FinsGateway

Network Service Provider SDK Manual

(FinsGateway Communication Unit Addition)

First Edition

Novenber 1999
OMRON Corporation

Contents

1 Network Service Provider SDK

The FinsGateway Network Service Provider SDK (NSP SDK) explains the procedure for developing Network Service Providers (NSPs), which enable FinsGateway (and applications running on FinsGateway) to communicate with the network. These NSPs are called communication units in FinsGateway.

FinsGatewayhas already been provided with NSPs for Ethernet, SYSMAC LINK, Controller Link, Serial, SYSMAC Board, SYSNET, and CompoBus/D. In addition to these networks, other NSPs for other networks can also be developed using the NSP SDK.

2 Copyright

Everything described or expressed in this FinsGateway Network Service Provider SDK (including documentation, API, electronic file, etc.) is copyrighted and owned by OMRON Corporation.

This FinsGateway Network Service Provider SDK is provided by OMRON Corporation to the user under special conditions. Therefore, the user is not allowed to divulge any of this information in any way, shape, or form, beyond the extent of this agreement, for any reason.

Any of the information concerning the SDK may be changed by OMRON Corporation without notice.

Any software developed using this SDK is not permitted to be marketed in competition with any product of OMRON Corporation.

3 Network Service Provider Concept

FinsGateway was developed to achieve the objective of interfacing with various networks through a single, unified API.

In the past, each network driver software had its own, individual API. This meant that applications had to provide programs for each network, with its distinct API. Each network also had its own protocol, meaning that applications had to treat the networks separately in this regard, as well.

FinsGateway not only unifies the API across various networks, but also unifies the communication protocol by using FINS. This enables the development of applications that do not rely heavily on the network. The part that does rely on the network is contained within the NSP (Network Service Provider), in FinsGateway.

The structure of FinsGateway and the NSP is described as follows:

- The NSP and the application are symmetrical. The NSP implementation method is normally identical with a FinsGateway application development. An NSP can therefore be implemented simply by learning to use FinsGateway with some proficiency.
- The network-dependent portions are contained within the NSP. The network-specific aspects can be made to appear identical to other NSP interfaces, or provided to the application as is.
- FinsGateway provides message queuing. Therefore, it is not necessary to provide queuing within the NSP.
- FinsGateway provides message routing. Therefore, it is not necessary to provide routing between networks within the NSP.
- Data link communication can be provided through the FinsGateway EventMemory. Using the EventMemory means that the NSP can use the memory allocation, exclusive control, and event notification already being provided by FinsGateway.
- Adding or removing an NSP is a simple matter. The FinsGateway and NSP structure makes them independent of each other. Adding or removing an NSP does require rebuilding FinsGateway. Adding a network to FinsGateway simply requires installing the NSP, and starting it.

4 Development Environment

The following items are required for NSP development:

- Windows NT/9x
- Visual C++5.0, or newer
- FinsGateway Runtime Edition
- FinsGateway SDK

The existing FinsGateway NSPs were developed with Microsoft Visual C++ 5.0. These implementations were done in C-language, using the standard C library and Win32. The fsvnavim extension uses C++ and the MFC library.

5 NSP Implementation

The following technical considerations are essential for designing and implementing an NSP:

- Win32 platform C/C++ programming
- Win32 Service programming
- General system programming, processing, threads, exclusive control, registry
- FINS protocol general knowledge about FINS service FINS address
- FinsGateway programming, FINS message communication, EventMemory
- System error handling, error log, etc.

6 Network Service Provider Internal Structure

Here we explain how the FinsGateway NSP supports the FINS protocol and the network protocol, and what the NSP has to do to support both.

