บทที่ 3

การพัฒนาเน็ตสเคปปลั๊กอินบนไมโครซอฟท์วินโดส์ 3.1

ในบทนี้จะกล่าวถึงการพัฒนาเน็ตสเคปปลั๊กอินบนไมโครซอฟท์วินโดส์ 3.1 ว่าต้องศึกษาอะไรบ้าง เครื่องมือที่ใช้ในการพัฒนามีอะไรบ้าง และมีขั้นตอนอย่างไร

เครื่องมือในการพัฒนา

1 . ชุดพัฒนาเน็ตสเคปปลั๊กอิน (Netscape plug-in SDK) ชุดพัฒนานี้มีอยู่ด้วยกันหลายรุ่นแต่รุ่นที่จะกล่าวถึงในบทนี้คือรุ่น 2.0 ชุดพัฒนาเหล่านี้สามารถหาได้จากอินเตอร์เน็ตโดยไปยังสถานที่ต่อไปนี้

ftp.netscape.com

2. ตัวแปลโปรแกรมภาษาซีที่สามารถสร้างคลังโปรแกรมเชื่อมโยงแบบพลวัตได้

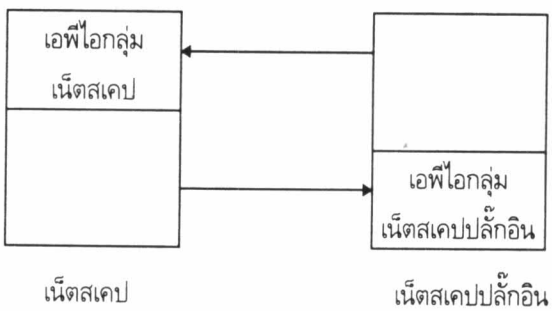
เอพีไอ

ในการที่จะพัฒนาเน็ตสเคปปลั๊กอินนั้น สิ่งหนึ่งที่ผู้พัฒนาจะต้องทำการศึกษาคือเอพีไอ (Netscape Communications, 1995; Oliphant, 1996) ในบทที่ผ่านมาได้มีการกล่าวถึงเอพีไอไปบ้างแล้วว่า เป็นช่องทางที่เน็ตสเคปใช้ในการส่งข้อมูลไปให้กับเน็ตสเคปปลั๊กอินเพื่อทำการจัดการ หรือเป็นช่องทางในการส่ง attribute ที่ปรากฏอยู่ในรหัสคำสั่ง <EMBED> ให้กับเน็ตสเคปปลั๊กอินเพื่อกำหนดการทำงาน นอกเหนือจากตัวอย่างทั้ง 2 นี้แล้ว ยังมีการใช้งานเอพีไอในด้านอื่นๆอีก แต่ก่อนที่จะไปดูว่ามีการใช้งานเอพีไอในด้านใดบ้างนั้น เรามาดูกันก่อนว่าเอพีไอคืออะไร

เอพีไอคือกลุ่มของฟังก์ชันและโครงสร้างข้อมูลที่ทางเน็ตสเคปได้กำหนดขึ้นมาเพื่อใช้เป็นช่องทางการสื่อสารระหว่างเน็ตสเคปกับเน็ตสเคปปลั๊กอินในการแจ้งถึงสถานภาพต่างๆ และยังใช้เป็นช่องทางในการขอใช้บริการอีกด้วย โดยปกติแล้วเอพีไอนี้จะแบ่งออกได้เป็น 2 กลุ่มด้วยกันคือ (รูปที่ 3- 1)

1. เอพีไอกลุ่มเน็ตสเคป เอพีไอในกลุ่มนี้ทางเน็ตสเคปจะเป็นผู้จัดเตรียมไว้ เพื่อให้เน็ตสเคปปลั๊กอินเรียกใช้งาน โดยส่วนใหญ่แล้วเอพีไอในกลุ่มนี้จะเป็นบริการทางด้านการรับส่งข้อมูลผ่านเครือข่าย ชื่อฟังก์ชันที่อยู่ในเอพีไอกลุ่มนี้ทั้งหมดจะขึ้นต้นด้วย NPN (Netscape Plug-in: Navigator defined)

