บทที่ 4

การเขียนโปรแกรมและการเรียกใช้ API บนเน็ตแวร์

การเขียนโปรแกรมเพื่อใช้งานในระบบเครือข่ายเน็ตแวร์นั้น จะมีความแตกต่างจากการเขียนโปรแกรมบนเครื่องคอมพิวเตอร์ส่วนบุคคลเครื่องเดียวอยู่บ้าง โดยมีข้อที่ควรต้องคำนึงถึง ดังนี้

- การตรวจสอบว่า เน็ตแวร์ Shell ได้ถูกบรรจุเข้าสู่หน่วยความจำ
- การตรวจสอบระดับการรักษาความปลอดภัย (Security) ของผู้ใช้
- การติดต่อสื่อสารกันระหว่างแต่ละสถานีงาน
- การพิมพ์ในระบบเครือข่าย

1. การเข้าถึงแฟ้มข้อมูลพร้อมกันหลายผู้ใช้ (Multiuser File Access)

เนื่องจากในระบบเครือข่าย อาจมีผู้ใช้หลายคนที่ต้องการเข้าถึงแฟ้มข้อมูลเดียวกันพร้อมกันดังนั้นการจัดการเกี่ยวกับการเข้าถึงแฟ้มข้อมูลจึงมีความสำคัญ เพื่อจะทำให้ข้อมูลมีความถูกต้องไม่ซ้ำซ้อน และผู้ใช้ทุกคนมองเห็นข้อมูลเหมือนกัน มีวิธีในการจัดการแฟ้มข้อมูลหลักๆ 3 วิธี คือ

1.1. การล็อกแฟ้มข้อมูล (File Locking)

เป็นการป้องกันไม่ให้ผู้อื่นมาใช้แฟ้มข้อมูลนั้น โดยปกติแล้วจะไม่ใช้วิธีนี้ เนื่องจากจะทำให้ผู้อื่นไม่สามารถใช้แฟ้มข้อมูลนั้นได้ นอกจากในกรณีที่ต้องการแก้ไขข้อมูลทั้งหมดในแฟ้มข้อมูล

1.2. การล็อกเรคคอร์ด (Record Locking)

เป็นการป้องกันไม่ให้ผู้อื่นใช้เรคคอร์ดใดเรคคอร์ดหนึ่งของแฟ้มข้อมูล ส่วนผู้ใช้อื่นที่ใช้แฟ้มข้อมูลเดียวกัน สามารถทำงานกับเรคคอร์ดอื่นในแฟ้มข้อมูลนั้นได้ตามปกติ

1.3. การติดตามรายการเปลี่ยนแปลง (Transaction Tracking)

การที่ต้องการแก้ไขเปลี่ยนแปลงข้อมูลบางอย่างในแฟ้มข้อมูล ซึ่งอาจจะเป็น 1 หรือมากกว่า 1 เรคคอร์ด หรือ มากกว่า 1 แฟ้มข้อมูล และต้องการให้แน่ใจว่า ข้อมูลได้ถูกแก้ไขครบทุกเรคคอร์ด และ แฟ้มข้อมูลที่ต้องการ หรือ ถ้าไม่สำเร็จข้อมูลทุกเรคคอร์ดจะต้องไม่ถูกแก้ไข (Roll Back) เหมือนกัน

2. การอ้างถึงไดเรกทอรี

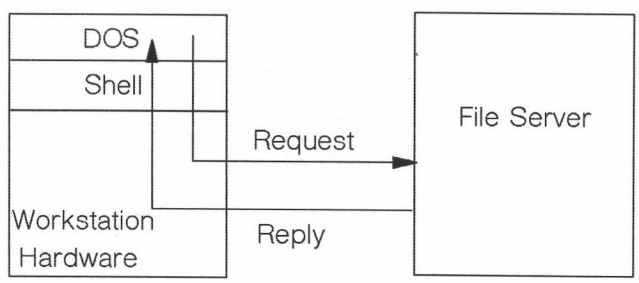
ในการเขียนโปแกรมไม่ควรอ้างถึงชื่อของ Volume ของเครื่องบริการแฟ้ม (File Server) โดยตรง เพราะจะทำให้เกิดข้อจำกัด เมื่อมีการเปลี่ยนแปลงแก้ไข Configuration ต่างๆ ของเครื่องบริการแฟ้ม ควรจะใช้ชื่อเสมือนของไดร์ฟโดยใช้คำสั่ง MAP ของ เน็ตแวร์ หรือให้โปรแกรมไปอ่านค่าตัวแปรสิ่งแวดล้อม (Environment Variable) จาก System หรือ User Login Script ของเน็ตแวร์แทน หรืออีกวิธีหนึ่งคือ ให้ผู้บริหารระบบ (System Administator) สร้างเส้นทางการค้นหา (Search Path) ขึ้น

3. การใช้เครื่องพิมพ์ร่วมกัน (Printer Sharing)

การพิมพ์ในระบบเครือข่ายเน็ตแวร์ สามารถที่จะใช้เครื่องพิมพ์ร่วมกันได้ โดยที่โปรแกรมที่จะส่งพิมพ์ไม่จำเป็นต้องจัดการในเรื่องการพิมพ์เอง เพียงแต่ทำการส่งข้อมูลไปที่พอร์ต LPT ของเครื่องคอมพิวเตอร์เท่านั้น ส่วนการจะเปลี่ยนทิศทาง (Redirection) ไปยังเครื่องพิมพ์ของระบบเครือข่ายนั้นเน็ตแวร์ มีคำสั่ง Capture ให้ใช้โดยต้องระบุชื่อเครื่องบริการแฟ้ม (File Server) และชื่อแถวคอย (Queue) ของเครื่องบริการแฟ้มนั้น

4. การติดต่อสื่อสารในเครือข่าย

โปรแกรมจะสามารถติดต่อกับเครื่องบริการแฟ้มได้โดยผ่าน (Shell) ซึ่งถูกฝังอยู่ในหน่วยความจำ (TSR) โดย Shell จะทำหน้าที่คอยดักคำสั่งการทำงานของ DOS และตรวจสอบว่าเป็นคำสั่งที่เกี่ยวกับระบบเครือข่ายหรือไม่ ถ้าเป็นคำสั่งที่เกี่ยวกับระบบเครือข่ายก็จะสร้างรูปแบบเป็นคำสั่งร้องขอ (Request) แล้วส่งไปยังเครื่องบริการแฟ้ม เมื่อเครื่องบริการแฟ้มได้รับคำร้องขอก็จะตอบกลับ (Reply) มายังสถานีงานที่ร้องขอ (Charles G. Rose,1990)



ภาพที่ 4.1 แสดงการติดต่อระหว่างสถานีงานและเครื่องบริการแฟ้ม

5. การสลับไบต์สูงต่ำ

ดังที่ได้กล่าวมาแล้วว่า การเก็บค่าตัวเลขของบางฟิลด์ในโครงสร้างข้อมูลของเน็ตแวร์ มีบางส่วนที่เก็บข้อมูลแบบ High-Low Format เนื่องจากระบบปฏิบัติการเน็ตแวร์ในตอนต้นถูกพัฒนาขึ้นโดยใช้หน่วยประมวลผลของ Motorola (Charles G. Rose,1990) ในขณะที่หน่วยประมวลผลของ Intel ที่ใช้กันแพร่หลายในไมโครคอมพิวเตอร์ปัจจุบัน มีการเก็บข้อมูลแบบ Low - High Format คือเก็บไบต์ที่มีนัยสำคัญน้อย (Least Significant Byte) ก่อนไบต์ที่มีนัยสำคัญมาก (Most Significant Byte) เช่น 1234h จะถูกเก็บเป็น 34 12 ดังนั้นการเขียนโปรแกรมจะต้องสลับลำดับของตัวเลขให้ถูกต้อง โดยเน็ตแวร์ได้เตรียมฟังก์ชันไว้ให้ คือ IntSwap และ LongSwap