6.1 Overall Configuration

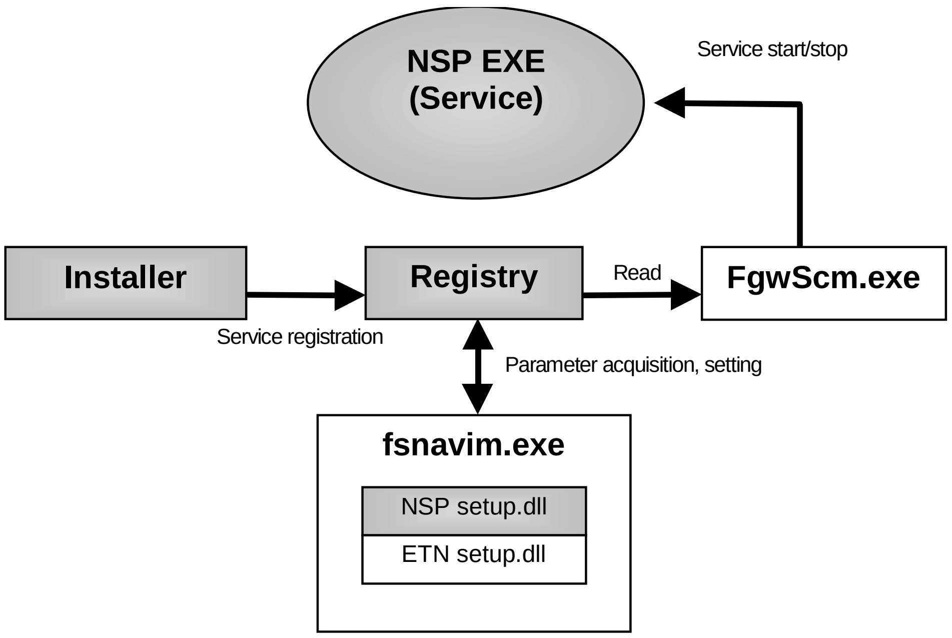
An NSP is configured of the following elements:

- The NSP to communicate with the device in its dedicated protocol
- The installer to register the NSP as a service

  It is registered in the registry as a FinsGateway service program
- The Service Control Manager (SCM) to start/stop the NSP

  The SCM obtains the parameters from the registry, and starts/stops the NSP.
- The setup.dll used by the Setting Utility, fsnavim.exe (Fsnavim)

  Fsnavim uses the setup.dll for each NSP to obtain/change the parameter settings.



To develop an NSP, the following items must be created:

1. The NSP.EXE to start/stop the service
2. The setup.dll to change the NSP communication settings
3. The installer
4. The registry settings

6.2 NSP Implementation

Here, we will use the sample program included with the NSP to explain the implementation.

In the sample program main process, the system service is implemented.

The main process creates two threads. The message service is implemented in the two threads.

6.2.1 NSP System Service Implementation

The system service must include the implementation of the SCM start/stop requests, event log record notification, etc.

The sample program was implemented according to the following flow:

1. Service processing registration

Registers the service main function when the NSP receives the service start request from the SCM

2. Event log initial processing

Creates the event log for error notification when the NSP detects any kind of error.

3. FinsGateway port initial processing

Opens the port to receive FinsGateway messages, and registers the local unit to FinsGateway as a communication unit.

4. Network port initial processing

Opens the port to receive network messages.

5. FINS receive thread creation

Creates the thread to receive FinsGateway messages.

6. Network receive thread creation

Creates the thread to receive network messages.

7. Wait for service stop

The primary thread is in a wait state until it receives the stop request from the SCM.

6.2.2 Message Service Implementation

When the application needs to communicate with a device on the network, the application uses the FinsGateway FINS messages.

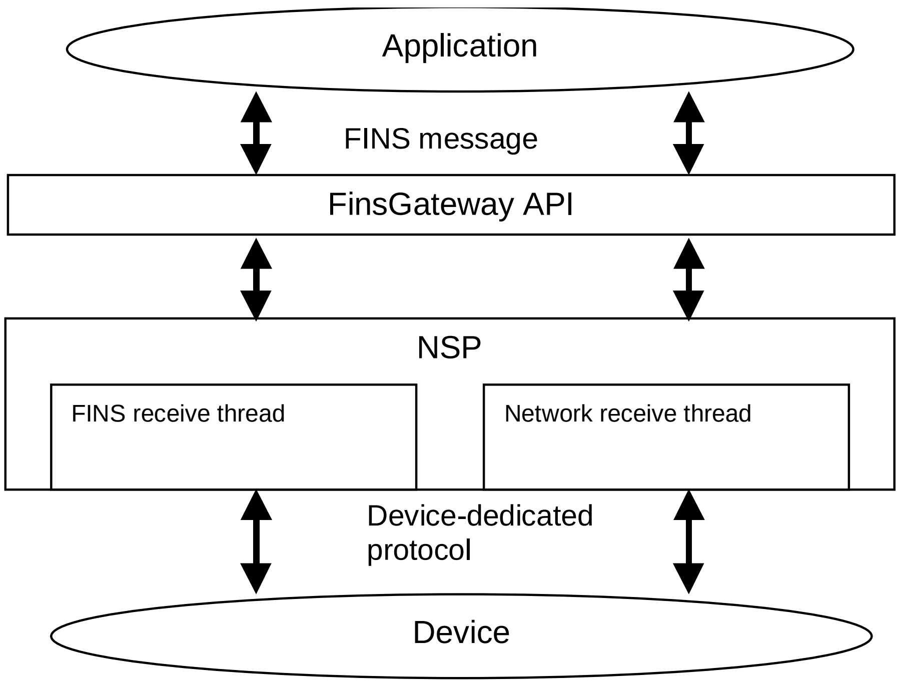
The NSP receives a FinsGateway, or network message, converts that message to the actual communication protocol, or a FINS message, then sends it to the device.