2. เอพีไอกลุ่มเน็ตสเคปปลั๊กอิน เอพีไอในกลุ่มนี้ทางผู้พัฒนาเน็ตสเคปปลั๊กอินจะเป็นผู้จัดเตรียมไว้ เพื่อให้เน็ตสเคปเรียกใช้งาน โดยส่วนใหญ่แล้วเอพีไอในกลุ่มนี้จะมีไว้เพื่อให้เน็ตสเคปแจ้งเหตุการณ์หรือสถานภาพต่างๆให้เน็ตสเคปปลั๊กอินทราบ ชื่อฟังก์ชันที่อยู่ในเอพีไอกลุ่มนี้ทั้งหมดจะขึ้นต้นด้วย NPP (Netscape Plug-in: Plug-in defined)



รูปที่ 3- 1 เอพีไอที่แบ่งออกเป็น 2 กลุ่ม

ตารางที่ 3- 1 แสดงฟังก์ชันทั้งหมดที่มีอยู่ในเอพีไอ (เฉพาะที่มีอยู่บนไมโครซอฟท์วินโดส์) โดยแสดงออกเป็น 2 กลุ่ม (รายละเอียดของแต่ละฟังก์ชันจะกล่าวถึงในหัวข้อถัดไป)

ฟังก์ชันกลุ่มเน็ตสเคป	ฟังก์ชันกลุ่มเน็ตสเคปปลั๊กอิน
NPN_DestroyStream	NPP_Destroy
NPN_GetURL	NPP_DestroyStream
NPN_MemAlloc	NPP_Initialize
NPN_MemFree	NPP_New
NPN_NewStream	NPP_NewStream
NPN_PostURL	NPP_Print
NPN_RequestRead	NPP_SetWindow

ตารางที่ 3- 1 ฟังก์ชันทั้งหมดในเอพีไอแบ่งตามกลุ่ม

ฟังก์ชันกลุ่มเน็ตสเคป	ฟังก์ชันกลุ่มเน็ตสเคปปลั๊กอิน
NPN_Status	NPP_Shutdown
NPN_UserAgent	NPP_StreamAsFile
NPN_Version	NPP_Write
NPN_Write	NPP_WriteReady

ตารางที่ 3- 1 ฟังก์ชันทั้งหมดในเอพีไอแบ่งตามกลุ่ม (ต่อ)

นอกจากการแบ่งเอพีไอออกเป็นกลุ่มตามที่กล่าวมาแล้วข้างต้น เรายังสามารถแบ่งเอพีไอออกเป็นกลุ่มๆ ตามหน้าที่การทำงานได้ด้วยคือ

1. เอพีไอที่ทำหน้าที่ตอนบรรจุเน็ตสเคปปลั๊กอินลงหน่วยความจำและตอนนำออกจากหน่วยความจำ

2. เอพีไอที่ทำหน้าที่สร้างและทำลายข้อมูลในอินสแตน (instance)

3. เอพีไอที่ทำหน้าที่เกี่ยวกับการรับข้อมูลของเน็ตสเคปปลั๊กอิน

4. เอพีไอที่ทำหน้าที่เกี่ยวกับการส่งข้อมูลจากเน็ตสเคปปลั๊กอิน

5. เอพีไอที่ใช้จองและปลดปล่อยหน่วยความจำ

6. เอพีไอที่ทำหน้าที่อื่นๆ เช่น การสอบถามรุ่นของเอพีไอ การพิมพ์ ฯลฯ

กลุ่มของเอพีไอ	ฟังก์ชันที่อยู่ในกลุ่ม
เอพีไอที่ทำหน้าที่ตอนบรรจุเน็ตสเคปปลั๊กอินลงหน่วยความจำและตอนนำออกจากหน่วยความจำ	NPP_Initialize, NPP_Shutdown

ตารางที่ 3- 2 ฟังก์ชันทั้งหมดในเอพีไอแบ่งตามหน้าที่

กลุ่มของเอพีไอ	ฟังก์ชันที่อยู่ในกลุ่ม
เอพีไอที่ทำหน้าที่สร้างและทำลายข้อมูลในอินสแตน	NPP_New, NPP_Destroy
เอพีไอที่ทำหน้าที่เกี่ยวกับการรับข้อมูลของเน็ตสเคปปลั๊กอิน	NPP_NewStream, NPP_DestroyStream, NPP_WriteReady, NPP_Write, NPP_StreamAsFile, NPN_GetURL, NPN_RequestRead
เอพีไอที่ทำหน้าที่เกี่ยวกับการส่งข้อมูลจากเน็ตสเคปปลั๊กอิน	NPN_NewStream, NPN_DestroyStream, NPN_Write, NPN_PostURL
เอพีไอที่ใช้จองและปลดปล่อยหน่วยความจำ	NPN_MemAlloc, NPN_MemFree
เอพีไอที่ทำหน้าที่อื่นๆ	NPP_SetWindow, NPP_Print, NPN_Version, NPN_Status, NPN_UserAgent

