,

**int** *EnableFlag*);

Connection handle of the given connection.

If set to True, then this client will begin receiving broadcast information.

If set to False, then this client will not receive broadcast information.

### Description

This function enables the reception (per ConnectionHandle) of client data sent via NET\_SEND\_TO\_ALL\_CLIENTS by other clients. This call is explicitly done at the end of [NetJoin\(\)](#). This method of receiving data prevents a client from receiving excessive amounts of data until the client is ready to regularly start calling [NetUpdate\(\)](#).

As a particular example, when clients are joining games late, this function helps throttle received network object update information. Clients can be all connected, but the incoming client is still loading lots of information off the DVD and is not calling [NetUpdate\(\)](#) every frame. The client may still call [NetJoin\(\)](#) (because it needs more information about other clients and about which network objects are currently active); however, since the client is still loading game resources off the DVD, we must disable network object update callbacks. To do this, call NetSetMyClientReceiveBroadcast() with the disable flag, then the client will get all the remote network object create callbacks with their initial data (needed to tie together resources from DVD), but will not receive any further updates from network objects until NetSetMyClientReceiveBroadcast() is called with the enabled flag.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetJoin\(\)](#)

## NetSetNATServiceAddr

This function sets the NAT service address used for resolving addresses. This function is reserved for use by the DME. A client application does not need to call this function.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	dme.h	2.09	November 11, 2004

### Syntax

**NetErrorCode** NetSetNATServiceAddr(

const **NetAddress** \*pAddress);

Pointer to input parameter with NAT service address information.

### Description

This function sets the NAT service address used for resolving network addresses.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetAddress](#), [NetResolveAddr\(\)](#)

## NetSetSendAggregationInterval

This function sets the send buffer aggregation interval of the DME.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	1.03	June 18, 2004

### Syntax

**NetErrorCode** NetSetSendAggregationInterval(

**HDME** *ConnectionHandle*,

**unsigned int** *ClientTimeMSecs*,

**unsigned int** *ServerTimeMSecs*);

Connection handle of the given connection.

Number of milliseconds between sends (to clients).

Number of milliseconds between sends (to servers).

### Description

This function sets the number of milliseconds waiting between sends for a given connection. The default value for both the client and server is set to 30 milliseconds.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetInitialize\(\)](#)



## NetStreamMediaEndRecording

This function completes any recording currently in progress.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_media.h	1.24	June 18, 2004

### Syntax

```
NetErrorCode NetStreamMediaEndRecording(  
    HDME ConnectionHandle,           Connection handle of the given connection.  
    NetStreamMediaAudioType AudioType);    Selects which audio codec is currently enabled.
```

### Description

This function completes any recording currently in progress.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetStreamMediaAudioType](#), [NetStreamMediaGetCurrentChannelState\(\)](#)

## NetStreamMediaGetChannelInfo

This function gets stream media channel information.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	rt_media.h	1.24	December 1, 2002

### Syntax

**NetErrorCode** NetStreamMediaGetChannelInfo(  
**HDME** *ConnectionHandle*,  
**unsigned int** *ChannelNum*,  
**NetStreamMediaChannelInfo** \**pChannelInfo*);

Connection handle of the given connection.  
Stream media channel for which to get information.  
Pointer to the returned stream media channel information.

### Description

This function gets stream media channel information about a specific stream media channel.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetStreamMediaChannelInfo](#)

## NetStreamMediaGetClientInfo

This function gets stream media information about a specific client.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_media.h	1.24	June 18, 2004

### Syntax

```
NetErrorCode NetStreamMediaGetClientInfo(
    HDMF ConnectionHandle,           Connection handle of the given connection.
    int ClientIndex,                 Client index for which to get stream media
                                      information.
    NetStreamMediaClientInfo *pClientInfo); The returned client information.
```

### Description

This function gets stream media client information.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetStreamMediaClientInfo](#)

## NetStreamMediaGetCurrentChannelState

This function gets current stream media channel state information.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_media.h	1.24	June 18, 2004

### Syntax

[NetErrorCode](#)

**NetStreamMediaGetCurrentChannelState**(

[HDME](#) *ConnectionHandle*,

[NetStreamMediaChannelStateData](#) \**pStateData*);

Connection handle of the given connection.

Pointer to the returned stream media channel state information.

### Description

This function retrieves state data on the currently joined channel. To determine when a player can start talking on their audio headset, the application calls [NetStreamMediaGetCurrentChannelState\(\)](#). If *pStateData->bCanRecord* is set to True, then the player's audio data is propagated on the network to other clients.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetStreamMediaChannelStateData](#), [NetStreamMediaEndRecording\(\)](#)

## NetStreamMediaJoinChannel

Joins the specified stream media channel.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_media.h	1.24	June 18, 2002

### Syntax

```
NetErrorCode NetStreamMediaJoinChannel(  

HDME ConnectionHandle,           Connection handle of the given connection.  

unsigned int ChannelNum);        Stream media channel to join.
```

### Description

Joins the specified channel. If it is currently in a different channel, the client is first removed from the current channel.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetStreamMediaQuitChannel\(\)](#)

## NetStreamMediaQuitChannel

This function quits the specified channel.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	rt_media.h	1.24	June 18, 2004

### Syntax

```
NetErrorCode NetStreamMediaQuitChannel(  

HDME ConnectionHandle,           Connection handle of the given connection.  

unsigned int ChannelNum);        Stream media channel to quit.
```

### Description

This function quits the specified channel.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetStreamMediaJoinChannel\(\)](#)

## NetStreamMediaSetDefaultIgnoreParams

This function sets default values for [NetStreamMediaIgnoreData](#).

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	<a href="#">rt_media.h</a>	1.31	December 8, 2003

### Syntax

#### NetErrorCode

**NetStreamMediaSetDefaultIgnoreParams**(

[NetStreamMediaIgnoreData](#) \*pIgnoreData);

Pointer to input parameter for which to set defaults.

### Description

This function initializes the input parameters that are used in [NetStreamMediaSetIgnoreState\(\)](#) with default values.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetStreamMediaIgnoreData](#), [NetStreamMediaSetIgnoreState\(\)](#)

## NetStreamMediaSetIgnoreState

This function sets stream media ignore information.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	<a href="#">rt_media.h</a>	1.31	December 8, 2003

### Syntax

```
NetErrorCode NetStreamMediaSetIgnoreState(  
    NetStreamMediaIgnoreData *pIgnoreData);    Pointer to
```

### Description

This function sets stream media ignore information for a particular client on the specified connection.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetStreamMediaIgnoreData](#), [NetStreamMediaSetDefaultIgnoreParams\(\)](#)



## NetTick

This function is a light weight version of [NetUpdate\(\)](#).