6. ภาษาคอมพิวเตอร์

การพัฒนาโปรแกรมบนเน็ตแวร์ สามารถใช้ได้ทุกภาษา ที่สามารถเรียกรูทีน (Routine) ภายนอกที่เป็น C หรือ Assembly ได้ แต่ภาษาที่เหมาะสมที่สุดคือ ภาษา C เนื่องจาก API ของเน็ตแวร์ ถูกเขียนด้วยภาษา C

7. Request และ Reply Packet

การทำงานของ Shell ของเน็ตแวร์ใช้วิธีดักอินเทอร์รัพท์ เช่น 08h, 2Fh, 7Ah ดังนั้น การเรียกฟังก์ชันของเน็ตแวร์ ส่วนใหญ่จะเรียกใช้โดยผ่านอินเทอร์รัพท์ เหล่านี้ แต่เนื่องจากฟังก์ชันของ เน็ตแวร์ ส่วนใหญ่ มีข้อมูลที่ต้องส่งไป และส่งกลับ จึงไม่สามารถใช้รีจีสเตอร์ (Register) ในการรับ - ส่ง ค่าได้หมด เน็ตแวร์ จึงได้ใช้วิธีที่เรียกว่า Request และ Reply Packet โดยจะใช้คู่ของรีจีสเตอร์ DS:SI ชี้ไปยังจุดเริ่มต้นของข้อมูลที่จะส่งไปให้เครื่องบริการแฟ้ม (Request Packet) และใช้ ES:DI รับข้อมูลที่เครื่องบริการแฟ้มส่งกลับมา (Reply packet)

8. การเรียกใช้ API ของ เน็ตแวร์

การเรียกใช้ API ของเน็ตแวร์ สามารถทำได้ 2 วิธีคือ การใช้อินเทอร์รัพท์เช่น 21h,7Ah และ โดยเรียกฟังก์ชันของภาษา C ที่เน็ตแวร์เตรียมไว้ ซึ่งวิธีการที่ 2 จะง่ายและสะดวกกว่าวิธีแรก

8.1 การใช้อินเทอร์รัพท์ เป็นวิธีที่ค่อนข้างยุ่งยากเนื่องจากต้องมีการอ้างถึงรีจิสเตอร์ต่างๆ และต้องเรียกใช้อินเทอร์รัพท์โดยตรง ซึ่งไม่เหมาะกับการเขียนด้วยภาษาระดับสูง การใช้งาน คือ เมื่อใช้ฟังก์ชัน 7Ah ของ อินเทร์รัพท์ 2Fh จะทำให้ทราบว่า IPX.COM ถูกฝังตัวอยู่ในหน่วยความจำหรือยัง ถ้าฝังตัวแล้วจะได้ค่าของ Offset และ Segment ของหน่วยความจำที่ฟังก์ชันของ IPX อยู่หลังจากนั้นก็จะใช้รีจิสเตอร์ BX เป็นตัวเก็บค่าฟังก์ชันที่จะใช้ แล้วกระโดด (Jump) ไปที่แอดเดรสของฟังก์ชัน IPX เพื่อเรียกใช้ฟังก์ชันของ IPX/SPX ได้ เช่น จากตัวอย่างจะเป็นการตรวจสอบการฝังตัวในหน่วยความจำของ IPX และใช้ฟังก์ชันหมายเลข 9 เพื่อหาค่าของแอดเดรสของสถานีงานนั้น

ตัวอย่าง โปรแกรมหาค่าแอดเดรสของสถานีงานโดยใช้ภาษา Assembly

```
_TEXT   SEGMENT BYTE PUBLIC 'CODE'
public    _ipx_func
          assume   CS:_TEXT,DS:DGROUP
_ipx_func proc     near
          mov      ax,7a00h
          int      2fh                    ;เช็คว่า IPX.COM ถูกฝังไว้ในหน่วยความจำหรือยัง
```

```
            inc     al
            jz      ipxload
            lea     dx,ds:err           ;
            mov     ah,9h               ;IPX not loaded
            int     21h                 ;
            jmp     end_prg
ipxload:
            lea     dx,ds:loaded
            mov     ah,9h               ;IPX loaded
            int     21h                 ;
;-------------------------------
            mov     IPXENTRY,di
            mov     IPXENTRY+2,es
            mov     bx,9h               ;Get Internetwork address
            push    ds
            pop     es
            lea     si,ds:reply_buf
            push    si
            call    dword ptr IPXENTRY
            pop     si
;-------------------------------
end_prg:
;           mov     ah,4ch
;           int     21h
            ret
_ipx_func   endp
_TEXT       ENDS
_DATA   SEGMENT PARA PUBLIC 'DATA'
            org 100h
IPXENTRY    dw ?
            dw ?
err         db 'NO IPX load.',10,13,'$'
loaded      db 'IPX loaded already.',10,13,'$'
reply_buf   db 10 dup (?)
_DATA       ENDS
            END
```

8.2 การเรียกใช้ฟังก์ชันจากไลบรารี (Library) เน็ตแวร์ได้เตรียมฟังก์ชันการทำงานต่าง ๆ สำหรับระบบเครือข่ายไว้ให้ เพื่อความสะดวกในการเรียกใช้ โดยรวมไว้เป็นไลบรารี ฟังก์ชันต่าง ๆ เหล่านี้ถูกเขียนขึ้นด้วยภาษา C ดังนั้นภาษา C จึงเป็นภาษาที่เหมาะสมที่สุดในการพัฒนาโปรแกรม ไลบรารีที่เน็ตแวร์เตรียมไว้มีทั้งโมเดล เล็ก (Small) , กลาง (Medium), และใหญ่ (Large) ให้ผู้พัฒนา สามารถเลือกใช้ได้ตามความเหมาะสม

ตัวอย่าง โปรแกรมหาค่าแอดเดรสของสถานีงานโดยใช้ภาษา C

```
#include <stdlio.h>
#include <nxt.h>

void main()
{
    int statusCode;
    IPXAddress networkAddress;

    statusCode = IPXInitialize();
    if (statusCode == 0)
       IPXGetInternetworkAddress(&networkAddress);
    else
       printf("IPX not loaded.\n");
}
```

เน็ตแวร์ได้แบ่งฟังก์ชันการทำงานต่าง ๆ ออกเป็นหมวดหมู่ ตามหน้าที่ของแต่ละฟังก์ชัน เพื่อให้สะดวกในการเรียกใช้งาน ดังนี้ (Novell Inc.,1989)

1) Accounting Service เป็นฟังก์ชันเกี่ยวกับบัญชีผู้ใช้

2) AFP Service เป็นฟังก์ชันเกี่ยวกับ AppleTalk Filing Protocol

3) Bindery Service เป็นฟังก์ชันเกี่ยวกับฐานข้อมูล Bindery ซึ่งจะเก็บข้อมูลเกี่ยวกับผู้ใช้ รหัสผ่าน แถวคอย (Queue) และข้อมูลอื่นๆอีก