A message sent from the application is passed to FinsGateway, and FinsGateway passes it to the NSP corresponding to the FINS message network address.

The NSP receives the FINS message from the application with the FinsGateway message receive thread (FINS receive thread), converts it to the device-specific protocol, and sends the message to the network.

If the FINS message network address is for the NSP itself, it executes the service corresponding to the FINS message, and sends the response to the application that sent the message.

A message from the network is received with the NSP network receive thread, converted from the device-specific protocol to a FINS message, and sent to the application. If it is for the NSP itself, the service is executed, and the response sent to the network.



6.3 Registry and Other System Settings

The NSP operation settings are saved in the registry.

The registration to the registry is performed at installation by the installer.

The registry key specifying the NSP operation is stored in the following location:

¥¥ HKEY_LOCAL_MACHINE¥SOFTWARE ¥OMRON¥FinsGateway¥Fgwscm

Sets the NSP names enumerated to the FinsGateway SCM.

The following values are set as the registry entries for these keys. The entry shown here is the one required to be set by the developer.

Entry Name	Form	Meaning
NSP name	REG_SZ	Service name displayed in the FinGateway SCM.

¥¥ HKEY_LOCAL_MACHINE¥SOFTWARE ¥OMRON¥FinsGateway¥Fsnavim¥EnhanceDll

Sets the Setup.dll for Fsnavim for each NSP.

The following values are set as the registry entries for these keys. The entry shown here is the one required to be set by the developer.

Entry Name	Form	Meaning
NSP name	REG_SZ	Specifies the full path of each service Setup.dll file used by Fsnavim. Environment variables can be used.

¥¥ HKEY_LOCAL_MACHINE¥SOFTWARE ¥OMRON¥FinsGateway¥NetWorkProvider¥NSPName

Sets the data for each service.

The following values are set as the registry entries for these keys. The entries shown here are the ones required to be set by the developer.

Entry Name	Form	Meaning
ProductType	REG_DWORD	Indicates Runtime edition, or Embedded edition. Runtime edition=0x02, Embedded edition=0x01
ReleaseDate	REG_SZ	Indicates the release date of the NSP. The format is MM/DD/YY.
Version	REG_SZ	Indicates the NSP version. The format is x.xx.

¥¥ HKEY_LOCAL_MACHINE¥SOFTWARE ¥CurrentControlSet¥Services¥NSPName

Sets the service names enumerated to the SCM.

The following values are set as the registry entries for these keys. The entries shown here are the ones required to be set by the developer.

NSP Name Entry Name	Meaning
Display Name	Service name displayed in the SCM. REG_SZ form
ImagePath	Specifies the full path of the NSP file. Environment variables can be used. REG_SZ form
Start	Specifies the start mode (auto/manual) of the service at log-in. REG_DWORD form

¥¥ HKEY_LOCAL_MACHINE¥SYSTEM ¥CurrentControlSet¥Services¥EventLog¥System

Sets the service names given to the event.

The following values are set as the registry entries for these keys. Only the main entry is shown here.

NSP Name Entry Name	Meaning
EventMessage File	Indicates the path of the service program that gives the log to the event log. REG_SZ form

6.4 Setting Utility

In FinsGateway, the properties for each NSP network are set using the fsnavim.exe (Fsnavim) utility. Fsnavim is a utility (with GUI) that performs the network number setting, relay network setting, and the properties for each network.

Fsnavim uses the NSP Setup.dll to set the properties for each network. The properties for each network set here include the unit number of the communication unit, node number, node definition, data link interval, etc.

When developing an NSP, the Setup.dll must also developed at the same time.

By developing a DLL for each network with the properties specific to that network, the following advantages are gained:

- It is never necessary to update the Fsnavim itself, as a result of adding/removing a network.
- The Fsnavim and the network property DLL can be implemented simultaneously.
- Other applications can also use the same network property DLL,

7 System Service

The NSP is a service program, and runs in the background. The NSP start/stop and state checking are handled by the SCM. The NSP therefore notifies the SCM at each start/stop state change.

Since the SCM function is only available in Windows NT, FinsGateway provides a Windows NT/9x service manager (FgwSCM). This binary compatibility for both is enabled with the Fsport.dll library. By using the Fsport.dll, service programs can be developed with binary compatibility for Windows NT/9x.