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
<a href="#">librtbasePS2.a</a>	dme.h	1.18	March 13, 2002

### Syntax

```
NetErrorCode NetTick(  
void );
```

### Description

This performs all services provided by [NetUpdate\(\)](#) but does not issue callbacks for received messages.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetUpdate\(\)](#)

**NetTokenListRelease**

Release the ownership of a list of tokens.

Link to file	Include file	Introduced	Last modified
librtbasePS2.a	dme.h	2.09	November 11, 2004

**Syntax**

```

NetErrorCode NetTokenListRelease(
    HDME pConnInfo,                Connection handle of the given connection.
    const NetBitMask *pInBitMask,  Pointer to an input parameter bitmask.
    NetBitMask *pOutBitMask);      Pointer to an output parameter bitmask.
    
```

**Description**

Release the ownership of a list of tokens.

**Notes**

N/A

**Return value**

NetErrorNone: If successful.

**Example**

N/A

**See also**

[NetBitMask](#), [NetTokenListRequest\(\)](#)

## NetTokenListRequest

This function requests the ownership of a list of tokens.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

```
NetErrorCode NetTokenListRequest(
    HDME pConnInfo,                Connection handle of the given connection.
    const NetBitMask *pInBitMask,   Pointer to an input parameter bitmask.
    NetBitMask *pOutBitMask);       Pointer to an output parameter bitmask.
```

### Description

This function requests the ownership of a list of tokens.

### Notes

If a partial or whole list of tokens are granted, the sender gets a granted message in its TokenOwnershipNotifyCallback. Otherwise, the sender gets an owned message about all of the owners of the tokens that were requested in the same callback.

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetBitMask](#), [NetTokenListRelease\(\)](#)

## NetTokenQuery

This function queries the ownership of a token.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

```
NetErrorCode NetTokenQuery(
HDME ConnectionHandle,           Connection handle of the given connection.
int TokenID,                     Token ID that is being queried about its ownership
unsigned int *pOwnerClientIndex); Pointer to the owner ID of the token
```

### Description

This function queries the ownership of a token.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetTokenSystemQuery\(\)](#)

## NetTokenRelease

This function releases the ownership of a token.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

**NetErrorCode** NetTokenRelease(

**HDME** *ConnectionHandle*,

int *TokenID*);

Connection handle of the given connection.

Token ID that is to have its ownership released.

### Description

This function releases the ownership of a token.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetTokenRequest\(\)](#)

## NetTokenRequest

This function requests the ownership of a token.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

**NetErrorCode** NetTokenRequest(

**HDME** *ConnectionHandle*,

int *TokenID*);

Connection handle of the given connection.

Token ID to request ownership of.

### Description

This function requests the ownership of a token.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetTokenRelease\(\)](#)

## NetTokenSystemQuery

This function queries the status of the token system.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

[NetErrorCode](#) NetTokenSystemQuery(

[HDME](#) *ConnectionHandle*,

int *\*pSystemReady*);

Connection handle of the given connection.

Status of the token system: 0 = Not Ready, 1 = Ready

### Description

This function queries the status of the token system.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetTokenQuery\(\)](#)

## NetUpdate

This function maintains client connections.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	June 12, 2001

### Syntax

```
NetErrorCode NetUpdate(  
void );
```

### Description

This function maintains client connection information. Internally, it triggers a number of network related tasks (processes network messages, adjusts the global timebase, etc.). From the application's perspective, it refreshes the object state data, creates new objects, deletes old objects, etc. The application must call this function periodically to ensure that the DME API gets a share of the processor for networking activity. The frequency at which this function must be called varies from application to application, but should occur at least once per second. The longer the interval between NetUpdates, the more you will notice network latency in your application.

### Notes

N/A

### Return value

NetErrorNone: If successful.

Use NetGetUpdateErrors to get extended error information.

### Example

N/A

### See also

[NetTick\(\)](#)



## NetUseACodecGSM

This function declares which audio codec to use.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

```
void NetUseACodecGSM();
```

### Description

This function lets the library specify which audio codec to use. It calls the export function, which copies the interface functions to the interface function pointers, causing the linker to link in the interface. Also, this function initializes data members of the interface structure

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetUseDme\(\)](#)

## NetUseACodecLPC

This function declares which audio codec to use.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

```
void NetUseACodecLPC();
```

### Description

This function lets the library specify which audio codec to use. It calls the export function, which copies the interface functions to the interface function pointers, causing the linker to link in the interface. Also, this function initializes data members of the interface structure.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetUseDme\(\)](#)

## NetUseACodecLPC10

This function declares which audio codec to use.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

```
void NetUseACodecLPC10();
```

### Description

This function lets the library specify which audio codec to use. It calls the export function, which copies the interface functions to the interface function pointers, causing the linker to link in the interface. Also this function initializes data members of the interface structure

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetUseDme\(\)](#)

## NetUseClientServer

This function declares that the application will be using client server.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

```
void NetUseClientServer();
```

### Description

This function declares that the application will use client server communication. Therefore, it lets the library use the DME MsgClient layer (needed for client server communication). This function calls the export function of msg client, which copies the interface functions to the interface function pointers, causing the linker to link in the interface.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetUseDme\(\)](#)

## NetUseCommAdhoc

This function declares that the application will use adhoc mode.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

```
void NetUseCommAdhoc();
```

### Description

This function declares that the application will use adhoc mode (currently only available on the PSP); therefore, enables the DME comm layer to support adhoc mode.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetUseDme\(\)](#)

## NetUseCommEenet

This function declares that this application will use eenet (EE-TCP/IP stack).

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

```
void NetUseCommEenet();
```

### Description

This function enables the DME comm layer to support using eenet on the PS2.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetUseDme\(\)](#)

## NetUseCommInet

This function Declares that this application will use inet (IOP-TCP/IP stack).