4) Communication Service เป็นฟังก์ชันเกี่ยวกับการรับส่งข้อมูลด้วย IPX/SPX

5) Connection Service เป็นฟังก์ชันเกี่ยวกับการติดต่อกับเครื่องบริการแฟ้ม เช่น Login, Logout เป็นต้น

6) Diagnostic Service เป็นฟังก์ชันเกี่ยวกับการตรวจสอบข้อมูลต่างๆ เช่นสถิติของการรับส่งข้อมูล IPX/SPX เวอร์ชันของ IPX เป็นต้น

7) Directory Service เป็นฟังก์ชันเกี่ยวกับการสร้างและลบไดเรกทอรี การให้สิทธิในการเข้าถึงไดเรกทอรี เป็นต้น

8) File Server Service เป็นฟังก์ชันเกี่ยวกับการทำงานของเครื่องบริการแฟ้ม เช่น การ

ปิดเครื่องบริการแฟ้ม การเก็บสถิติต่างๆของเครื่องบริการแฟ้ม

9) File Service เป็นฟังก์ชันเกี่ยวกับการให้บริการแฟ้มเช่น เรื่องแอตตริบิวต์ (Attribute) ของแฟ้ม

10) Message Service เป็นฟังก์ชันเกี่ยวกับการรับส่งข้อความโดยผ่านเครื่องบริการแฟ้ม

11) Miscellaneous Service เป็นฟังก์ชันเกี่ยวกับ

12) Name Space Service เป็นฟังก์ชันเกี่ยวกับ

13) Print Server Service เป็นฟังก์ชันเกี่ยวกับการติดตั้งเครื่องพิมพ์ของระบบเครือข่าย

14) Print Service เป็นฟังก์ชันเกี่ยวกับการพิมพ์ และการดัก (Capture) ข้อมูลจากพอร์ตขนานของสถานีงาน

15)Queue Service เป็นฟังก์ชันเกี่ยวกับแถวคอยสำหรับการพิมพ์ เช่น การสร้าง การลบ การส่งงานไปเข้าแถวคอย เป็นต้น

16) Service Advertising Protocol Service เป็นฟังก์ชันเกี่ยวกับการบรอดแคสท์ข้อมูลต่างๆเช่น ชื่อเครื่องบริการแฟ้ม

17) Synchronization Service เป็นฟังก์ชันเกี่ยวกับล็อคแฟ้ม ล็อคเรคคอร์ด

18)Transaction Tracking System Service เป็นฟังก์ชันเกี่ยวกับการป้องกันความผิดพลาดในการเขียนข้อมูล

19) Value Added Processes Service เป็นฟังก์ชันเกี่ยวกับการใช้ VAP

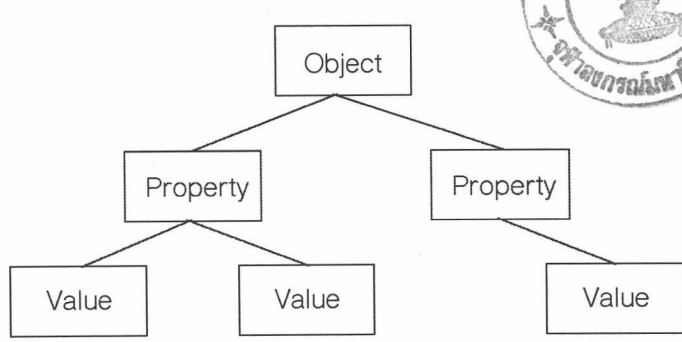
20) Workstation Environment Service เป็นฟังก์ชันเกี่ยวกับสถานีงาน เช่นการหาหมายเลขการเชื่อมต่อ Connection ID

8.3 การให้บริการเกี่ยวกับฐานข้อมูล (Bindery Service)

เน็ตแวร์จะมีฐานข้อมูลเล็กๆ สำหรับใช้เก็บข้อมูลเกี่ยวกับผู้ใช้ (User), กลุ่ม (Group), แถวคอย (Queue) และข้อมูลจำเป็นอื่นๆ และยังทำหน้าที่จัดการวัตถุที่เปลี่ยนแปลงได้ (Dynamic Object) เช่น เครื่องบริการแฟ้มที่อยู่บนระบบเครือข่าย

ฐานข้อมูลของเน็ตแวร์ประกอบด้วย 3 ส่วน (Ralph Davis, 1991) คือ

1. วัตถุ (Object)
2. พรอพเพอร์ตี (Properties)
3. ค่าของข้อมูล (Value)



ภาพที่ 4.2 ภาพแสดงความสัมพันธ์ระหว่าง วัตถุ พรอพเพอร์ตีและค่าของวัตถุ

8.3.1 วัตถุ (Object) การระบุถึงวัตถุใด ๆ สามารถระบุได้ด้วยชื่อของวัตถุ และชนิดของวัตถุซึ่งอาจจะเป็นผู้ใช้, กลุ่ม, เครื่องบริการแฟ้ม, แถวคอยสำหรับการพิมพ์ (Print Queue) หรืออาจจะเป็นอะไรก็ได้ ที่ผู้พัฒนาจะกำหนดขึ้นมา ดังนั้นชื่อของวัตถุอาจจะซ้ำกันก็ได้ แต่ชนิดจะต้องไม่ซ้ำก้นถ้าใช้ชื่อเดียวกัน Object ประกอบด้วยฟิลด์ต่าง ๆ ดังนี้

- ชื่อของวัตถุ (Object Name) มีความยาวสูงสุด 48 ตัวอักษร โดยที่ตัวสุดท้ายจะต้องปิดด้วยศูนย์ ตัวอักษรที่ใช้ได้ คือ ตัวอักษรจาก 21h ถึง 70h ตามตาราง Ascii โดยไม่รวมถึงตัวอักษรต่อไปนี้ ( ), (/), (\), (:), (;), (,), (*), (?) และเนื่องจากชื่อจะถูกเก็บเป็นตัวพิมพ์ใหญ่ ดังนั้นในการค้นหาจะต้องใช้ตัวอักษรพิมพ์ใหญ่ด้วย

- หมายเลขของวัตถุ (Object Identification Number) เป็นตัวเลขขนาด 4 ไบต์ และต้องไม่ซ้ำกันภายใน 1 เครื่องบริการแฟ้ม

- ชนิดของวัตถุ (Object Type) มีขนาด 2 ไบต์ 0 หมายถึง ไม่สามารถระบุชนิดได้ -1 ใช้สำหรับการค้นหาวัตถุที่ไม่ทราบชนิด 1-8000h ถูกใช้โดยเน็ตแวร์ ชนิดของวัตถุที่ใช้บ่อย เช่น

ชนิดของวัตถุ	ความหมาย
1	User
2	User Group
3	Print Queue
4	File Server
5	Job Server
6	Gateway
7	Print Server
8	Archive Queue
9	Archive Server
A	Job Queue
B	Administration
24	Remote Bridge Server
47	Advertising Printer Server

