Appendix 9A
NDS APIs

Contents

Authentication APIs

NDSAuthenticateWithHandle 4
NDSCloseAuthenticationHandle 5
NDSCreateAuthenticationHandle 6
NDSGetAuthenticationInfo 8
NDSUnauthenticate 9

Name Service APIs

NDSGetPreferredTree 10
NDSSetPreferredTree 11
NDSResolveObjectToId 12
NDSResolveNameToAddress 15
NDSResolveIdToObject 17

NDS-Specific APIs

NDSGetDefaultNameContext 18
NDSSetDefaultNameContext 20
NDSRequestReply 21

VLM-Compatibility APIs

NDSVLMSetUserObjectId 23
NDSVLMGetSetMonitored 24
NDSVLMReadTDS 25
NDSVLMWriteTDS 26

Unicode APIs

TBD

NDSAuthenticateWithHandle

Description

Authenticates the connection with the specified authentication information generated with NDSCreateAuthenticationHandle.

Syntax

```
UINT32
NDSAuthenticateWithHandle(
    UINT32          authHandle,
    CONN_HANDLE connHandle)
```

Input

authHandle Handle to authentication information to authenticate with.

connHandle Connection to authenticate.

Output

None.

Return values

SUCCESS_CODE
INVALID_CONNECTION
INVALID_PARAMETER
INVALID_AUTHEN_HANDLE
AUTHENTICATION_FAILED
OUT_OF_CLIENT_MEMORY
DS network errors

See also

NDSUnauthenticate

NDSCloseAuthenticationHandle

Description Logs the user from the tree. Both the user and tree name are obtained from the authentication information.

Syntax

```
UINT32
NDSCloseAuthenticationHandle(
    UINT32    authHandle)
```

Input *authHandle* Handle of authentication information to be released.

Output None.

Return values

SUCCESS_CODE
INVALID_AUTHEN_HANDLE
DS network errors

See also NDSCreateAuthenticationHandle

NDSCreateAuthenticationHandle

Description

Creates the authentication information for the user in the specified tree.

Syntax

```
UINT32
NDSCreateAuthenticationHandle(
    UINT32          processGroupId,
    UINT32          processId,
    SPECT_DATA      *userName,
    SPECT_DATA      *password,
    SPECT_DATA      *treeName,
    VOID            *pAuthSpecInfo,
    UINT32          **pauthHandle)
```

Input

processGroupId Process group ID.

processId Process ID.

userName Pointer to distinguished name of user for which to create authentication information. This string may be ASCII or Unicode, and may be typed or typeless. A maximum of 256 characters is allowed.

password Pointer to password for user (max 128 bytes).

treeName Pointer to the name of the tree to log the user into.

pAuthSpecInfo Authentication-specific information.

Output

pauthHandle Handle to authentication information.

See also

SUCCESS_CODE
INVALID_PARAMETER
OUT_OF_CLIENT_MEMORY
DS network errors

Remarks

This call logs the user into the tree and generates the authentication materials needed to authenticate to connections within the tree. If authentication information already exists for the user in the specified tree, the existing handle will be returned along with its associated error code. If we are already logged into this tree as another user, an error will be returned.

See also

NDSAuthenticateWithHandle
NDSCloseAuthenticationHandle

NDSGetAuthenticationInfo

Description

Returns the distinguished user name and tree name associated with the specified authentication handle.

Syntax

```
UINT32
NDSGetAuthenticationInfo(
    UINT32          authHandle,
    SPECT_DATA      *userName,
    SPECT_DATA      *treeName,
    VOID            *pAuthSpecInfo)
```

Input

authHandle	Handle to authentication information to retrieve information from.

Output

userName	The complete Unicode distinguished name of the user associated with the authentication information. This string will contain a maximum of 512 characters.
treeName	The tree name associated with authentication information.
pAuthSpecInfo	Authentication-specific information. Ignored.

Return values

SUCCESS_CODE
INVALID_PARAMETER
INVALID_AUTHEN_HANDLE
MORE_DATA_ERROR
DS network errors

NDSUnauthenticate

Description Unauthenticates the specified connection.

Syntax

```
UINT32
NDSUnauthenticate(
    UINT32          authHandle,
    CONN_HANDLE connHandle)
```

Input

authHandle Handle to authentication information.

connHandle Connection to unauthenticate.

Output None.

Return values

SUCCESS_CODE
INVALID_CONNECTION

Remarks The connection will be authenticated as *public*.

See also NDSAuthenticateWithHandle

NDSGetPreferredTree

Description

Returns the current preferred tree name for the process group and process ID. The NET.CFG-configured preferred tree name will be returned if the preferred tree name has not been set for this process group and process ID.