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

```
void NetUseCommInet();
```

### Description

This function enables the DME comm layer to support using inet on the PS2.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetUseDme\(\)](#)

## NetUseCommSockets

This function declares that this application will use sockets on Windows or Unix.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

```
void NetUseCommSockets();
```

### Description

This function enables the DME comm layer to support using sockets on Windows or Unix.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetUseDme\(\)](#)



## NetUseCrypt

This function declares that this application will use security.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

```
void NetUseCrypt();
```

### Description

This function lets the library use Crypt (the SCE-RT security library). It calls the export function of crypt, which copies the interface functions to the interface function pointers, causing the linker to link in the interface.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetUseDme\(\)](#)

## NetUseDme

This function declares that this application will use SCE-RT libraries.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

```
void NetUseDme();
```

### Description

This function declares that this application will use SCE-RT libraries. Thus, every application must call this function.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetInitialize\(\)](#)

## NetUseObjects

This function declares that this application will use DME network objects.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

```
void NetUseObjects();
```

### Description

This function lets the library use DME network objects. It calls the export function of Objects, which copies the interface functions to the interface function pointers, causing the linker to link in the interface.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetUseDme\(\)](#)

## NetUsePeer2Peer

This function declares that the application will use peer-to-peer communication.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

```
void NetUsePeer2Peer();
```

### Description

This function lets the library use peer-to-peer communication. This function calls the export function of P2P (a DME layer), which copies the interface functions to the interface function pointers, causing the linker to link in the interface.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetUseDme\(\)](#)

## NetUseStreamMedia

This function declares that this application will use Stream Media.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

```
void NetUseStreamMedia();
```

### Description

This function lets the library use Stream Media. It calls the export function of Stream media, which copies the interface functions to the interface function pointers, causing the linker to link in the interface. Also, this function initializes data members of the interface structure.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetUseDme\(\)](#)

## NetUseVCodecJPEG

This function declares that this application will use a video codec and which one.

<i>Link to file</i>	<i>Include file</i>	<i>Introduced</i>	<i>Last modified</i>
librtbasePS2.a	dme.h	2.09	November 11, 2004

### Syntax

```
void NetUseVCodecJPEG();
```

### Description

This function lets the library specify which video codec to use. It calls the export function, which copies the interface functions to the interface function pointers, causing the linker to link in the interface. Also, this function initializes data members of the interface structure.

### Notes

N/A

### Return value

NetErrorNone: If successful.

### Example

N/A

### See also

[NetUseDme\(\)](#)

Index

This page intentionally left blank.

## Index

### A

aAccessKey ..... 4-14  
aAddressList ..... 4-4  
Address ..... 4-3, 4-39, 4-40  
AddressList ..... 4-14, 4-52  
AddressType ..... 4-3  
aErrors ..... 4-95  
ApplicationID ..... 4-28, 4-37  
ApplicationName ..... 4-28, 4-37  
aSessionKey ..... 4-14  
asziPAddresses ..... 4-87  
AudioDataCharacteristics ..... 4-67  
AudioType ..... 4-59, 4-60  
AuxUDPBindPort ..... 4-12  
AverageRate ..... 4-9

### B

base\_id ..... 4-7  
bCanRecord ..... 4-62  
bDataProcessed ..... 4-69  
bDataStored ..... 4-70  
bDefaultSet.. 4-12, 4-23, 4-26, 4-28, 4-31, 4-36, 4-53, 4-67  
bDefaultsSet ..... 4-66  
bEnabDisconnectFwd ..... 4-23  
bEnableLANBroadcastComms ..... 4-29  
bEndOfMessage ..... 4-94  
blgnore ..... 4-66

blsSet ..... 4-17, 4-25, 4-27, 4-30, 4-32, 4-52, 4-54  
bitmask ..... 4-7  
bJoinedToChannel ..... 4-63  
bRespondToSender ..... 4-35  
BufferComplete ..... 4-78  
BufferEnd ..... 4-78  
BufferStart ..... 4-78  
BufSize ..... 4-59, 4-60  
bUseStreamMedia ..... 4-67  
bUseTimeBase ..... 4-12, 4-23  
bUseToken ..... 4-73  
BytesRead ..... 4-59  
BytesStored ..... 4-60

### C

cbSize ..... 4-10, 4-55  
ChannelNum ..... 4-61, 4-63  
CharacterEncodingType ..... 4-45  
ChildFieldOffset ..... 4-93  
ChildFieldType ..... 4-93  
CircularBuffer ..... 4-78  
ClientCount ..... 4-61  
ClientIndex .. 4-51, 4-59, 4-63, 4-64, 4-66, 4-69, 4-76, 4-89,  
4-90  
ClientIpString ..... 4-76  
ClientList ..... 4-47  
ClientMask ..... 4-8  
ClientObjectIndex ..... 4-75  
ClientStatus ..... 4-76  
ClientStatusConnected ..... 2-7  
ClientStatusJoined ..... 2-7



ClientStatusJoinedSessionMaster .....	2-7
ClientStatusJoining .....	2-7
ClientStatusNone .....	2-7
ClientStatusNotConnected .....	2-7
ConnectFailureReasonAccessKey .....	2-8
ConnectFailureReasonAuth .....	2-8
ConnectFailureReasonAuxUDPFailure .....	2-8
ConnectFailureReasonClientVer .....	2-8
ConnectFailureReasonEncryption .....	2-8
ConnectFailureReasonError .....	2-8
ConnectFailureReasonFull .....	2-8
ConnectFailureReasonNone .....	2-8
ConnectFailureReasonServerVer .....	2-8
ConnectFailureReasonWorldID .....	2-8
ConnectionHandle 4-17, 4-19, 4-25, 4-31, 4-34, 4-42, 4-44, 4-48, 4-51, 4-56, 4-58, 4-61, 4-63, 4-66, 4-71, 4-72, 4- 76, 4-77, 4-89, 4-90, 4-94, 4-96 .....	
ConnectionInfo .....	4-12
ConnectivityType .....	4-28
ConnectStatus .....	4-77
ConnectStatusClosed .....	2-11
ConnectStatusDisconnected .....	2-11
ConnectStatusDisconnecting .....	2-11
ConnectStatusFailed .....	2-11
ConnectStatusNeedDisconnect .....	2-11
ConnectStatusOpen .....	2-11
ConnectStatusPending .....	2-11
ConnectTime .....	4-75
CreatorClientIndex .....	4-88
CurrentObjectData .....	4-88

## D

data .....	4-10, 4-55
DataRate .....	4-78
DataStreamCount .....	4-31
DestClient .....	4-56
Details .....	4-33, 4-34, 4-35, 4-36
DmeVersion .....	4-37

## E

eEncodingType .....	4-94
eLanguageType .....	4-94
ElementCount .....	4-81
ElementSize .....	4-81
EnumNetPlatformID .....	2-3
EnumNetPlatformID_PS2 .....	2-3
EnumNetPlatformID_PSP .....	2-3
EnumNetPlatformID_Unknown .....	2-3
ErrorCode... 4-17, 4-25, 4-27, 4-30, 4-32, 4-42, 4-50, 4-52, 4-54, 4-57, 4-71, 4-72, 4-77, 4-87 .....	
ErrorThresholdMagnitude .....	4-74
ErrorThresholdType .....	4-74
EventType .....	4-51
ExtraConnectFailureReason .....	2-8