- สถานะของวัตถุ (Object Flag) ขนาด 1 ไบต์ ใช้ระบุว่า Object นั้นเป็นแบบคงที่ (Static) มีค่า 0 หรือ เปลี่ยนแปลงได้ (Dynamic) มีค่า 1 วัตถุคงที่จะถูกสร้าง และลบไปโดยผู้ดูแลระบบ เช่น ผู้ใช้ (User) แต่วัตถุเปลี่ยนแปลงได้จะถูกสร้างขึ้น และ ลบ โดยอัตโนมัติ เช่น เครื่องบริการแฟ้มอื่นในเครือข่าย

- การรักษาความปลอดภัยของวัตถุ (Object Security Flag) มีขนาด 1 ไบต์ 4 บิตแรกใช้บอกว่าใครสามารถอ่านข้อมูลวัตถุนั้นได้ ส่วน 4 บิตหลัง ใช้บอกผู้ที่สามารถเขียน หรือ แก้ไขข้อมูลของวัตถุได้ โดยแต่ละ 4 บิต มีค่าดังนี้

0 หมายถึง ทุกวัตถุ

1 หมายถึง วัตถุที่ Login เข้าสู่เครื่องบริการแฟ้ม

2 หมายถึง วัตถุที่ Login เข้าสู่เครื่องบริการแฟ้ม โดยมีชื่อชนิดและรหัสผ่านตรงกัน

3 หมายถึง เฉพาะผู้ดูแลระบบ (Supervisor) หรือผู้ที่มีสิทธิเท่าเทียมกับผู้ดูแลระบบ

4 หมายถึง เฉพาะเน็ตแวร์เท่านั้น

8.3.2 พรอพเพอร์ตี (Property) แต่ละวัตถุอาจมีพรอพเพอร์ตีที่สัมพันธ์กันหนึ่ง หรือมากกว่า เช่น วัตถุชื่อ KITTI เป็น User (ชนิดเป็น 1) อาจสัมพันธ์กับพรอพเพอร์ตี GROUP_I'M_IN , ACCOUNT_BALANCE และ PASSWORD พรอพเพอร์ตีจะไม่มีการเก็บข้อมูลไว้ในตัวเอง เช่น พรอพเพอร์ตี Password จะไม่เก็บข้อมูลของ Password เอาไว้ แต่มีสิ่งที่สัมพันธ์กับพรอพเพอร์ตีและใช้เก็บข้อมูลเหล่านี้ คือ ค่าของข้อมูล (Value) ซึ่งจะกล่าวถึงต่อไปพรอพเพอร์ตี ประกอบด้วยฟิลด์ต่างๆ ดังนี้

- ชื่อพรอพเพอร์ตี (Property Name) ความยาว 15 ตัวอักษร และประกอบด้วยตัวอักษรพิมพ์ใหญ่เหมือนกับชื่อของวัตถุ

- ความปลอดภัยของพรอพเพอร์ตี (Property Security Flag) ขนาด 1 ไบต์ เหมือนความปลอดภัยของวัตถุ

- สถานะของพรอพเพอร์ตี (Property Flag) ขนาด 1 ไบต์ บิตที่ 0 เป็น 0 หมายถึงคงที่ (Static) เป็น 1 หมายถึงเปลี่ยนแปลงได้ (Dynamic) บิตที่ 1 เป็น 0 หมายถึง พรอพเพอร์ตีเดี่ยว (Item property) 1 หมายถึง พรอพเพอร์ตีชุด (Set property) โดยพรอพเพอร์ตีเดี่ยวสัมพันธ์กับค่าของข้อมูลขนาด 128 ไบต์ เช่น พรอพเพอร์ตีเดี่ยว ACCOUNT_BALANCE ในขณะที่พรอพเพอร์ตีชุด สัมพันธ์กับค่าของข้อมูล ขนาด 128 ไบต์ เรียกว่า เซกเมนต์ (Segment) ซึ่ง 1 พรอพเพอร์ตีชุดอาจสัมพันธ์กับเซกเมนต์ มากกว่า 1 โดยที่แต่ละเซกเมนต์ประกอบด้วยหมายเลขวัตถุขนาด 4 ไบต์ ตั้งแต่ 1 ถึง 32 วัตถุ เช่น พรอพเพอร์ตี GROUP_I'M_IN เน็ตแวร์กำหนดพรอพเพอร์ตี ไว้ดังนี้ (ในวงเล็บหมายถึง Property นั้นเป็นแบบคงที่ (static) หรือ เปลี่ยนแปลงได้ (Dynamic) และเป็นแบบเดี่ยว (Item) หรือ ชุด (Set)) พรอพเพอร์ตีมีดังต่อไปนี้

ACCOUNT_BALANCE (Static,Item) เก็บข้อมูลเกี่ยวกับเครดิตที่ใช้ไปและเหลืออยู่

ACCOUNT_HOLDS (Dynamic,Item) เป็นการบอกว่าเครื่องบริการแฟ้มใดกำลัง Hold ผู้ใช้อยู่ การ Hold จะเกิดขึ้นเมื่อผู้ใช้มีการร้องขอข้อมูลไปยังเครื่องบริการแฟ้ม

ACCOUNT_SERVER (Static,Set) เก็บรายชื่อเครื่องบริการแฟ้มที่อนุญาติให้คิดเครดิตกับผู้ใช้

ACCOUNT_LOGOUT (Static,Item) บอกว่า Account ใดถูกล็อค

BLOCKS_READ (Static,Item) บอกว่าเครื่องบริการแฟ้มจะคิดเครดิตเท่าไร จากจำนวนบล็อคที่มีการอ่าน

BLOCKS_WRITTEN (Static,Item) บอกว่าเครื่องบริการแฟ้มจะคิดเครดิตเท่าไรจากจำนวนบล็อคที่มีการเขียน

CONNECT_TIME (Static,Item) บอกว่าเครื่องบริการแฟ้มจะคิดเครดิตเท่าไรจากจำนวนเวลาที่ผู้ใช้ Login

DISK_STORAGE (Static,Item) บอกว่าเครื่องบริการแฟ้มจะคิดเครดิตเท่าไรจากจำนวนเนื้อที่ของจานแม่เหล็กที่ผู้ใช้ใช้เก็บข้อมูล

GROUP_MEMBERS (Static,Set) บอกชื่อของสมาชิกที่อยู่ในกลุ่ม

GROUP_I'M_IN (Static,Set) บอกชื่อของกลุ่มที่ผู้ใช้เป็นสมาชิกอยู่

IDENTIFICATION (Static,Item) เก็บชื่อของ Object

LOGIN_CONTROL (Static,Item) เก็บข้อมูลเกี่ยวกับการควบคุมการ Login เช่น ความยาวของ Password วันหมดอายุของ Password

NET_ADDRESS (Dynamic,Item) เก็บข้อมูลแอดเดรสของเครื่องบริการแฟ้มที่ได้จากการกระจาย (Broadcast) ของ SAP จากเครื่องบริการแฟ้ม

NODE_CONTROL (Static,Item) เก็บรายชื่อผู้ใช้ที่ Login อยู่

OLD_PASSWORDS (Static,Item) เก็บ Password เก่าที่ผู้ใช้เคยใช้งาน โดยเก็บได้สูงสุด 8 Password

OPERATORS (Static,Set) เก็บ Object ที่อนุญาตให้ใช้งาน Console และใช้ฟังก์ชันของ FOPERATOR

PASSWORD (Static,Item) เก็บ Password ที่มีการ Encrypt

Q_DIRECTORY (Static,Item) เก็บแถวคอยของการพิมพ์ (Print Queue) และไดเรกทอรีของแถวคอย