8 Event Log and UDM

An event log corresponding to Win32 is used to report errors detected by the NSP. The NSP status is monitored by the UDM (Universal Data Monitor). The event log has a system-wide, common record area which stores the logs reported from the various sources. This is generally used for recording the OS-level error states. The UDM has record area divided into selectable categories. Each record area stores the logs reported by the source specified for that area. This is generally used for tracing the operation of the specified source. As an example of using the event log and the UDM, the SerialUnit notifies the event log of operation failures. The data being sent through the SerialUnit is being traced by the UDM.

Using the event log and UDM to report is not a necessity of the NSP. Implementation of this function is optional. However, for system service debugging, or troubleshooting, implementation of this log function is recommended.

8.1 Event Log

Errors that occur during NSP operation need to be notified to the user. However, since the NSP operates as a system service, notifying the user of errors with a window is not always appropriate. For the NSP to notify the user, the Win32 event log is used.

For an event log model and operation details, refer to the Win32 Event Logging OverView. The event log characteristics in brief are as follows:

- Dividing the events broadly, they can be classified as System, Security, and Application.
- The event types are Errors, Warnings, and Information.
- The exact event message expressions are included in the files including the resources compiled by the Message Compiler (MC).

8.2 UDM

The UDM was designed as a general data monitor, and is exclusive to FinsGateway. Using the UDM is helpful for recording changes in the NSP status, debugging, and tracing execution during system operation. The UDM characteristics in brief are as follows:

- There is a log file for each category.
- When logging, the LDH (Log Data Handler) filter DLL can be specified.
- The detailed log display can be done with the specified LDH conversion DLL.
- Log start/stop can be specified from an application (normally a viewer application).

9 Setting Program Addition

Set all the network properties for the NSP with the Fsnavim.exe (Fsnavim) command. Fsnavim reads each network property setting DLL (Setup.dll), and performs the settings for all the network properties.

The interface between Fsnavim and the Setup.dll is pre-determined. This interface consists of the interface from Fsnavim to the Setup.dll, the interface from the Setup.dll to Fsnavim, and the data used by the interfaces. This is all defined in the C-language header file, fsnaviif.h.

9.1 Interface from Fsnavim to Setup.dll

The function entry name the Setup.dll must export is as follows. The entry number does not need to be defined:

- EXTERN_C EXPORT BOOL WINAPI newDll (pNet netDef, PNETINFO netInfo)

The Setup.dll must have the following three interfaces to be called by Fsnavim:

Function Name	Meaning
newDll	Initializes the DLL, and sets up the interface data.
doDialog	Displays the modal dialog to set the properties.
deleteDll	Unloads the DLL.

newDLL

Fsnavim calls the Setup.dll newDLL function entry, which obtains and changes the settings.

The Setup.dll must perform the following within the newDLL function:

- Determine whether or not the network specified by Fsnavim in the PNETINFO data is a network that it should handle. (This is the current specification. This specification is to be changed in the next version.)
- Set the DLL interface function address to the pNet data -> dllProvideProcs.
- Set the general name of the network being recognized by the DLL to the pNet data -> typeName.

doDialog

Fsnavim calls the corresponding Setup.dll doDialog function when the user selects a specific network property.

The Setup.dll doDialog function creates the modal dialog, displays the property dialog to the user, receives the setting data input by the user, and sets the properties accordingly. Control does not return to Fsnavim until the doDialog function is finished.

deleteDLL

Fsnavim calls the Setup.dll deleteDLL function when the user specifies to exit the program.

The Setup.dll performs the DLL deletion processing within the deleteDLL function, when necessary.

9.2 Interface from Setup.dll to Fsnavim

Fsnavim has the following interfaces for the Setup.dll:

getByNetId	Obtains the network data registered to the system, using the network number.
getByUnitId	Obtains the network data registered to the system, using the unit number.

Fsnavim manages the network data (network number, node number, unit number) registered to the entire system. The two interfaces described above provide the means of accessing the management data from the DLL.

These interfaces provide the method of checking for network number, unit number duplication in the system when the user changes the network number, or unit number using the DLL doDialog.

9.3 Data Structure

The data used by the interface between Fsnavim and the Setup.dll is as follows:

```
typedef struct TagNet{
          pNet next;                 //Pointer to next network data
          NETID netId;               //FINS network number
          tNetworkTypeCode netType;           //Network type
          BOOL isMyNet;              //Local/Relay node
          union {
                  struct {
                    NODEID nodeId;   //Local node number
                    UNITID unitID;   //Local unit number
                  } mNet;
                  RERAY relayNet;    //Relay node
          } u;
          CHAR typeName[40];
```

This data is managed by Fsnavim for each network, and is related to the items in the list box in the initial screen of Fsnavim.