Syntax

```
UINT32
NDSGetPreferredTree(
    UINT32          processGroupId,
    UINT32          processId,
    SPECT_DATA      *treeName)
```

Input

processGroupId Process group ID.

processId Process ID.

treeName *Length* field in the SPECT_DATA *treeName* structure contains the length the buffer to receive the preferred tree. NOTE: Buffer must be at least 32 bytes.

Output

treeName *Name* field in the SPECT_DATA *treeName* structure contains preferred server name. *Length* field contains the length of *name*.

Return values

SUCCESS_CODE
MORE_DATA_ERROR

Remarks

If MORE_DATA_ERROR is returned, as much of the tree name as possible was copied into the buffer, but there was more to copy.

See also

NDSSetPreferredTree

NDSSetPreferredTree

Description

Sets the current preferred tree name for the process group and process IDs.

Syntax

```
UINT32
NDSSetPreferredTree(
    UINT32          processGroupId,
    UINT32          processId,
    SPECT_DATA      *treeName)
```

Input

processGroupId Process group ID.

processId Process ID.

treeName Pointer to structure containing preferred tree name. The *Length* field in structure *treeName* contains the length of the preferred tree name begin set. NOTE: To reset preferred tree, set *length* field in *treeName* structure to zero.

Output

None.

Return values

SUCCESS_CODE
INVALID_PARAMETER
OUT_OF_CLIENT_MEMORY

See also

NDSGetPreferredTree

NDSResolveObjectToId

Description

Resolves the supplied object name to an object ID and transport address.

Syntax

```
UINT32
NDSResolveObjectToId(
    CONN_HANDLE         reqConnId,
    SPECT_DATA          *reqObject,
    SPECT_DATA          *reqObjectType,
    UINT32              reqTranType,
    NDS_RESOLVE_INFO    *reqNSSpec,
    UINT32              *repObjectId,
    UINT32              *repSessionServType,
    TRAN_ADDR_TYPE      *repTranBuf,
    UINT32              *repTranAddrCount)
```

Input

reqConnId — Connection handle where name might be resolved. This value may be NULL.

reqObject — Pointer to object name to be resolved. This string may by ASCII of Unicode and may be typed of typeless. A maximum of 256 characters is allowed.

reqObjectType — Pointer to the object type (CLASS) the *unserName* belongs to. NOTE: See list of object types in *Chapter 9B: NDS Structures and Definitions.*

reqTranType — Specifies the preferred or required transport type to be used as follows:

TRAN_TYPE_IPX
TRAN_TYPE_TCP
TRAN_TYPE_WILD

reqNSSpec Pointer to NDS_RESOLVE_INFO below:

```
typedef struct _NDS_RESOLVE_INFO{
   UINT32    tag;
   UINT32    flags;
   UINT32    reqFlags;
   UINT32    reqScope;
   UINT32    repResolveType;
   UINT32    repFlags;
   UINT32    resolvedOffset;
   UINT32    derefNameLength;
   UNICODE  *derefName;
} NDS_RESOLVE_INFO;
```

If the *tag* is specified and is equal to Client32_NAME_SVC_V1_00, then the *reqFlags* and *reqScope* passed in will be used in the resolve name request. Otherwise, the request flags will be set to DS_WRITABLE | DS_DEREFERENCE_ALIASES and the *reqScope* will be zero.

repTranAddrCount

Pointer to value containing the maximum number of TRAN_ADDR_TYPE arrays available for the name service provider to return.

Output

repObjectId Object ID of object name on first resolved address.

repSession ServiceTypeNCP session protocol module ID.

repTranAddrBuf Contains TRAN_ADDR_TYPE array(s) of address information for resolved object name.

repTranAddrCount

The number of TRAN_ADDR_TYPE arrays copied into reqTranAddrBuf.

reqNSSpec See NDS_RESOLVE_INFO structure in the input parameter list.

If the specified *repResolveType* will contain the

resolved name union tag specifying what the name was resolved to. If the *repResolveType* is DS_RESOLVED_OFFSET then *resolvedOffset* will contain the offset. If the *repResolveType* is DS_ENTRY_AND_REFERRALS then *repFlags* will contain the result flags.