Q_OPERATORS (Static,Set) เก็บแถวคอยของการพิมพ์ (Print Queue) และผู้ใช้ที่สามารถจัดการกับแถวคอย

Q_SERVERS (Static,Set) เก็บแถวคอยของการพิมพ์ (Print Queue) และเครื่องบริการแฟ้มที่สามารถจัดการกับงานในแถวคอย

Q_USERS (Static,Set) เก็บแถวคอยของการพิมพ์ (Print Queue) และผู้ใช้ที่สามารถส่งงานเข้าสู่แถวคอย

REQUESTS_MADE (Static,Item) บอกว่าเครื่องบริการแฟ้มจะคิดเครดิตเท่าไรจาก Request ของโปรแกรมเมอร์

SECURITY_EQUALS (Static,Set) บอกรายชื่อ Object ที่ผู้ใช้มีสิทธิเทียบเท่า

USER_DEFAULTS (Static,Item) เก็บค่าเริ่มต้นที่จะให้กับผู้ใช้ใหม่

8.3.3 ค่าของข้อมูล (Value) เป็นส่วนที่เก็บข้อมูลจริงที่สัมพันธ์กับพรอพเพอร์ตี มีขนาด 128 ไบต์ เรียกว่า 1 เซกเมนต์ ซึ่งอาจมีมากกว่า 1 Segment ขึ้นอยู่กับชนิดของ Property ว่าเป็นเดี่ยว หรือ ชุด

API ที่เกี่ยวข้องกับ Bindery Service (Novell Inc.,1989)

- Add Bindery Object to Set สำหรับเพิ่ม Object เข้าไปใน Property

รูปแบบการเรียกใช้ภาษา C

```
int AddBinderyObjectToSet(char *objectName,WORD objectType,
    char *propertyName,char *memberName,WORD memberType);
```

- Change Bindery Object Password สำหรับเปลี่ยน Password ของ Object

รูปแบบการเรียกใช้ภาษา C

```
int ChangeBinderyObjectPassword(char *objectName,WORD objectType,
    char *oldPassword,char *newPassword);
```

- Change Bindery Object Security สำหรับเปลี่ยนการรักษาความปลอดภัย (Security) ของ Object

รูปแบบการเรียกใช้ภาษา C

```
int ChangeBinderyObjectSecurity(char *objectName,WORD objectType,
    BYTE newObjectSecurity);
```

- Change Property Security สำหรับเปลี่ยนการรักษาความปลอดภัย (Security) ของ Property

รูปแบบการเรียกใช้ภาษา C

```
int ChangePropertySecurity(char *objectName,WORD objectType,
    char *propertyName,BYTE newPropertySecurity);
```

- Close Bindery สำหรับปิดแฟ้มข้อมูล Bindery

รูปแบบการเรียกใช้ภาษา C

```
int CloseBindery(void);
```

- Create Bindery Object สำหรับสร้าง Object ของ Bindery

รูปแบบการเรียกใช้ภาษา C

```
int CreateBinderyObject (char *objectName,WORD objectType,
    BYTE objectFlag,BYTE objectSecurity);
```

- Create Property สำหรับสร้าง Property ใหม่

รูปแบบการเรียกใช้ภาษา C

```
int CreateProperty(char *objectName,WORD objectType,char *propertyName,
    BYTE propertyFlags,BYTE propertySecurity);
```

- Delete Bindery Object สำหรับลบ Object ของ Bindery

รูปแบบการเรียกใช้ภาษา C

```
int DeleteBinderyObject (char *objectName,WORD objectType);
```

- Delete Bindery Object From Set สำหรับลบ Object ของ Property ชนิด Set

รูปแบบการเรียกใช้ภาษา C

```
int DeleteBinderyObjectFromSet(char *objectName,WORD objectType,
    char *propertyName,char *memberName,WORD memberType);
```

-Delete Property สำหรับลบ Property จาก Object

รูปแบบการเรียกใช้ภาษา C

```
int DeleteProperty(char *objectName,WORD objectType,char *propertyName);
```

- Get Bindery Access Level สำหรับหาระดับการเข้าถึงของ Object ที่กำหนด

รูปแบบการเรียกใช้ภาษา C

```
int GetBinderyAccessLevel(BYTE *securityAccessLevel,long *objectID);
```

- Get Bindery Object ID สำหรับหา Object ID ของ Object ตามชื่อที่ระบุ

รูปแบบการเรียกใช้ภาษา C

int GetBinderyObjectID(char *objectName,Word objectType,long *objectID);

- Get Bindery Object Name สำหรับหาชื่อของ Object ID ที่ระบุ

รูปแบบการเรียกใช้ภาษา C

int GetBinderyObjectName(long objectID,char *objectName,WORD *objectType);

- Is Bindery Object In Set สำหรับหาว่า Object ที่ระบุเป็นสมาชิกของ Property หรือไม่

รูปแบบการเรียกใช้ภาษา C

int IsBinderyObjectInSet(char *objectName,WORD objectType,
char *propertyName,char *memberName,WORD memberType);

- Open Bindery สำหรับเปิดแฟ้มข้อมูล Bindery จะต้องเรียกก่อนที่จะใช้ฟังก์ชันอื่น ๆ ของ Bindery

รูปแบบการเรียกใช้จากภาษา C

int OpenBindery(void);

- Read Property Value สำหรับอ่านค่า Value ของ Property

รูปแบบการเรียกใช้จากภาษา C

int ReadPropertyValue(char *objectName,WORD objectType,
char *propertyName,int segmentNumber,BYTE *propertyValue,BYTE
*moreSegments,BYTE *propertyFlags);

- Rename Bindery Object สำหรับเปลี่ยนชื่อ Object

รูปแบบการเรียกใช้จากภาษา C

int RenameBinderyObject(char *objectName,char *newObjectName,
WORD objectType);

- Scan Bindery Object สำหรับค้นหา Object ใน Bindery

รูปแบบการเรียกใช้ภาษา C

int ScanBinderyObject (char *searchObjectName,WORD searchObjectType,
long *objectID,char *objectName,WORD *objectType,
char *objectHasProperties,char *objectFlag,
char *objectSecurity);

- Scan Bindery Object Trustee Paths สำหรับค้นหาเส้นทางของไดเรกทอรีของ Object

รูปแบบการเรียกใช้ภาษา C

int ScanBinderyObjectTrusteePaths(long objectID,BYTE volumeNumber,
int *sequenceNumber,WORD *trusteeAccessMask,char *trusteePathName);

- Scan Property สำหรับค้นหา Property ตามชื่อ Object ที่ระบุ

รูปแบบการเรียกใช้ภาษา C

int ScanProperty(char *objectName,WORD objectType,char
*searchPropertyName,long *sequenceNumber,char *propertyName,
char *propertyFlags,char *propertySecurity,char *propertyHasValue,

I16884115

```
        char *moreProperties);
```

- Verify Bindery Object Password สำหรับตรวจสอบ Password ของ Object

รูปแบบการเรียกใช้ภาษา C

```
int VerifyBinderyObject(char *objectName,WORD objectType,char *password);
```

-Write Property Value สำหรับเขียนค่าลงใน Property

รูปแบบการเรียกใช้ภาษา C

```
int WritePropertyValue(char *objectName,WORD objectType,char
    *propertyName,int segmentNumber,BYTE *propertyValue,BYTE
     moreSegments);
```