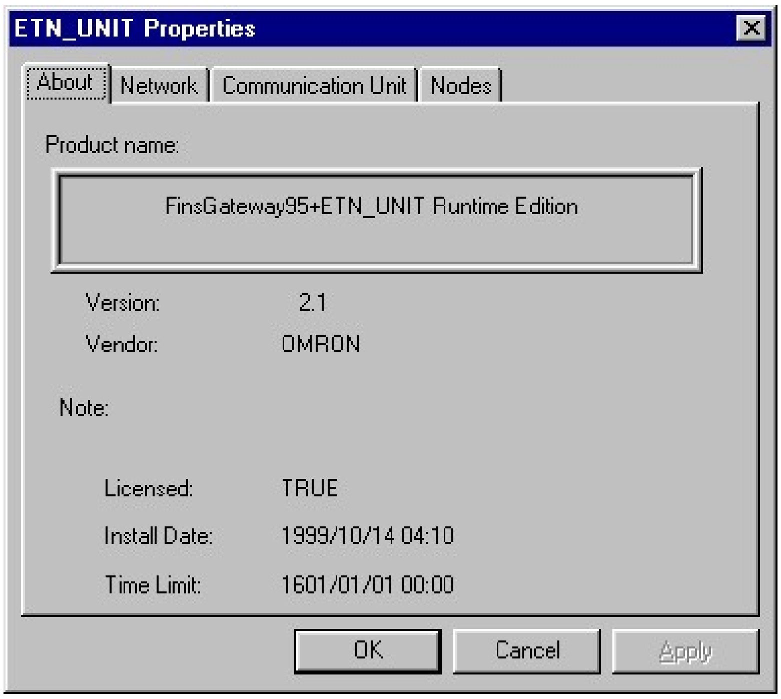
The data set by the DLL using newDLL is dllProvideProcs and typeName. The dllProvideProcs is the interface from Fsnavim to the Setup.dll.

The specifiedData can store the data that is set freely by the DLL. When using the specifiedData to store memory allocation data (Alloc), be sure to release the memory using the deleteDLL function.

The interface from the Setup.dll to Fsnavim is stored in the exeProvideProcs.

9.4 Setup.dll GUI Guideline

The modal dialog displayed by the Setup.dll is created as a property sheet. If there is no particular problem the property sheet tab order is as follows. This consistent among all the NSPs. The example shown here is the ETN_UNIT dialog. The first tab is the About tab, showing the product information:



Product name:

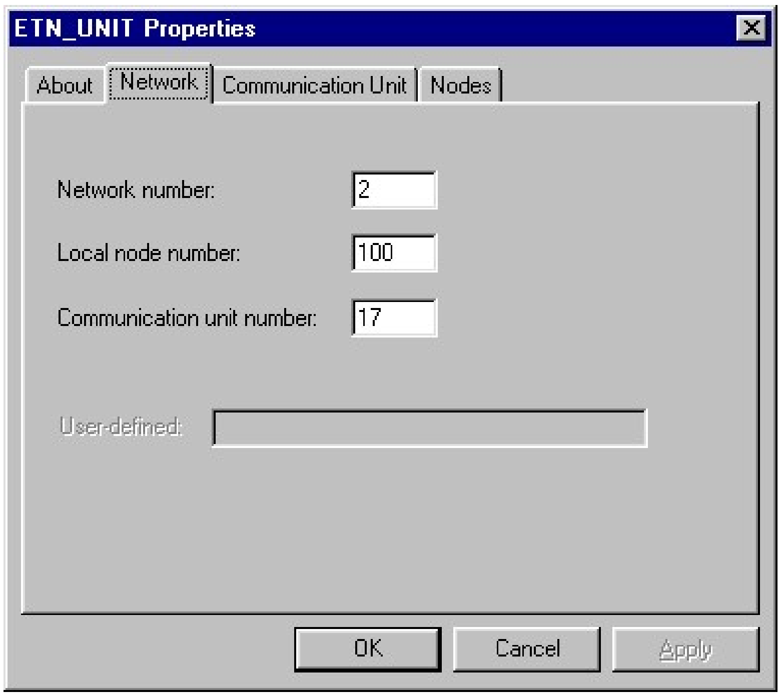
FinsGateway95+ETN_UNIT Runtime Edition

Version:	2.1
Vendor:	OMRON

Note:

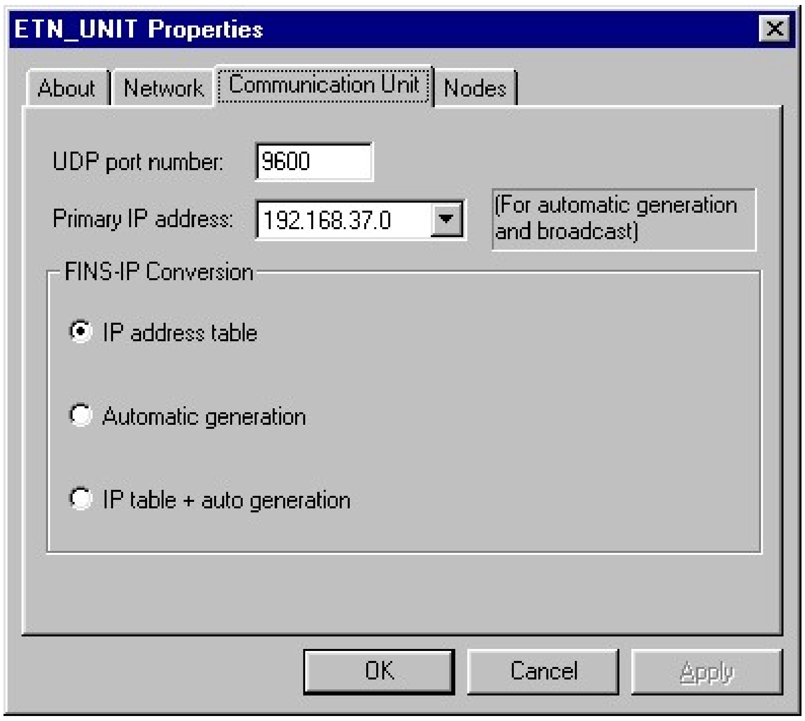
Licensed:	TRUE
Install Date:	1999/10/14 04:10
Time Limit:	1601/01/01 00:00

The second tab is the Network tab, showing the network settings. This is used the set FINS address of the NSP itself. Set the data shown in the Network tab as follows:



Network number:	2
Local node number:	100
Communication unit number:	17
User-defined:	

The third tab is the Communication Unit tab, showing the communication unit settings. This used to set data required for the operation of the NSP itself. The following example shows this tab for the ETN_UNIT:



ETN_UNIT Properties

About | Network | Communication Unit | Nodes

UDP port number: 9600

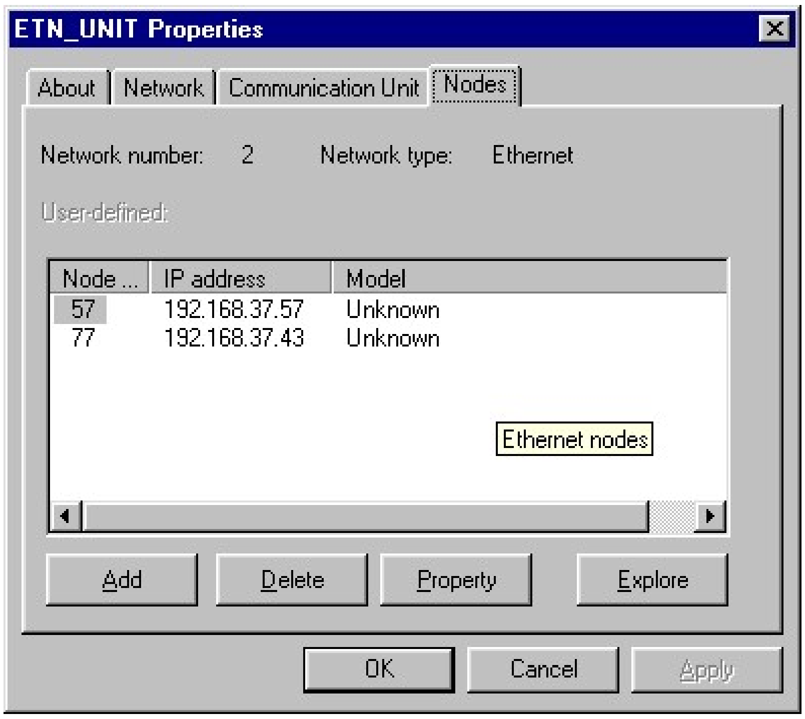
Primary IP address: 192.168.37.0

(For automatic generation and broadcast)

FINS-IP Conversion

- [✓ IP address table]
- Automatic generation
- IP table + auto generation

The fourth tab is the Nodes tab, showing the node data. This is used to set the data for the other nodes with which to perform FINS communication. It is strongly recommended also to provide a listing for the nodes of other networks that cannot be set with this dialog (SYSMAC LINK, etc.).



