Appendix 7B
Name Service Multiplexor Constants and Definitions

Constants

The following constant definitions can be found in header file NAME_SVC.H included with NIOS. These constant definitions are used by the name service multiplexor and the name service providers in order to implement the name service interface defined in this document.

Name Service Types

```
#define   NAME_SVC_ANY            0x00000000
#define   NAME_SVC_BINDERY_ID     0x00000001
#define   NAME_SVC_NDS_ID         0x00000002
#define   NAME_SVC_PNW_ID         0x00000003
#define   NAME_SVC_WILD           0x80000000
```

String Types

```
#define   SPECT_DATA_ASCII    0x00000001
#define   SPECT_DATA_UNICODE      0x00000002
```

Transport Types

```
#define   TRAN_TYPE_IPX           0x00000001
#define   TRAN_TYPE_TCP           0x00000002
#define   TRAN_TYPE_WILD          0x80000000
```

Service Types

```
#define   SVC_TYPE_NCP_SERVER     "NCP_SERVER"
```

Object Types

```
#define   USER_OBJECT_TYPE        "USER"
#define   USER_GROUP_OBJECT_TYPE      "GROUP"
#define   PRINT_QUEUE_OBJECT_TYPE     "QUEUE"
#define   NCP_SERVER_OBJECT_TYPE      "NCP_SERVER"
```

Structure Definitions

The following structure definitions can be found in header file NAME_SVC.H included with NIOS. These structure definitions are used by the name service multiplexor and the name service providers in order to implement the name service interface defined in this document.

SPECT_DATA

The data structure for specifying a string in either Unicode or in local code page.

```
typedef struct {
   UINT32   Length;
   UINT8    *Data;
   UINT32   DataType;
   UINT16   LocalCodePage;
   UINT16   CountryCode;
} SPECT_DATA;
```

Fields:

Length	Length of name pointed to by *name*.
Data	Pointer to a string that can be encoded in either Unicode or in a local code page.
DataType	Specifies whether *name* is encoded in Unicode or in the local code page. Must be one of the following values: SPECT_DATA_ASCII SPECT_DATA_UNICODE
LocalCodePage	Decimal value of local code page if *string* is of type SPECT_DATA_ASCII. A value of zero means to use the default local code page.
CountryCode	Decimal value of country. A value of zero means to use the default local code page.

TRAN_ADDR_TYPE

The data structure definition for a transport address returned by a name service provider.

```
typedef struct {
   UINT32    transportType;
   UINT32    transportLen;
   UINT8     transportAddr[32];
} TRAN_ADDR_TYPE;
```

Fields:

transportType	Type of transport address returned (for example, IPX or TCP).
transportLen	Length of returned transport address.
transportAddr	Buffer that contains the transport address. (It is assumed that 32 bytes is large enough to hold any transport address to be used by this interface).

NAME_SVC_DESC_BLOCK

Describes the data structure that a name service provider registers with the name service multiplexor that further describes the name service provider being registered. This information can be obtained by other NLMs by calling **NSMEnumerateNameSvc**.

```
typedef struct {
    UINT8      majorVersion;
    UINT8      minorVersion,
    UINT8      revision;
    UINT8      name[13];
    UINT8      description[80];
    UINT32     nameSvcID;
} NAME_SVC_DESC_BLOCK;
```

Fields:

majorVersion Major version of this name service provider.

minorVersion Minor version of this name service provider.

revision Revision of this name service provider.

name ASCIIZ name of this name service provider.

description ASCIIZ description of this name service provider.

nameSvcID Unique name service ID assigned to this name service provider.

NAME_SVC_API_SET_TYPE

The following functions must be implemented by a name service provider to be compatible with the name service interface described in this document. A name service provider will register these functions with the name service multiplexor by calling the service **NSMRegisterNameSvc**.

```
typedef struct {
    UINT32     (*NSPGetPreferredName)  (
                   UINT32           processGroupID,
                   UINT32           processID,
                   SPECT_DATA       *name);

    UINT32     (*NSPSetPreferredName)  (
                   UINT32           processGroupID,
                   UINT32           processID,
                   SPECT_DATA       *name);

    UINT32     (NSPResolveNameToAddress)   (
                   UINT32           processGroupID,
                   UINT32           processID,
                   CONN_HANDLE      connHandle,
                   SPECT_DATA       *objectName,
                   SPECT_DATA       *objectType,
                   UINT32           transportType,
                   VOID             *nameSvcSpec,
                   UINT8            *repSessSvcID,
                   TRAN_ADDR_TYPE   *repTranAddr,
                   UINT32           *repTranAddrCount );

    UINT32     (NSPResolveObjectToID)  (
                   UINT32           processGroupID,
                   UINT32           processID,
                   CONN_HANDLE      connHandle,
                   SPECT_DATA       *objectName,
                   SPECT_DATA       *objectType,
                   UINT32           transportType,
                   VOID             *nameSvcSpec,
                   UINT32           *repObjectID,
                   UINT8            *repSessSvcID,
                   TRAN_ADDR_TYPE   *repTranAddr,
                   UINT32           *repTranAddrCount );

} NAME_SVC_API_SET_TYPE;
```

Return Codes

Following are the codes that can be returned by the Name Service Multiplexor/Providers that implement the Name Service Interface.

Code	Meaning
SUCCESS_CODE	Operation completed successfully.
NAME_SVC_NOT_REGISTERED	Specified name service provider is not registered with the name service multiplexor.
NAME_SVC_ALREADY_REGISTERED	Specified name service provider is already registered with the name service multiplexor.
RESOLVE_NAME_FAILED	No name service provider could resolve the supplied name to a network address.
RESOLVE_OBJECT_FAILED	No name service provider could resolve the supplied object name to an object ID.
INVALID_PARAMETER	Supplied input/output parameter is not valid for the operation being performed.
MORE_DATA_ERROR	Output buffer is not large enough to receive results of operation being performed.