8.4 การให้บริการการเชื่อมต่อ (Connection Service)

ก่อนที่ผู้ใช้จะสามารถเข้าไปใช้ทรัพยากรของเครื่องบริการแฟ้มได้ จะต้องมีการสร้างเส้นทางเชื่อมต่อระหว่างสถานีงาน และ เครื่องบริการแฟ้ม ทั้งในแง่การเชื่อมต่อกายภาพ (Physical Connection) และการเชื่อมต่อเสมือน (Logical Connection)

การเชื่อมต่อกายภาพ คือ การนำการ์ดระบบเครือข่ายมาเสียบเข้าเครื่องคอมพิวเตอร์ที่เป็นสถานีงาน และ เครื่องบริการแฟ้ม และเชื่อมต่อคอมพิวเตอร์ทั้งสองเข้าด้วยกัน

การเชื่อมต่อเสมือน คือ การใช้คำสั่ง Login หรือ Attach ของเน็ตแวร์นั่นเอง การสร้างการเชื่อมต่อเสมือนนั้น สถานีงานจะใช้ตาราง 2 ตาราง คือ

- ตารางชื่อเครื่องบริการแฟ้ม (Server Name Table)
- ตารางการเชื่อมต่อ (Connection ID Table)

ในขณะที่เครื่องบริการแฟ้มจะใช้ตาราง 2 ตาราง คือ

- ตารางการเชื่อมต่อของเครื่องบริการแฟ้ม (File Server Connection Table)
- ตารางรหัสผ่าน (Password Table)

โดยที่ตารางชื่อเครื่องบริการแฟ้มแต่ละเรคคอร์ด คือชื่อของเครื่องบริการแฟ้มมีขนาด 48 ไบต์ มีได้ไม่เกิน 8 เรคคอร์ด ส่วนตารางการเชื่อมต่อ 1 เรคคอร์ด มีขนาด 32 ไบต์ ดังตารางต่อไปนี้

ตำแหน่ง	ค่าที่เก็บ	ชนิด
0	In Use Flag	BYTE
1	Order Number	BYTE
2	Network Number	BYTE[4]
6	Node Address	BYTE[6]
12	Socket Number	BYTE[2]
14	Recieve Timeout	BYTE[2]
16	Routing Node	BYTE[6]
22	Packet Sequence	BYTE
23	Connection Number	BYTE
24	Connection Status	BYTE
25	Maximum Timeout	BYTE[2]
27	Padding/Reserved	BYTE[5]

ตารางที่ 4.1 ตารางการเชื่อมต่อ (Connection ID Table)

Inuse Flag แสดงสถานะของการทำงานของเชลล์บนสถานีงาน

Order Number คือ ลำดับของเครื่องบริการแฟ้มที่เชื่อมต่อโดยเรียงตามแอดเดรสของเครื่องบริการแฟ้ม และระบบเครือข่าย จากน้อยไปมาก

Network Number คือ หมายเลขระบบเครือข่ายที่เครื่องบริการแฟ้มอยู่

Node Address คือ แอดเดรสของเครื่องบริการแฟ้ม

Socket Number คือ ซอคเก็ตของเครื่องบริการแฟ้มที่จะรับข้อมูล

Recieve Timeout คือ จำนวนของสัญญาณนาฬิกา ที่สถานีงานจะรอการตอบรับจากเครื่องบริการแฟ้ม และจะถูกปรับค่าตามความเหมาะสมโดยเชลล์ แต่จะไม่มากกว่า Maximum Timeout

Routing Node คือ แอดเดรสของบริดจ์ (Bridge) หรือ เราท์เตอร์ (Router) ที่เป็นตัวส่ง ผ่านข้อมูลมาให้

Packet Sequence Number คือ ลำดับของข้อมูลที่ถูกส่งไปยังเครื่องบริการแฟ้ม

Connection Number คือ หมายเลขการเชื่อมต่อที่เครื่องบริการแฟ้มส่งมาให้

Connection Status 0 หมายถึง ไม่มีการเชื่อมต่อ FF หมายถึง มีการเชื่อมต่อ

Maxium Timeout คือ จำนวนสูงสุดของสัญญาณนาฬิกาที่ Receive Timeout จะยอมรับได้

API ที่เกี่ยวข้องกับ Connection Service (Novell Inc.,1989)

- Attach To File Server สำหรับสร้างการเชื่อมต่อกับเครื่องบริการแฟ้ม

รูปแบบการเรียกใช้ภาษา C

int AttachToFileServer (char *serverName, WORD * connectionID);

- Attach To FileServer With Address สำหรับสร้างการเชื่อมต่อกับเครื่องบริการแฟ้มโดยระบุแอดเดรส

รูปแบบการเรียกใช้ภาษา C

int AttachToFileServerWithAddress (char *serverName, WORD *connectID,
BYTE *netAddress);

- Detach From File Server สำหรับสิ้นสุดการเชื่อมต่อกับเครื่องบริการแฟ้ม

รูปแบบการเรียกใช้ภาษา C

int DetachFromFileServer (WORD connectionID);

- Enter Login Area สำหรับเปลี่ยนไดเรกทอรีไปยังไดเรกทอรีที่ Login utility อยู่

รูปแบบการเรียกใช้ภาษา C

int EnterLoginArea (char *loginSubdirectioryName,
int numberOfLocalDrives);

- Get Connection Information สำหรับหาข้อมูลของ Object ตามหมายเลข Connection ที่กำหนด

รูปแบบการเรียกใช้ภาษา C

int GetConnectionInformation (Word connectionNumber,
char *objectName, Word *objectType, long *objectID,
BYTE *loginTime);

- Get Connection Number สำหรับหาหมายเลข Connection ของสถานีงานนั้น

รูปแบบการเรียกใช้ภาษา C

WORD GetConnectionNumber (void);

- Get Internet Address สำหรับหาแอดเดรสของสถานีงานตามหมายเลข Connection ที่กำหนด

รูปแบบการเรียกใช้ภาษา C

int GetInternetAddress (WORD connection Number, char *networkNumber,
char *physicalNodeAddress, WORD *socketNumber);

- Get Object Connection Number สำหรับหาหมายเลข Connection ตามชื่อ Object ที่กำหนด

รูปแบบการเรียกใช้ภาษา C

int GetObjectConnectionNumber (char *objectName, WORD objectType,
WORD *numberOfConnections, WORD *connectionList,
WORD maxConnections);

- Get Station Address สำหรับหาแอดเดรสของสถานีงานนั้น

รูปแบบการเรียกใช้ภาษา C

void GetStationAddress (BYTE *physicalNodeAddress);

- Login To File Server สำหรับ Login เข้าสู่เครื่องบริการแฟ้ม

รูปแบบการเรียกใช้ภาษา C

int LoginToFileServer (char *objectName, WORD objectType,
char *objectPassword);

- Logout สำหรับ Logout ออกจากเครื่องบริการแฟ้มและสิ้นสุดการเชื่อมต่อกับเครื่องบริการแฟ้มทุกเครื่องที่มี Connection อยู่

รูปแบบการเรียกใช้ภาษา C

void Logout (void);

- Logout From File Server สำหรับ Logout ออกจากเครื่องบริการแฟ้มตามหมายเลข Connection ที่กำหนด

รูปแบบการเรียกใช้ภาษา C

int LogoutFromFileserver (WORD connectionID);