Return values

SUCCESS_CODE
INVALID_CONNECTION
RESOLVE_SVC_FAILED
DS network errors

See also

NDSResolveNameToAddress

NDSResolveNameToAddress

Description

Resolves the supplied name to a transport address(es).

```
UINT32
NDSResolveNameToAddress(
    CONN_HANDLE reqConnId,
    SPECT_DATA          *treeName,
    SPECT_DATA          *reqNameType,
    UINT32              reqTranType,
    VOID                *reqNSSpec,
    UINT32              *repSessionServType,
    TRAN_ADDR_TYPE      *repTranType,
    UINT32              *repTranAddrCount)
```

Input

reqConnId Connection handle where name might be resolved. This value may be NULL. Ignored.

treeName Pointer to the tree name to be resolved.

reqNameType Pointer to the object type (CLASS) the *unserName* belongs to. NOTE: See list of object types *Chapter 9B: NDS Structures and Definitions.* Ignored.

reqTranType The preferred or required transport type to be used, as follows:

TRAN_TYPE_IPX
TRAN_TYPE_TCP
TRAN_TYPE_WILD

repTranAddrCount

Pointer to the maximum number of TRAN_ADDR_TYPE arrays available for the name service provider to return.

Output

repSessionSerType NCP session protocol module ID.

repTranAddrBuf Contains TRAN_ADDR_TYPE array(s) of address information the name was resolved to.

repTranAddrCount
Contains the number of TRAN_ADDR_TYPE arrays copied into *reqTranAddrBuf*.

reqNSSpec Name service-specific information. Ignored.

Return values

SUCCESS_CODE
INVALID_PARAMETER
RESOLVE_SVC_FAILED

See also NDSResolveObjectToId

NDSResolveIdToObject

Description

Resolves the supplied object ID to an object name and transport address.

Syntax

```
UINT32
NDSResolveIdToObject (
    CONN_HANDLE reqConnId,
    UINT32          objectId,
    VOID            *reqNSSpec,
    SPECT_DATA      *repObjectName,
    SPECT_DATA      *repObjectType)
```

Input

reqConnId Server on which to resolve the given *objectId*.

objectID Object ID to resolve

reqNSSpec Name service data. Ignored.

Output

repObjectName Pointer to structure into which to return *objectName*.

repObjectType Pointer to structure into which to return *objectID*. This will be the default class of the object.

Return values

SUCCESS_CODE
INVALID_CONNECTION
INVALID_PARAMETER
OUT_OF_CLIENT_MEMORY
MORE_DATA_ERROR
DS network errors

NDSGetDefaultNameContext

Description

Returns the current default name context for the specified tree, process group, and process.

Syntax

```
UINT32
NDSGetDefaultNameContext(
    UINT32          processGroupId,
    UINT32          processId,
    SPECT_DATA      *treeName,
    SPECT_DATA      *dncData
    UINT32          *dncType
```

Input

processGroupId Process group ID.

processId Process ID.

treeName Pointer to the tree name to retrieve the default name context for.

Output

dncData Pointer to the buffer filled with the default name context for the specified tree, process group, and process.

NOTE: This string is not intentionally NULL terminated.

dncType The type of string being returned. This value was set by **NDSSetDefaultNameContext**:

SPECT_DATA_ASCII
SPECT_DATA_UNICODE

Return values

SUCCESS_CODE
INVALID_PARAMETER
MORE_DATA_ERROR

Remarks

If a default name context has not been set up for this tree and process group, process, and the specified tree name matches the

NET.CFG-configured preferred tree, then the NET.CFG-configured default name context will be returned; otherwise, a NULL string will be returned.

If the tree name is NULL and a preferred tree has been set for this process group and process, the preferred tree will be used. If a tree structure does not exist for the specified tree name but the tree matches the NET.CFG-configured tree name the NET.CFG default name context will be returned; otherwise a NULL string will be returned.

If MORE_DATA_ERROR is returned, as much of the default name as possible was copied to *dncData*, but there was more to copy.

See also NDSSetDefaultNameContext

NDSSetDefaultNameContext

Description

Sets the current default name context for the specified tree, process group, and process.

Syntax

```
UINT32
NDSSetDefaultNameContext(
    UINT32          processGroupId,
    UINT32          processId,
    SPECT_DATA      *treeName
    SPECT_DATA      *dncData
    UINT32          DNCType)
```

Input

processGroupId Process group ID.

processId Process ID.

treeName Pointer to the tree name to set the default name context for.

dncData Pointer to default name context to set.

Output

None.

Return values

```
SUCCESS_CODE
    INVALID_PARAMETER
    OUT_OF_CLIENT_MEMORY
```

Remarks

If a tree name is not specified, the preferred tree for this process group and process will be used. If a preferred tree for this process group and process has not been established, the system default (NET.CFG) preferred tree will be used. If there is no system default preferred tree, then the system default (NET.CFG) default name context will be modified.