ExtraConnectStaus .....	2-11
ExtraEnumNetPlatformID .....	2-3
ExtraNetAddressType .....	2-4
ExtraNetCharacterEncodingType .....	2-5
ExtraNetClientEventType .....	2-6
ExtraNetClientStatus .....	2-7
ExtraNetConnectionType .....	2-9
ExtraNetConnectivityType .....	2-10
ExtraNetDisconnectReason .....	2-12
ExtraNetErrorCode .....	2-20
ExtraNetFieldTypes .....	2-21
ExtraNetLanguageType .....	2-22
ExtraNetMessageClass .....	2-23
ExtraNetObjectLifespan .....	2-24
ExtraNetObjectOwnershipType .....	2-25
ExtraNetOwnershipStatus .....	2-26
ExtraNetSessionType .....	2-27
ExtraNetStreamMediaAudioType .....	2-28
ExtraNetStreamMediaGridType .....	2-29
ExtraNetSystemStatus .....	2-30
ExtraNetThresholdMethod .....	2-31
ExtraNetUpdateType .....	2-32

## F

FailureReason .....	4-77
FieldCount .....	4-22, 4-93
FieldIndex .....	4-22
FieldSize .....	4-22
FieldsUpdated .....	4-47
FieldTypeChildStructure .....	2-21
FieldTypeDouble .....	2-21
FieldTypeDoubleVector2 .....	2-21
FieldTypeDoubleVector3 .....	2-21
FieldTypeFloat .....	2-21
FieldTypeFloatVector2 .....	2-21
FieldTypeFloatVector3 .....	2-21
FieldTypeIntVector2 .....	2-21
FieldTypeIntVector3 .....	2-21
FieldTypeShortVector2 .....	2-21
FieldTypeShortVector3 .....	2-21
FieldTypeSignedChar .....	2-21
FieldTypeSignedInt .....	2-21
FieldTypeSignedShort .....	2-21
FieldTypeUnsignedChar .....	2-21
FieldTypeUnsignedInt .....	2-21
FieldTypeUnsignedShort .....	2-21
Filter .....	4-36
FilterType .....	4-88

## G

GridType .....	4-67
----------------	------

## H

HDME .....	3-3
------------	-----

HiFieldChangeSet .....	4-88
HostClientIndex .....	4-48

**I**

IncomingAddressList .....	4-26
---------------------------	------

**K**

key .....	4-98
KM_GetSoftwareID .....	6-3

**L**

LanguageType .....	4-45
LastGlobalObjectDataUpdate .....	4-88
LatencyAvg .....	4-43
LatencyCritical .....	4-88
LatencyMax .....	4-43
LatencyMetricsInfo .....	4-42
LatencyMin .....	4-43
LifespanType .....	4-88
lineSize .....	4-10, 4-55
Localization .....	4-28, 4-37
LocalUserData .....	4-88
LoFieldChangeSet .....	4-88

**M**

max_id .....	4-7
MaxAverageRate .....	4-9
MaxClients .....	4-12, 4-23
MaxDisconnectReason .....	2-12
MaxIncomingAudioStreams .....	4-67
MaxMessageClasses .....	2-23
MaxPacketSize .....	4-78
MaxUpdateInterval .....	4-88
MessageClassApplication .....	2-23
MessageClassDME .....	2-23
MessageClassLobby .....	2-23
MessageClassLobbyAuthentication .....	2-23
MessageClassLobbyExt .....	2-23
MessageClassLobbyReport .....	2-23
MessageData .....	4-56
MessageLength .....	4-56
MessageType .....	4-56
MinPacketSize .....	4-78
MinUpdateInterval .....	4-74
myConnectStatus .....	4-16

**N**

Name .....	4-75, 4-88, 4-93
NatServiceAddress .....	4-53
nBitsPerSampleIn .....	4-5
nBitsPerSampleOut .....	4-5
nBufferSize .....	4-65
nBytesAvailable .....	4-64

nBytesProcessed .....	4-64
nBytesStored .....	4-65
nChannelsIn .....	4-5
nChannelsOut .....	4-5
nConnectedClientCount .....	4-16
NET_ADDRESS_LIST_COUNT .....	1-3
NET_ALL_CLIENTS .....	1-4
NET_CLIENT_LIMIT .....	1-5
NET_CLIENT_MASK .....	1-6
NET_CONNECTION_INVALID .....	1-7
NET_CONNECTION_UDP .....	1-8
NET_DEFAULT_RECEIVE_BUFFER_SIZE .....	1-9
NET_DEFAULT_SEND_BUFFER_SIZE .....	1-10
NET_DELIVERY_CRITICAL .....	1-11
NET_FULL_OBJECT_UPDATE .....	1-12
NET_INVALID_CLIENT_INDEX .....	1-13
NET_LAN_RAW_MESSAGE_CALLBACK .....	5-3
NET_LAN_TEXT_MESSAGE_CALLBACK .....	5-4
NET_LANFIND_FILTER_APP .....	1-14
NET_LANFIND_FILTER_PLATFORM .....	1-15
NET_LATENCY_CRITICAL .....	1-16
NET_MAX_ADDRESS_STR_LENGTH .....	1-17
NET_MAX_APPLICATION_CHAR_LEN .....	1-18
NET_MAX_APPLICATION_NAME_LEN .....	1-19
NET_MAX_APPLICATION_NAME_SIZE .....	1-20
NET_MAX_BITMASK_ARRAY .....	1-21
NET_MAX_CLIENT_NAME_LENGTH .....	1-22
NET_MAX_CONNECTIONS .....	1-23
NET_MAX_HOSTNAME_LENGTH .....	1-24
NET_MAX_IP_LENGTH .....	1-25
NET_MAX_LANFIND_DETAILS_SIZE .....	1-26
NET_MAX_MEDIA_CHANNELS .....	1-27
NET_MAX_NETADDRESS_LENGTH .....	1-28
NET_MAX_OBJECT_NAME_LENGTH .....	1-29
NET_MAX_RESPONSES .....	1-30
NET_MAX_STRUCT_NAME_LENGTH .....	1-31
NET_NO_CONNECTION .....	1-32
NET_OBJECT_NOT_FILTERED .....	1-33
NET_OBJECT_OWNERSHIP_SHARED .....	1-34
NET_OBJECT_OWNERSHIP_UPDATE .....	1-35
NET_ORDER_CRITICAL .....	1-36
NET_SEND_TO_ALL_CLIENTS .....	1-37
NET_SEND_TO_CLIENT_MASK .....	1-38
NET_SEND_TO_SERVER .....	1-39
NET_SERVER_QOS_CRITICAL .....	1-40
NET_SESSION_KEY_LEN .....	1-41
NET_TIMESTAMP_STRING_LENGTH .....	1-42
NET_TOKEN_FREE_OWNER .....	1-43
NET_VERSION_STRING_LENGTH .....	1-44
NetAddress .....	4-3
NetAddressList .....	4-4
NetAddressNone .....	2-4
NetAddressToStringAddress .....	6-4
NetAddressType .....	2-4
NetAddressTypeBinaryExternal .....	2-4