8.5 การให้บริการรับ-ส่งข้อความ (Message Service)

เป็นการให้บริการเกี่ยวกับการรับ - ส่ง ข้อความจากสถานีงานหนึ่ง ไปยังอีกสถานีงานหนึ่งหรือไปยังเครื่องบริการแฟ้ม มี 2 ชนิด คือ

- ข้อความแบบกระจาย (Broadcast Message) เมื่อต้องการส่งเพียงข้อความเดียวหรือไม่มาก

- ไปป์ (Pipe) เมื่อต้องการส่งข้อความมากขึ้น

การให้บริการนี้เป็นการให้บริการในระดับสูง สำหรับการส่งข้อความแบบเครื่องถึงเครื่อง (Station to Station) ซึ่งจะต้องมีเครื่องบริการแฟ้มเข้ามาเกี่ยวข้องในการรับส่ง วิธีนี้จึงไม่ใช่การส่งแบบ Peer to Peer ที่สมบูรณ์ จึงไม่เหมาะสำหรับการส่งข้อมูลมากๆ และต้องการความรวดเร็ว

8.5.1 การส่งข้อความแบบกระจาย ตัวอย่างของการส่งแบบนี้ คือการใช้คำสั่ง Send ของเน็ตแวร์ ซึ่งข้อความจะปรากฏที่บรรทัด 25 ของจอภาพ สามารถส่งข้อความได้สูงสุด 55 ไบต์ โหมด (Mode) ของการรับ - ส่งข้อความแบบนี้มี 4 โหมด

โหมด 0 จะรับข้อความที่มาจากสถานีงานอื่น และจากเครื่องบริการแฟ้ม และ แสดงผลหน้าจอ

โหมด 1 จะรับข้อความที่มาจากเครื่องบริการแฟ้มเท่านั้น และแสดงผลหน้าจอ

โหมด 2 จะรับข้อความที่มาจากเครื่องบริการแฟ้มเท่านั้น แต่ไม่แสดงผลหน้าจอ

โหมด 3 จะรับข้อความที่มาจากสถานีงาน และจากเครื่องบริการแฟ้ม แต่ไม่แสดงผล

8.5.2 ไปป์ (Pipe) จะมีความสามารถรับส่งข้อมูลได้ดีกว่าการส่งข้อความแบบกระจาย สามารถส่ง ข้อความได้ยาว 126 ไบต์ และมีแถวคอยถึง 6 แถว ไปป์มีข้อยุ่งยากมากกว่าการส่งข้อความแบบกระจาย คือ ต้องมีการเปิด (Open) เพื่อใช้งาน และ ปิด (Close) เมื่อไม่ใช้งาน

API ที่เกี่ยวข้องกับ Message Service (Novell Inc.,1989)

- Get Broadcast Mode สำหรับหาค่าโหมด

รูปแบบการเรียกใช้ภาษา C

BYTE GetBroadcastMode (void);

- Set Broadcast Mode สำหรับตั้งค่าโหมด

รูปแบบการเรียกใช้ภาษา C

void SetBroadcastMode (BYTE broadcastMode);

- Send Broadcast Message สำหรับส่งข้อความ

รูปแบบการเรียกใช้ภาษา C

Int SendBroadcastMessage (char *message, WORD *connectionList,
BYTE *resultList, WORD connectionCount);

- Get Broadcast Message สำหรับรับข้อความ

รูปแบบการเรียกใช้ภาษา C

Int GetBroadcastMessage (char *messageBuffer);

- Broadcast To Console สำหรับส่งข้อความไปยังจอ console ของเครื่องบริการแฟ้ม

รูปแบบการเรียกใช้ภาษา C

Int BroadcastToConsole (char *message);

- Send Personal Message สำหรับส่งข้อความไปยังสถานีงานตามหมายเลข Connection ที่กำหนด

รูปแบบการเรียกใช้ภาษา C

Int SendPersonalMessage (char *message, WORD *connectionList
BYTE *resultList, WORD connectionCount);

- Get Personal Message สำหรับรับข้อความที่ส่งมาด้วยคำสั่ง SendPersonnelMessage

รูปแบบการเรียกใช้ภาษา C

Int GetPersonalMessage (char *messageBuffer, WORD *sourceConnection);

- Log Network Message สำหรับเก็บข้อความที่ต้องการไว้ในไฟล์ชื่อ NET$LOG.MSG ที่ไดเรกทอรี SYS:SYSTEM

รูปแบบการเรียกใช้ภาษา C

Int LogNetworkMessage (char *message);

- Open Message Pipe สำหรับเริ่มต้นการรับส่งข้อความกับสถานีงานตามหมายเลข Connection ที่กำหนด

รูปแบบการเรียกใช้ภาษา C

Int OpenMessagePipe (WORD *connectionList, BYTE *resultList,
WORD connectionCount);

- Close Message Pipe สำหรับสิ้นสุดการรับส่งข้อความกับสถานีงานตามหมายเลข Connection ที่กำหนด

รูปแบบการเรียกใช้ภาษา C

Int CloseMessagePipe (WORD *connectionList, BYTE *resultList,
WORD connectionCount);

- Check Pipe Status สำหรับตรวจสอบสถานะของการติดต่อตามหมายเลข Connection ที่กำหนด

รูปแบบการเรียกใช้ภาษา C

```
Int Check Pipe Status (WORD *connectionList, BYTE *resultList,
  WORD connectionCount);
```

8.6 การให้บริการด้านติดต่อสื่อสาร (Communication Service)

การให้บริการด้านการติดต่อสื่อสารบนเน็ตแวร์ จะใช้โปรโตคอล 2 แบบ คือ

1. IPX (Internetwork Packet Exchange)
2. SPX (Sequence Packet Exchange)

ซึ่งเป็นโปรโตคอลที่มีต้นแบบมาจาก โปรโตคอล XNS (Xerox Network System) การติดต่อสื่อสารด้วยโปรโตคอล IPX และ SPX นี้จะเป็นการติดต่อสื่อสารแบบ Peer to Peer ที่แท้จริง คือ คอมพิวเตอร์ 2 เครื่อง สามารถติดต่อกันได้เอง โดยไม่ต้องอาศัยเครื่องบริการแฟ้ม เพียงแค่มีระบบเครือข่ายถึงกัน และ คอมพิวเตอร์ทั้ง 2 เครื่อง ได้บรรจุโปรแกรมเชลล์เข้าไว้ในหน่วยความจำ ดังนั้นการส่งข้อมูลด้วยวิธีนี้ จะรวดเร็ว และ มีประสิทธิภาพมากกว่าการใช้ Message Service จากข้อ 4.8.5

การให้บริการรับส่งข้อมูลด้วย IPX และ SPX นี้จะมีโครงสร้าง (Structure) ที่เรียกว่า ECB (Event Control Block) ใช้สำหรับควบคุม โดยที่จะมีฟิลด์หนึ่งของ ECB จะชี้ไปยังโครงสร้างของ IPX หรือ SPX นอกจากนี้ยังสามารถกำหนดได้ว่า เมื่อมีเหตุการณ์เกิดขึ้น เช่น มีแพคเก็ตส่งมาจะให้ทำฟังก์ชันอะไร ดังนั้นในการรับ-ส่ง ข้อความด้วย IPX / SPX จะต้องทำความเข้าใจโครงสร้างของ ECB ก่อน

ตำแหน่ง	ค่าที่เก็บ	ชนิด
0	Link Address	byte[4]
4	ESR Address	byte[4]
8	In Use Flag	byte
9	Completion Code	byte
10	Socket Number	word
12	IPX Workspace	byte[4]
16	Driver Workspace	byte[12]
28	Immediate Address	byte[6]
34	Fragment Count	word
36	Fragment Address 1	byte[4]
40	Fragment Size 1	word
42	Fragment Address 2	byte[4]
46	Fragment Size 2	word
...	...	...