ตารางที่ 3- 2 ฟังก์ชันทั้งหมดในเอพีไอแบ่งตามหน้าที่ (ต่อ)

ตารางที่ 3- 2 แสดงฟังก์ชันที่อยู่ในแต่ละกลุ่ม การแบ่งกลุ่มในลักษณะนี้ทำให้ง่ายต่อการเข้าใจรายละเอียดของแต่ละฟังก์ชัน ซึ่งผู้เขียนจะยึดถือการแบ่งกลุ่มนี้เป็นหลักในการอธิบายรายละเอียดของแต่ละฟังก์ชัน โดยจะอธิบายทีละกลุ่มไล่จากกลุ่มแรกเป็นต้นมา การอธิบายนี้จะใช้ไวยากรณ์ (syntax) ของภาษาซีในการประกาศโครงสร้างข้อมูลและฟังก์ชันต่างๆ

ชนิดข้อมูลพื้นฐาน

ในหัวข้อนี้จะแสดงชนิดข้อมูลต่างๆที่มีการอ้างอิงถึงในหัวข้อต่อๆไป ชนิดข้อมูลดังกล่าวมีดังนี้

```
typedef unsigned short  uint16;
typedef unsigned long   uint32;
typedef short           int16;
typedef long            int32;
typedef unsigned char   NPBool;
typedef void*           NPEvent;
typedef int16           NPError;
```

typedef char* NPMIMEType;

เอพีไอที่ทำหน้าที่ตอนบรรจุเน็ตสเคปปลั๊กอินลงหน่วยความจำและตอนนำออกจากหน่วยความจำ

เมื่อเน็ตสเคปพบข้อมูลที่มีเน็ตสเคปปลั๊กอินสามารถจัดการได้ เน็ตสเคปก็จะทำการบรรจุเน็ตสเคปปลั๊กอินนั้นลงในหน่วยความจำถ้าไม่มีการบรรจุมาก่อน หลังจากที่ทำการบรรจุเน็ตสเคปปลั๊กอินลงในหน่วยความจำแล้ว เน็ตสเคปก็จะทำการเรียกฟังก์ชัน NPP_Initialize เพื่อให้โอกาสแก่เน็ตสเคปปลั๊กอินในการตระเตรียมตัวให้พร้อมที่จะทำงาน และเนื่องจากฟังก์ชันนี้จะถูกเรียกเพียงครั้งเดียวเท่านั้นตลอดการทำงานของเน็ตสเคปปลั๊กอิน ดังนั้นสิ่งต่างๆที่ควรจะทำภายในฟังก์ชันนี้ก็คือการตระเตรียมสิ่งของที่ต้องใช้ร่วมกันเช่น ตัวแปรร่วมหรือหน่วยความจำร่วม เป็นต้น

และเมื่อเน็ตสเคปพบว่าไม่มีความจำเป็นที่จะต้องใช้เน็ตสเคปปลั๊กอินอีกต่อไปก็จะนำเน็ตสเคปปลั๊กอิน นั้นออกจากหน่วยความจำ โดยก่อนที่จะทำการนำออกนั้นเน็ตสเคปจะเรียกใช้ฟังก์ชัน NPP_Shutdown เพื่อให้โอกาสแก่เน็ตสเคปปลั๊กอินได้ทำการคืนทรัพยากรต่างๆที่ใช้งานอยู่กลับสู่ระบบ เช่นทำการปลดปล่อยหน่วยความจำที่จองไว้ หรือปิดแฟ้มข้อมูลที่เปิดอยู่ เป็นต้น

รายละเอียดของฟังก์ชันทั้งหมดที่อยู่ในกลุ่มนี้มีดังนี้

NPError NPP_Initialize(void)

จุดประสงค์: ใช้สำหรับเตรียมตัวของเน็ตสเคปปลั๊กอินเองให้พร้อมที่จะทำงานหลังจากถูกบรรจุลงหน่วยความจำแล้ว

พารามิเตอร์: ไม่มี

ค่าส่งกลับ: NPERR_NO_ERROR: แสดงว่าไม่มีข้อผิดพลาดเกิดขึ้น
NPERR_GENERIC_ERROR: แสดงว่ามีข้อผิดพลาดเกิดขึ้น
NPERR_OUT_OF_MEMORY_ERROR: แสดงว่าหน่วยความจำไม่พอ