NetAddressTypeBinaryExternalVport .....	2-4	NetDisconnectMessageLengthMismatch .....	2-12
NetAddressTypeBinaryInternal .....	2-4	NetDisconnectNone .....	2-12
NetAddressTypeBinaryInternalVport .....	2-4	NetDisconnectNormal .....	2-12
NetAddressTypeBinaryNATServices .....	2-4	NetDisconnectParams .....	4-19
NetAddressTypeExternal .....	2-4	NetDisconnectReason .....	2-12
NetAddressTypeInternal .....	2-4	NetDisconnectShutdown .....	2-12
NetAddressTypeNATService .....	2-4	NetDisconnectStreamMediaFail .....	2-12
NetAddToDataStream .....	6-5	NetDisconnectUpdateFail .....	2-12
NetAssignLocalServer .....	6-6	NetDmeVersion .....	4-20
NetAudioDataCharacteristics .....	4-5	NetEnableLANMessaging .....	6-16
NetBandwidthInfo .....	4-6	NetEnableLanMessagingInParams .....	4-21
NetBitMask .....	4-7	NetEndDataStream .....	6-17
NetBitMaskIsSet .....	6-7	NetError .....	4-96
NetBitMaskSet .....	6-8	NetErrorBadConnectionIndex .....	2-15
NetBitMaskUnSet .....	6-9	NetErrorBadConnectivityType .....	2-20
NetCharacterEncodingISO8859_1 .....	2-5	NetErrorBadDataStreamChannel .....	2-14
NetCharacterEncodingNone .....	2-5	NetErrorBadIndex .....	2-14
NetCharacterEncodingType .....	2-5	NetErrorBadMode .....	2-15
NetCharacterEncodingUTF_8 .....	2-5	NetErrorBadPacketReceived .....	2-14
NetClientEventJoin .....	2-6	NetErrorBadPointer .....	2-14
NetClientEventLeave .....	2-6	NetErrorBadPort .....	2-20
NetClientEventType .....	2-6	NetErrorBadServerVersion .....	2-18
NetClientList .....	4-8	NetErrorBadSessionMaster .....	2-13
NetClientMetric .....	4-9	NetErrorBufferError .....	2-18
NetClientStatus .....	2-7	NetErrorClientNotValid .....	2-16
NetClose .....	6-10	NetErrorClientRejected .....	2-13
NetColorArray .....	4-10	NetErrorCode .....	2-13
NetCompletionData .....	4-11	NetErrorCommError .....	2-18
NetConnect .....	6-11	NetErrorConnectionFailed .....	2-13
NetConnectFailureReason .....	2-8	NetErrorConnectionLost .....	2-13
NetConnectInParams .....	4-12	NetErrorDeprecated .....	2-18
NetConnectionInfo .....	4-14	NetErrorDisconnectFailed .....	2-13
NetConnectionNone .....	2-9	NetErrorDmeNotInitialized .....	2-15
NetConnectionStatus .....	4-16	NetErrorGamelsFull .....	2-17
NetConnectionType .....	2-9	NetErrorHostGameFailed .....	2-17
NetConnectionTypeClientListenerTCP .....	2-9	NetErrorHostnameError .....	2-19
NetConnectionTypeClientServerTCP .....	2-9	NetErrorInitFailed .....	2-16
NetConnectionTypeClientServerTCPAuxUDP .....	2-9	NetErrorInvalidArg .....	2-16
NetConnectionTypePeerToPeerUDP .....	2-9	NetErrorLookupError .....	2-19
NetConnectivityInternet .....	2-10	NetErrorMemory .....	2-16
NetConnectivityLAN .....	2-10	NetErrorMsgError .....	2-16
NetConnectivityNone .....	2-10	NetErrorMsgTooLarge .....	2-17
NetConnectivityType .....	2-10	NetErrorMutex .....	2-19
NetConnectOutParams .....	4-17	NetErrorNewClient .....	2-13
NetConnectStatus .....	2-11	NetErrorNoFreeObject .....	2-16
NetCreateObject .....	6-12	NetErrorNone .....	2-13
NetData .....	4-18	NetErrorNotConnected .....	2-13
NetDataStreamBytesFree .....	6-13	NetErrorNotImplemented .....	2-17
NetDataStreamCount .....	4-75	NetErrorObjectNotShared .....	2-14
NetDataStreamStart .....	4-75	NetErrorOpenDataStreamFailed .....	2-16
NetDisableLanMessaging .....	6-14	NetErrorSecurity .....	2-17
NetDisconnect .....	6-15	NetErrorSendFailed .....	2-15
NetDisconnectAppDefinedStart .....	2-12	NetErrorServerError .....	2-19
NetDisconnectConnectFail .....	2-12	NetErrorSetDefaultsNotCalled .....	2-18
NetDisconnectInactivity .....	2-12	NetErrorSizeofParamMismatch .....	2-18