ตารางที่ 4.2 โครงสร้างของ ECB

Link Address ถูกใช้โดย IPX/SPX เอง

Event Service Routine Address เก็บตำแหน่งของรูทีนที่จะทำงานเมื่อมีเหตุการณ์ (Event) เกิดขึ้นเช่น มีการรับหรือส่งข้อมูล

In use Flag บอกสถานะของเหตุการณ์

Completion Code บอกสถานะหลังการเรียกใช้ ECB นั้น

Socket number บอกหมายเลขของซอคเก็ต

IPX Wrokspace ใช้โดย IPX เอง

Driver Workspace ใช้โดย IPX เอง

Immediate Address เก็บตำแหน่งของสถานีที่ส่งแพคเก็ตนั้น หรือตำแหน่งของสถานีที่ส่งผ่านแพคเก็ตนั้นมา

Fragment Count บอกจำนวนบัฟเฟอร์ที่ใช้รับหรือส่งข้อมูล

Fragment address เก็บตำแหน่งของบัฟเฟอร์โดยที่ Fragment address ที่ 1 จะต้องชี้ไปยังส่วนหัว (Header) ของโครงสร้างของ IPX หรือ SPX

Fragment size เก็บขนาดของบัฟเฟอร์ Fragment size ที่ 1 จะมีค่าเป็น 30 ถ้าใช้ IPX หรือ 42 ถ้าใช้ SPX

API ที่เกี่ยวข้องกับ Communication Service (Novell Inc.,1989)

- IPX Cancel Event สำหรับยกเลิกการทำงานตาม ECB ที่กำหนด

รูปแบบการเรียกใช้ภาษา C

int IPXCancelEvent (ECB *eventControlBlock);

- IPX Get Interval Marker สำหรับหาค่าช่วงเวลา

รูปแบบการเรียกใช้ภาษา C

WORD IPXGetIntervalMarker (void);

- IPX Relinquish Control สำหรับปล่อยการทำงานจาก CPU ชั่วขณะเพื่อให้ CPU ไปทำงานอย่างอื่น

รูปแบบการเรียกใช้ภาษา C

void IPXRelinquishControl (void);

- IPX Schedule IPX Event สำหรับหน่วงเวลาการทำงาน

รูปแบบการเรียกใช้ภาษา C

void IPXScheduleIPXEvent (WORD timeUnits, ECB *eventControlBlock);

- IPX Close Socket สำหรับปิด Socket ที่ถูกเปิดอยู่

รูปแบบการเรียกใช้ภาษา C

void IPXCloseSocket (WORD socketNumber);

- IPX Disconnect From Target สำหรับยกเลิกการติดต่อกับสถานีงานตามแอดเดรสที่กำหนด

รูปแบบการเรียกใช้ภาษา C

void IPXDisconnectFromTarget (BYTE *networkAddress);

- IPX Get Data Address สำหรับหาแอดเดรสของข้มูลที่กำหนด

รูปแบบการเรียกใช้ภาษา C

void IPXGetDataAddress (BYTE *data, WORD *address);

- IPX Get Internetwork Address สำหรับหาแอดเดรสของสถานีงานนั้น

รูปแบบการเรียกใช้ภาษา C

void IPXGetInternetworkAddress (BYTE *networkAddress);

- IPX Get Local Target สำหรับหาช่วงเวลาในการส่งข้อมูลและแอดเดรสของบริดจ์ (Bridge) ถ้าต้องมีการส่งผ่าน ไปยังสถานีงานที่กำหนด

รูปแบบการเรียกใช้ภาษา C

int IPXGetLocalTarget (BYTE *networkAddress, BYTE *imediateAddress,
int *transportTime);

- IPX Initialize สำหรับเริ่มต้นการทำงานในฟังก์ชันต่างๆของ IPX/SPX

รูปแบบการเรียกใช้ภาษา C

int IPXInitalize (void);

- IPX Listen For Packet สำหรับรับข้อมูลที่ส่งมาโดย IPX

รูปแบบการเรียกใช้ภาษา C

void IPXListenForPacket (ECB *eventControlBlock);

- IPX Open Socket สำหรับเปิด Socket

รูปแบบการเรียกใช้ภาษา C

int IPXOpenSocket (BYTE *socketNumber, BYTE socketType);

- IPX Send Packet สำหรับส่งข้อมูลที่ส่งมาโดย IPX

รูปแบบการเรียกใช้ภาษา C

void IPXSendPAcket (ECB *eventControlBlock);

- SPX Abort Connection สำหรับยกเลิกการติดต่อโดยไม่บอกให้ฝ่ายตรงข้ามรู้

รูปแบบการเรียกใช้ภาษา C

void SPXAbortConnection (WORD connectionIDnumber);

- SPX Establish Connection สำหรับสร้างการเชื่อมต่อกับสถานีงานที่กำหนดไว้ใน ECB

รูปแบบการเรียกใช้ภาษา C

int SPXEstablishConnection(BYTE retryCount, BYTE watchDog,
WORD *connectionID, ECB *eventControlBlock);

- SPX Get Connection Status สำหรับหาสถานะของการเชื่อมต่อตามหมายเลข Connection ที่กำหนด

รูปแบบการเรียกใช้ภาษา C

int SPXGetConnectionStatus (WORD connectIDNumber,
CONNECTION_INFO *connectionInfo);

- SPX Initialize สำหรับตรวจสอบการติดตั้ง SPX และเวอร์ชันของ SPX ที่ใช้งานอยู่

รูปแบบการเรียกใช้ภาษา C

BYTE SPXInitialize (BYTE *majorRevisionNumber, BYTE
*minorRevisionNumber,WORD *maxConnections, WORD
*availableConnections);

- SPX Listen For Connection สำหรับเตรียมรับการเชื่อมต่อจากการเรียกฟังก์ชัน SPXEstablishConnection

รูปแบบการเรียกใช้ภาษา C

void SPXListenForConnection (BYTE retryCount, BYTE watchDog,
ECB *eventControlBlock);

- SPX Listen For Sequenced Packet สำหรับการรับข้อมูลที่ส่งด้วย SPX ตามหมายเลข Connection ที่กำหนด

รูปแบบการเรียกใช้ภาษา C

void SPXListenForSequencedPacket (ECB *eventControlBlock);

- SPX Send Sequenced Packet สำหรับการส่งข้อมูลด้วย SPX ตามหมายเลข Connection ที่กำหนด

รูปแบบการเรียกใช้ภาษา C

void SPXSendSequencedPacket (WORD connectionIDnumber,
ECB *eventContrlBlock);

- SPX Terminate Connection สำหรับยกเลิกการติดต่อโดยบอกให้ฝ่ายตรงข้ามรู้ด้วย

รูปแบบการเรียกใช้ภาษา C

```
void SPXTerminateConnection (WORD connectionIDNumber,
ECB *eventControlBlock);
```