ETN_UNIT Properties

About | Network | Communication Unit | Nodes

Network number: 2 Network type: Ethernet

User-defined:

Node ...	IP address	Model
57	192.168.37.57	Unknown
77	192.168.37.43	Unknown

Ethernet nodes

If there are more settings that need to be made separately for each network, add more tabs after this. For example, the SYSMACLINK property dialog has a Hardware tab added to set the IRQ and I/O address.

9.5 Setup.dll Implementation

The following is an example of the Setup.dll implementation.

1. Create a new Visual C++ 5.0 MFC AppWizard (dll) project.

- Use the _-_MFC common DLL for the DLL type.
- Create it without the functions supported by the DLL (automation, socket).

2. Create the functions exported by the DLL.

3. Copy the property sheet and necessary property pages from the sample Setup.dll project. (The detailed procedure is described below.)

4. The simplest way to copy is as follows:

- Open the existing DLL project in Visual C++, and register the property sheet class, and property page class to the component gallery.
- Open the new project, and add the classes registered to the component gallery to the project to be implemented. By doing this, the class implementation and resources are automatically added to the project.

5. Proceed with the implementation based on the actual software.

Note: Creation or addition of a new property page uses the resource, IDD_PROPPAGE_LARGE from Dialog.

10 Installer

The installer registers to the registry the network settings initial values necessary for the NSP to communicate with the devices. It also registers each NSP name and NSP installation directory, and each NSP Setup.dll installation directory, necessary for the service program to be started by the service control manager. The created NSP can be used from any folder.

Refer to 6.3

Registry and Other System Settings for further details about the data registered to the registry by the installer.

The service control manager (SCM) and Fsnavim search for, and load the NSP file and NSP Setup.dll from the fixed entry values in the registry.

The NSP SDK provides the Fsauth library and command line program (Fsregist.exe), which registers the system to the registry and services, according to the FinsGateway system settings file.

11 NSP Development Precautions

The NSP is a system program operating as a system service, and uses the resources common to the system. Therefore, consideration must be given to the NSP design so that it does not adversely affect any other system services. The following are the points that must be specifically considered in the design of a system service:

- There is no user interface. The NSP does not provide the user with error data or other information in the form of a dialog box.
- The means of providing error data are limited. The capacity for providing the user with data through a window is limited, so error data and other information is normally provided through the event log or UDM (Universal Data Monitor). Windows NT provides an event log. FinsGateway provides the UDM and the Windows95 event log.
- Common services are provided for multiple processes. In other words, it is not a good idea to design a service for a specific application. It is necessary to design the service thinking that varying applications will make asynchronous requests.
- A reliable design is the highest priority. The user is not normally aware of the operation of a service. A service built into the system generally keeps running after it is started, until the system shuts down.

For OLE/COM programming, the following point has proven to be important:

- COM instances are created for each user. A COM object design that assumes there is only ever one instance in the system can therefore encounter unexpected errors. This occurs when the system service and the logged on user are different.

11.1 Troubleshooting and Precautions

The NSP developer must take sufficient precaution for handling potential errors. NSP communication involves quite a large number of elements. When an error occurs, it can therefore be quite challenging to troubleshoot the system. For this reason, it is best to design means of troubleshooting into the NSP from the beginning.

The event log and UDM are two of the best tools for troubleshooting, and debugging. Using these effectively should be included in the original design of the NSP. The following operations must be considered carefully:

- Messages sent from the application
- Messages received from the network
- Messages sent to the network
- Messages sent to the application
- Protocol conversion (send/receive)
- Communication procedures with other devices

11.2 Debugging

In normal systems, the NSP operates as a system service. However, for debugging, it is best to run the NSP execution file (EXE) alone. If the NSP is implemented not to operate as a service, but to be executed by double-clicking with the mouse, or specifying the command name, it can be operated as a console application and debugging is much simpler.

When the NSP is started, an MS-DOS prompt or command prompt will be displayed. When it is started in this manner, operations as a service become invalid, and starting/stopping from the service manager is impossible.

When starting an NSP as a console application, a means must be provided to stop it. Stopping it with the signal generated by pressing Ctrl+C on the keyboard is a common implementation.