NetErrorStreamMedia .....	2-18	NetJoinOutParams .....	4-32
NetErrorStreamMediaComm.....	2-18	NetLANFind.....	6-50
NetErrorThresholdCallbackData .....	4-22	NetLANFindCallback .....	5-6
NetErrorTimebaseError .....	2-19	NetLANFindCallbackDataArgs .....	4-33
NetErrorTimedOut.....	2-15	NetLANFindCancel .....	6-51
NetErrorTokenError.....	2-19	NetLANFindEnableExchange.....	6-52
NetErrorTokenNotEnabled .....	2-19	NetLANFindExchangeCallback .....	5-7
NetErrorTooManyPendingEvents .....	2-17	NetLANFindExchangeCallbackInArgs .....	4-34
NetErrorUDPError .....	2-18	NetLANFindExchangeCallbackOutArgs .....	4-35
NetErrorUDPNotEnabled .....	2-17	NetLANFindInParams .....	4-36
NetErrorUnknown .....	2-17	NetLanguageChinese .....	2-22
NetErrorUpdate .....	2-17	NetLanguageDutch .....	2-22
NetFieldTypes.....	2-21	NetLanguageFinnish.....	2-22
NetFieldUpdate.....	2-32	NetLanguageFrench.....	2-22
NetFreeAllObjects .....	6-18	NetLanguageGerman .....	2-22
NetFreeCallback .....	5-5	NetLanguageItalian.....	2-22
NetFreeObject .....	6-19	NetLanguageJapanese.....	2-22
NetFullObjectUpdate.....	2-32	NetLanguageKorean .....	2-22
NetGenerateClientList .....	6-20	NetLanguageNone .....	2-22
NetGenerateJoinedClientList.....	6-21	NetLanguageNorwegian .....	2-22
NetGetAverageDelayToClient.....	6-22	NetLanguagePortuguese.....	2-22
NetGetBandwidthInfo .....	6-23	NetLanguageSpanish .....	2-22
NetGetBufferStatus.....	6-24	NetLanguageTaiwanese .....	2-22
NetGetBuildTimeStamp .....	6-25	NetLanguageType .....	2-22
NetGetClient .....	6-26	NetLanguageUKEnglish.....	2-22
NetGetClientIpAddress .....	6-27	NetLanguageUSEngish.....	2-22
NetGetClientStatus .....	6-28	NetLANPeerDesc .....	4-37
NetGetClientVersion.....	6-29	NetLanRawMessageCallbackParams .....	4-38
NetGetConnectionStatus .....	6-30	NetLANSendRawMessage .....	6-53
NetGetConnectStatus.....	6-31	NetLANSendRawMessageInParams.....	4-39
NetGetDataStream .....	6-32	NetLANSendTextMessage .....	6-54
NetGetHostByName .....	6-33	NetLANSendTextMessageInParams.....	4-40
NetGetLatencyMetrics .....	6-34	NetLANSetDefaultEnableMessagingInParams .....	6-55
NetGetLocalTime .....	6-35	NetLANSetDefaultSendRawMessageInParams.....	6-56
NetGetMyClientIndex .....	6-36	NetLANSetDefaultSendTextMessageInParams.....	6-57
NetGetMyIpAddress .....	6-37	NetLANSetUserName.....	6-58
NetGetMyNetAddress.....	6-38	NetLanTextMessageCallbackParams .....	4-41
NetGetNetUpdateErrors.....	6-39	NetLatencyMetricsDataArgs .....	4-42
NetGetObject.....	6-40	NetLatencyMetricsInfo .....	4-43
NetGetPeerToPeerHostClientIndex .....	6-41	NetLatencyMetricsParams.....	4-44
NetGetServerVersion .....	6-42	NetLeave.....	6-59
NetGetSessionMasterClientIndex.....	6-43	NetLocalizationParams .....	4-45
NetGetTime .....	6-44	NetMallocCallback.....	5-8
NetGetValidClientCount .....	6-45	NetMemoryCallbackParams .....	4-46
NetHostPeerToPeer.....	6-46	NetMessageClass .....	2-23
NetHostPeerToPeerInParams .....	4-23	NetObjectBufferCount .....	4-75
NetHostPeerToPeerOutParams .....	4-25	NetObjectBufferStart .....	4-75
NetIncomingClient .....	6-47	NetObjectCount .....	4-31
NetIncomingClientInParams.....	4-26	NetObjectField.....	6-60
NetIncomingClientOutParams.....	4-27	NetObjectFilterData .....	4-47
NetInitialize .....	6-48	NetObjectLifespan .....	2-24
NetInitializeInParams .....	4-28	NetObjectOwnershipDenied .....	2-25
NetInitializeOutParams.....	4-30	NetObjectOwnershipGranted.....	2-25
NetJoin .....	6-49	NetObjectOwnershipNone.....	2-25
NetJoinInParams .....	4-31	NetObjectOwnershipNotShared .....	2-25

NetObjectOwnershipShared .....	2-25	NetSetDefaultResolveAddrParams .....	6-96
NetObjectOwnershipType .....	2-25	NetSetDefaultStreamMediaParams .....	6-97
NetOpenDataStream .....	6-61	NetSetFieldChanged .....	6-98
NetOwnershipStatus .....	2-26	NetSetMyClientObject .....	6-99
NetPeerToPeerHostChangeData .....	4-48	NetSetMyClientReceiveBroadcast .....	6-100
NetPing .....	6-62	NetSetNATServiceAddr .....	6-101
NetPingIP .....	6-63	NetSetSendAggregationInterval .....	6-102
NetPingNetAddress .....	6-64	NetSMChangeData .....	4-58
NetPlatformID .....	4-37	NetStreamMediaAudioPlayData .....	4-59
NetReallocCallback .....	5-9	NetStreamMediaAudioRecordData .....	4-60
NetRegisterApplicationMessage .....	6-65	NetStreamMediaAudioType .....	2-28
NetRegisterDataStream .....	6-66	NetStreamMediaAudioTypeCUSTOM .....	2-28
NetRegisterMemoryCallbacks .....	6-67	NetStreamMediaAudioTypeGSM .....	2-28
NetRegisterMessage .....	6-68	NetStreamMediaAudioTypeLPC .....	2-28
NetRegisterObjectField .....	1-45	NetStreamMediaAudioTypeLPC10 .....	2-28
NetRegisterObjectFilter .....	6-69	NetStreamMediaAudioTypeRAW .....	2-28
NetRegisterObjectFilterInParams .....	4-49	NetStreamMediaChannelInfo .....	4-61
NetRegisterObjectFilterOutParams .....	4-50	NetStreamMediaChannelStateData .....	4-62
NetRegisterObjectStart .....	6-70	NetStreamMediaClientInfo .....	4-63
NetRegisterRemoteObjectCallback .....	6-71	NetStreamMediaCustomVideoPlayData .....	4-64
NetRegisterStructure .....	6-72	NetStreamMediaCustomVideoRecordData .....	4-65
NetReleaseObjectPrivateOwnership .....	6-73	NetStreamMediaEndRecording .....	6-103
NetRemoteClientEventData .....	4-51	NetStreamMediaGetChannelInfo .....	6-104
NetRequestCreateRemoteNamedObject .....	6-74	NetStreamMediaGetClientInfo .....	6-105
NetRequestObjectPrivateOwnership .....	6-75	NetStreamMediaGetCurrentChannelState .....	6-106
NetResolveAddr .....	6-76	NetStreamMediaGridType .....	2-29
NetResolveAddrData .....	4-52	NetStreamMediaGridTypeDirect .....	2-29
NetResolveAddrInParams .....	4-53	NetStreamMediaGridTypeRelay .....	2-29
NetResolveAddrOutParams .....	4-54	NetStreamMediaIgnoreData .....	4-66
NetRGBArray .....	4-55	NetStreamMediaJoinChannel .....	6-107
NetSendApplicationMessage .....	6-77	NetStreamMediaParams .....	4-67
NetSendAppMessage .....	6-78	NetStreamMediaQuitChannel .....	6-108
NetSendFieldUpdates .....	6-79	NetStreamMediaSetDefaultIgnoreParams .....	6-109
NetSendMessage .....	6-80	NetStreamMediaSetIgnoreState .....	6-110
NetSendMessageInParams .....	4-56	NetStreamMediaVideoPlayData .....	4-69
NetSendMessageOutParams .....	4-57	NetStreamMediaVideoRecordData .....	4-70
NetSendMyClientUpdate .....	6-81	NetSystemStatus .....	2-30
NetSendObjectFullUpdate .....	6-82	NetSystemStatusData .....	4-71
NetSessionType .....	2-27	NetSystemTokenReady .....	2-30
NetSessionTypeGame .....	2-27	NetThresholdMethod .....	2-31
NetSessionTypeIntegratedServer .....	2-27	NetTick .....	6-111
NetSessionTypePeer .....	2-27	NetTokenListRelease .....	6-112
NetSetDefaultAppMessageParams .....	6-83	NetTokenListRequest .....	6-113
NetSetDefaultBitMask .....	6-84	NetTokenOwnershipNotifyData .....	4-72
NetSetDefaultConnectParams .....	6-85	NetTokenParams .....	4-73
NetSetDefaultDisconnectParams .....	6-86	NetTokenQuery .....	6-114
NetSetDefaultHostPeerToPeerParams .....	6-87	NetTokenRelease .....	6-115
NetSetDefaultIncomingClientParams .....	6-88	NetTokenRequest .....	6-116
NetSetDefaultInitializeParams .....	6-89	NetTokenSystemQuery .....	6-117
NetSetDefaultJoinParams .....	6-90	NetTypeBroadcastSchedule .....	4-74
NetSetDefaultLANFindParams .....	6-91	NetTypeClient .....	4-75
NetSetDefaultLatencyMetricsParams .....	6-92	NetTypeClientConnectCallback .....	5-10
NetSetDefaultLookupParams .....	6-93	NetTypeClientConnectCallbackData .....	4-76
NetSetDefaultMemoryCallbackParams .....	6-94	NetTypeCompletionCallback .....	5-11
NetSetDefaultRegisterObjectFilterParams .....	6-95	NetTypeConnectCallback .....	5-12