See also

NDSGetDefaultNameContext

NDSRequestReply

Description

Provides NDS request/reply functionality for applications and NLMs.

Syntax

```
UINT32
NDSRequestReply(
    CONN_HANDLE connHandle,
    UINT32      flags,
    UINT32      ndsVerb,
    UINT32      sendFragCount,
    FRAG_TYPE   *sendFragList,
    UINT32      recvFragCount,
    FRAG_TYPE   *recvFragList,
    UINT32      *actualRecvLen )
```

Input

connHandle Connection handle on which to send/receive NCP requests.

flags NCP_C_RequestReply error handler flags

0x00000000	Use default error handler mode
0x00000001	Return network errors to the caller
0x00000002	Handle network errors in request/reply

ndsVerb NDS Verb to be executed.

sendFragCount
Number of send fragments to send.

sendFragList Points to array of FRAG_TYPE structures describing NCP buffers to send.

recvFragCount
Number of receive fragments to receive reply into.

recvFragList Points to array of FRAG_TYPE structures describing how to receive response into buffers.

Output	*actualRecvLen*	Actual length of NCP reply packet received from server. Only valid on successful completion of request/reply. Buffer fragments described in recvFragList filled with server response if request/reply successful.

Return values SUCCESS_CODE

Remarks NDS request/reply fragments can be larger than 576 bytes, which is the standard maximum packet size for traversing bridges/routers.This routine will perform any necessary fragmentation required to send/receive NDS request/replies to or from the server.

NDSVLMSetUserObjectId

Description Sets the user's directory object ID used during authentication to the specified connection.

Syntax

```
UINT32
NDSVLMSetUserObjectId(
    UINT32          processGroupId,
    UINT32          processId,
    CONN_HANDLE reqConnId,
    UINT32          dataSize,
    UINT32          *data)
```

Input

processGroupId Process group ID.

processId Process ID.

reqConnId Connection handle authenticated with user's object ID.

dataSize Size of object ID.

data Pointer to user object ID to set.

Output None.

Return values

SUCCESS_CODE
INVALID_CONNECTION

NDSVLMGetSetMonitored

Description

Sets or gets the monitored connection for this process group and process.

```
UINT32
NDSVLMGetSetMonitored(
    UINT32              processGroupId,
    UINT32              processId,
    CONN_HANDLE *reqConnId,
    UINT32              operationFlag)
```

Input

processGroupId Process group ID.

processId Process ID.

reqConnId Pointer to connection handle to set/get as the monitored connection. NOTE: DomainName within the connection table specifies the tree name the monitored connection is being set for.

operationFlag Specifies either SET_MONITORED or GET_MONITORED

Output

reqConnId The monitored connection handle when GET_MONITORED is specified and return value is success.

Return values

SUCCESS_CODE
INVALID_PARAMETER
OUT_OF_CLIENT_MEMORY

NDSVLMReadTDS

Description

Reads the TDS information based on the current monitored connection for this process group and process.

Syntax

```
UINT32
NDSVLMReadTDS(
    UINT32    processGroupId,
    UINT32    processId,
    UINT32    tdsTag,
    UINT32    tdsOffset,
    UINT32    *dataLength,
    UINT8     *data)
```

Input

processGroupId Process group ID.

processId Process ID.

tdsTag NDS_AUTHENTICATION_TAG is the only valid tag.

tdsOffset Offset within TDS to begin reading from.

dataLength Pointer to the maximum number of bytes that can be copied.

Output

dataLength Pointer to the number of bytes copied.

data Contains the TDS information.

Return values

SUCCESS_CODE
TDS_INVALID_TAG
INVALID_PARAMETER

See also

NDSVLMWriteTDS

NDSVLMWriteTDS

Description

Writes the TDS information based on the current monitored connection for this process group and process.

Syntax

```
UINT32
NDSVLMWriteTDS(
    UINT32    processGroupId,
    UINT32    processId,
    UINT32    tdsTag,
    UINT32    tdsOffset,
    UINT32    dataLength,
    UINT8     *data)
```

Input

processGroupId Process group ID.

processId	Process ID.
tdsTag	NDS_AUTHENTICATION_TAG is the only valid tag.
tdsOffset	Offset within TDS to begin writing to.
dataLength	Number of bytes to write to the TDS.
data	Contains the TDS information.

Output

None.

Return values

SUCCESS_CODE
TDS_INVALID_TAG
INVALID_PARAMETER
TDS_WRITE_TRUNCATED
OUT_OF_CLIENT_MEMORY

See also

NDSVLMReadTDS