The following precautions apply to debugging:

- When force stopping an NSP in Windows95 (other than by using Ctrl+C), the process to unload the DLL used by the NSP is not properly called. As a result, the FinsGateway unit remains in the used state, and the NSP cannot be started the next time. In this case, close all the applications using FinsGateway.
- The startup of an NSP implemented as a service for Windows NT takes about 5 seconds.
- When starting an NSP as a console application, do not use the service manager to start or stop the NSP.

12 FinsGateway API

This section describes the functions provided by FinsGateway. For more details about using the functions, refer to the online manual Fsport-related references, and Fsauth-related references.

12.1 Fsport Library

The Fsport library provides the following functions. Refer to the online reference for function details:

Event Log Functions

Function Name	Description
FsPort_RegisterEventSource	Returns the handle to record an event.
FsPort_OpenEventLog	Opens the event log handle.
FsPort_DeregisterEventSource	Closes the event log handle.
FsPort_CloseEventLog	Closes the event log handle.
FsPort_ReportEvent	Writes an entry to the end of the specified event log.
FsPortCompatibleVersion_get	Obtains the recorded compatible version.
FsPortCompatibleVersion_set	Records the compatible version.

System Service Functions

Function Name	Description
FgwScm_OpenSCManager	Connects to the service control manager.
FgwScm_CloseServiceHandle	Disconnects from the service control manager, or discards the service object.
FgwScm_OpenService	Obtains the handle to distinguish an existing service.
FgwScm_CreateService	Creates the service.
FgwScm_DeleteService	Deletes the specified service.
FgwScm_RegisterServiceCtrlHandler	Registers the service control handler.
FgwScm_QueryServiceStatusEx	Queries the execution status of a service.
FgwScm_ControlServiceEx	Controls service execution.
FgwScm_SetServiceStatus	Updates the service status data of the service control manager.
FgwScm_ChangeServiceConfig	Updates the service status data of the service control manager.
FgwScm_QueryServiceConfigByAlloc	Obtains the configuration parameters of the specified service.
FgwScm_getAutoRunStatusOfScm	Confirms whether or not the service manager is set start automatically.
FgwScm_setAutoRunStatusOfScm	Sets the automatic startup of the service manager.

12.2 Fsauth Library

The Fsauth library provides the following functions:

Function Name	Description
FsAuth_installRegistryAndService	Installs the registry and service to the system according to the FinsGateway system settings file.
FsAuth_getInstalledPath	Defines the registry and service to install to the system.
FsAuth_setInstalledPath	Searches for the product installation path.
FsAuth_createDirectory	Sets the product installation path.
FsSynch_startExclusiveApplication	Starts the application exclusively.
FsSynch_exitExclusiveApplication	Stops the exclusive execution between applications.
FsAuth_addFileTimeAnyDay	Adds number of days to the FILETIME format time.
FsAuth_getCurrentFileTime	Requests the system time in the FILETIME format.

The two threads created by the NSP primary thread, and performing the message service, are an infinite loop. Until they receive a service stop request from the SCM, they continue to receive messages from FinsGateway applications, or networks.

The message service is implemented in the following flow. Refer also to the figure that follows:

1. Receive a message.

A message is received from an application or a network.

2. Analyze the message received.

The FINS address of the message is interpreted, and it is determined whether it is for the NSP itself, or a network.

3. Execute the message service.

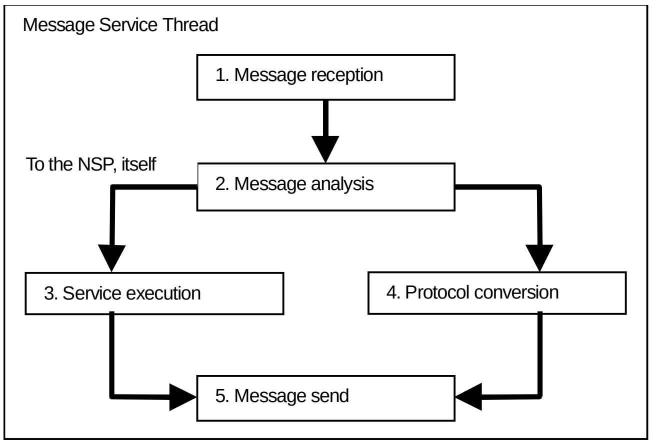
If the FINS address of the message is for the NSP itself, the service for that message is executed.

4. Convert the protocol.

If the FINS address of the message is for a network device, it is converted to the network-specific protocol.

5. Send the message.

The message is sent to the target device specified by the FINS address.


Message Service Thread
1. Message reception
To the NSP, itself
2. Message analysis
3. Service execution
4. Protocol conversion
5. Message send