NetTypeConnectCallbackData .....	4-77
NetTypeDataStream .....	4-78
NetTypeDataStreamEndCallback .....	5-13
NetTypeDataStreamFilterCallBack .....	5-14
NetTypeDataStreamUpdateCallback .....	5-15
NetTypeDoubleVector2 .....	4-79
NetTypeDoubleVector3 .....	4-80
NetTypeErrorThresholdCallback .....	5-16
NetTypeField .....	4-81
NetTypeFloatVector2 .....	4-82
NetTypeFloatVector3 .....	4-83
NetTypeIntVector2 .....	4-84
NetTypeIntVector3 .....	4-85
NetTypeLatencyMetricsCallback .....	5-17
NetTypeLookupCallback .....	5-18
NetTypeLookupParams .....	4-86
NetTypeLookupResponse .....	4-87
NetTypeMessageParser .....	5-19
NetTypeObject .....	4-88
NetTypeObjectCallback .....	5-20
NetTypeObjectFilterCallback .....	5-21
NetTypeObjectUpdateCallback .....	5-22
NetTypeOwnershipRequestCallback .....	5-23
NetTypeOwnershipRequestData .....	4-89
NetTypeOwnershipUpdateCallback .....	5-24
NetTypeOwnershipUpdateData .....	4-90
NetTypePeerToPeerHostChangeCallback .....	5-25
NetTypePingCallback .....	5-26
NetTypeRemoteClientEventCallback .....	5-27
NetTypeResolveAddrCallback .....	5-28
NetTypeShortVector2 .....	4-91
NetTypeShortVector3 .....	4-92
NetTypeSMChangeCallback .....	5-29
NetTypeStreamMediaAudioPlayCallback .....	5-30
NetTypeStreamMediaAudioRecordCallback .....	5-31
NetTypeStreamMediaCustomVideoPlayCallback .....	5-32
NetTypeStreamMediaCustomVideoRecordCallback .....	5-33
NetTypeStreamMediaVideoPlayCallback .....	5-34
NetTypeStreamMediaVideoRecordCallback .....	5-35
NetTypeStructure .....	4-93
NetTypeSystemMessageCallback .....	5-36
NetTypeSystemMessageData .....	4-94
NetTypeSystemStatusCallback .....	5-37
NetTypeTokenOwnershipNotifyCallback .....	5-38
NetUpdate .....	6-118
NetUpdateConnErrors .....	4-95
NetUpdateError .....	4-96
NetUpdateType .....	2-32
NetUseACodecGSM .....	6-119
NetUseACodecLPC .....	6-120
NetUseACodecLPC10 .....	6-121
NetUseClientServer .....	6-122
NetUseCommAdhoc .....	6-123
NetUseCommEEnet .....	6-124
NetUseCommInet .....	6-125

NetUseCommSockets .....	6-126
NetUseCrypt .....	6-127
NetUseDme .....	6-128
NetUseObjects .....	6-129
NetUsePeer2Peer .....	6-130
NetUseStreamMedia .....	6-131
NetUseVCodecJPEG .....	6-132
NetVideoDataCharacteristics .....	4-97
nIPAddresses .....	4-87
nMaxNumClients .....	4-33
nMessageLength .....	4-94
nMsgSize .....	4-38, 4-39, 4-41
nNumClients .....	4-33
NoThreshold .....	2-31
nSampleRateIn .....	4-5
nSampleRateOut .....	4-5
nSize .....	4-18
nTextMsgSize .....	4-40
NumChannels .....	4-67
nValidClientCount .....	4-16
nVersion .....	4-20

## O

ObjectFilterCallBack .....	4-49
ObjectFilterType .....	4-50
ObjectIndex .....	4-22, 4-47, 4-89, 4-90
ObjectLifespanClient .....	2-24
ObjectLifespanOneUpdate .....	2-24
ObjectLifespanPermanent .....	2-24
ObjectLifespanSession .....	2-24
ObjectLifespanTimeout .....	2-24
Offset .....	4-81
OwnerClientIndex .....	4-72, 4-78, 4-88
OwnershipNone .....	2-26
OwnershipNotAvailable .....	2-26
OwnershipPrivate .....	2-26
OwnershipShared .....	2-26

## P

pApplicationKeyPair .....	4-28
pAudioPlayCallbackData .....	4-67
pAudioRecordCallbackData .....	4-67
pBuffer .....	4-59, 4-60, 4-64, 4-65
pCb .....	4-70
pCr .....	4-70
pCurrentData .....	4-22
pCustomVideoPlayCallbackData .....	4-68
pCustomVideoRecordCallbackData .....	4-67
pData .....	4-18, 4-38
PeerAddress .....	4-37
PeerDesc .....	4-33, 4-34
PeerDescription .....	4-38, 4-41
pfAudioPlayCallback .....	4-67
pfAudioRecordCallback .....	4-67

pfCustomVideoPlayCallback ..... 4-67  
 pfCustomVideoRecordCallback ..... 4-67  
 pfFreeCallback ..... 4-46  
 pfLatencyMetricsCallback ..... 4-44  
 pfLocalConnectCallback ..... 4-12, 4-23  
 pfLocalDisconnectCallback ..... 4-12, 4-19, 4-23  
 pfLocalJoinCallback ..... 4-31  
 pfLookupResponse ..... 4-86  
 pfMallocCallback ..... 4-46  
 pfNLANFindCallback ..... 4-36  
 pfOwnershipRequestCallback ..... 4-31  
 pfOwnershipUpdateCallback ..... 4-31  
 pfPeerToPeerHostChangeCallback ..... 4-12  
 pfReallocCallback ..... 4-46  
 pfRemoteClientConnectCallback ..... 4-12, 4-23  
 pfRemoteClientDisconnectCallback ..... 4-12, 4-23  
 pfRemoteClientEventCallback ..... 4-31  
 pfResolveAddrCallback ..... 4-53  
 pfSMChangeCallback ..... 4-31  
 pfSystemMessageCallback ..... 4-29  
 pfSystemStatusCallback ..... 4-12, 4-23  
 pfThresholdCallback ..... 4-74  
 pfTokenOwnershipNotifyCallback ..... 4-73  
 pfVideoPlayCallback ..... 4-67  
 pfVideoRecordCallback ..... 4-67  
 pHostChangeCallbackData ..... 4-12  
 pJoinCallbackData ..... 4-31  
 pLanRawMessageCallback ..... 4-21  
 pLanTextMessageCallback ..... 4-21  
 pLastUpdateData ..... 4-22  
 pLocalConnectCallbackData ..... 4-12, 4-23  
 pLocalDisconnectCallbackData ..... 4-12, 4-23  
 pLocalKeyPair ..... 4-28  
 pMessage ..... 4-94  
 pMsg ..... 4-39  
 Port ..... 4-3  
 pOwnershipRequestCallbackData ..... 4-31  
 pOwnershipUpdateCallbackData ..... 4-31  
 pRawMessageCallbackUserData ..... 4-21  
 pRemoteClientConnectCallbackData ..... 4-12, 4-23  
 pRemoteClientDisconnectCallbackData ..... 4-12, 4-23  
 pRemoteClientEventCallbackData ..... 4-31  
 pRGBArray ..... 4-69  
 privateKey ..... 4-99  
 pSMChangeCallbackData ..... 4-31  
 pSystemMessageCallbackData ..... 4-29  
 pSystemStatusCallbackData ..... 4-12, 4-23  
 pTextMessageCallbackUserData ..... 4-21  
 pTextMsg ..... 4-40, 4-41  
 pTokenOwnershipNotifyCallbackData ..... 4-73  
 pubKey ..... 4-26  
 publicKey ..... 4-99  
 pUserData .. 4-11, 4-19, 4-33, 4-34, 4-36, 4-38, 4-41, 4-42,  
 4-44, 4-47, 4-48, 4-49, 4-51, 4-58, 4-59, 4-60, 4-64, 4-  
 65, 4-69, 4-70, 4-71, 4-72, 4-76, 4-77, 4-89, 4-90, 4-94

pVideoPlayCallbackData ..... 4-67  
 pVideoRecordCallbackData ..... 4-67  
 pY ..... 4-70

## Q

QueuedClient ..... 4-57

## R

ReadPtr ..... 4-78  
 Reason ..... 4-19  
 RecordNoDataTimeout ..... 4-67  
 RecvBufferSize ..... 4-13, 4-23, 4-67  
 RecvBytes ..... 4-6  
 Recvs ..... 4-6  
 RemoteBuffer ..... 4-78  
 Result ..... 4-11  
 RSA\_KEY ..... 4-98  
 RSA\_KEYPAIR ..... 4-99

## S

SendBufferSize ..... 4-12, 4-23, 4-67  
 SendBytes ..... 4-6  
 Sends ..... 4-6  
 SendToAll ..... 4-78  
 ServerKey ..... 4-14  
 SessionMasterStatus ..... 4-31  
 SessionType ..... 4-33, 4-34, 4-36  
 Severity ..... 4-94  
 SizeofNetUpdateErrors ..... 4-95  
 SMClientIndex ..... 4-58  
 state ..... 4-90  
 Status ..... 4-71, 4-78  
 StreamMediaParams ..... 4-12, 4-23  
 StreamType ..... 4-78  
 StructureIndex ..... 4-88  
 szClientName ..... 4-31  
 szHostName ..... 4-86  
 szServerIP ..... 4-86  
 szVersion ..... 4-20, 4-30

## T

TargetClient ..... 4-8  
 TargetClientIndex ..... 4-42, 4-78  
 TargetClientList ..... 4-44  
 ThresholdAbsoluteMagnitude ..... 2-31  
 ThresholdAnchorDelta ..... 2-31  
 ThresholdCallback ..... 2-31  
 ThresholdEquality ..... 2-31  
 ThresholdRatioMagnitude ..... 2-31  
 TimeOfExpiration ..... 4-88  
 TimeOfLastClientFieldUpdate ..... 4-88  
 TimeOfLastClientUpdate ..... 4-88  
 TimeOfLastGlobalUpdate ..... 4-88  
 TimeOfLastUpdate ..... 4-78

TokenID .....	4-72
TokenParams .....	4-12, 4-23
TotalSize.....	4-93
TransportFlags.....	4-56, 4-60, 4-65, 4-74
Type .....	4-14, 4-81

## U

UdpBindPort.....	4-28
UDPErrors .....	4-95
UDPPort .....	4-36
UpdateSchedule .....	4-81
UpdateType.....	4-47
UPnPMemoryCeiling .....	4-29
UserName .....	4-37
UserSpecified .....	4-12, 4-23, 4-76

## V

VideoDataCharacteristics.....	4-67
-------------------------------	------

## W

WorldID .....	4-14
WritePtr .....	4-78

## X

x 4-79, 4-80, 4-82, 4-83, 4-84, 4-85, 4-91, 4-92	
xsize.....	4-10, 4-55
XSize .....	4-97

## Y

y 4-79, 4-80, 4-82, 4-83, 4-84, 4-85, 4-91, 4-92	
ysize.....	4-10, 4-55
YSize.....	4-97

## Z

z 4-80, 4-83, 4-85, 4-92	
--------------------------	--