---

void NPP_Shutdown(void)

จุดประสงค์: ใช้สำหรับคืนทรัพยากรต่างๆที่ใช้อยู่กลับสู่ระบบก่อนที่จะถูกนำออกจากหน่วยความจำ

พารามิเตอร์: ไม่มี

ค่าส่งกลับ: ไม่มี

เอพีไอที่ทำหน้าที่สร้างและทำลายข้อมูลในอินสแตน

ทุกๆครั้งที่เน็ตสเคปได้รับข้อมูลที่มีเน็ตสเคปปลั๊กอินสามารถจัดการได้ ก่อนที่จะส่งข้อมูลนี้ไปให้เน็ตสเคปปลั๊กอินจัดการ เน็ตสเคปจะทำการสร้างอินสแตนขึ้นมาก่อน (เน็ตสเคปปลั๊กอินต้องบรรจุลงหน่วยความจำแล้ว) อินสแตนคือโครงสร้างข้อมูลที่จำเป็นต่อการทำงานใดๆที่เกี่ยวข้องกับข้อมูลนั้น เช่นการแสดงผลของข้อมูล การจัดเก็บข้อมูล เป็นต้น โครงสร้างของอินสแตนมีดังนี้

```
typedef struct _NPP
{
        void    *pdata;
        void    *ndata;
} NPP_t;
typedef NPP_t   *NPP;
```

NPP_t คือโครงสร้างข้อมูลของอินสแตน ข้อมูลในอินสแตนจะแบ่งออกเป็น 2 ส่วนด้วยกันคือ ส่วนที่ใช้งานโดยเน็ตสเคปและส่วนที่ใช้งานโดยเน็ตสเคปปลั๊กอิน ส่วนที่ใช้งานโดยเน็ตสเคปนั้นจะถูกชี้โดยตัวชี้ ndata ขณะที่ส่วนที่ใช้งานโดยเน็ตสเคปปลั๊กอินนั้นจะถูกชี้โดยตัวชี้ pdata ทุกครั้งที่เน็ตสเคปสร้างอินสแตนขึ้นมาใหม่ ข้อมูลส่วนที่ชี้โดย ndata จะถูกสร้างขึ้นมาด้วย ขณะที่ข้อมูลส่วนที่ชี้โดย pdata จะไม่ถูกสร้างขึ้นมา (pdata มีค่าเป็น NULL) เนื่องจากเป็นข้อมูลที่ถูกใช้โดยเน็ตสเคปปลั๊กอิน ดังนั้นเน็ตสเคปจึงต้องทำการเรียกฟังก์ชัน NPP_New ทุกครั้งที่สร้างอินสแตนขึ้นมาเพื่อให้โอกาสเน็ตสเคปปลั๊กอินได้ทำการสร้างข้อมูลที่ต้องการใช้ขึ้นมาแล้วตั้งค่า pdata ให้ชี้ไปยังข้อมูลนั้น

จำนวนอินสแตนในระบบ ณ. ขณะใดขณะหนึ่งอาจจะมีมากกว่าหนึ่งตัวได้ เนื่องจากเราสามารถมีเน็ตสเคปปลั๊กอินหลายตัวทำงานพร้อมกันได้ หรือการที่มีเน็ตสเคปปลั๊กอินตัวเดียวแต่จัดการกับข้อมูลหลายชุด ตัวอย่างเช่น เอกสารเอชทีเอ็มแอลในรูปที่ 3- 2 จะทำให้เกิด 2 อินสแตน

```
<HTML>
<EMBED SRC=song1.wav WIDTH=100 HEIGHT=50>
<EMBED SRC=song2.wav WIDTH=70 HEIGHT=50>
</HTML>
```

รูปที่ 3- 2 ตัวอย่างเอกสารเอชทีเอ็มแอล

อินสแตนของข้อมูลจะคงอยู่ตลอดตราบใดที่ยังมีการใช้งานข้อมูลนั้นอยู่ แต่เมื่อข้อมูลนั้นไม่ได้ถูกใช้งานอีกต่อไปแล้วเน็ตสเคปก็จะทำลายอินสแตนของข้อมูลนั้น แต่ก่อนที่จะทำลายเน็ตสเคปจะเรียกใช้ฟังก์ชัน NPP_Destroy เพื่อให้เน็ตสเคปปลั๊กอินทำการทำลายข้อมูลที่ชี้โดย pdata ก่อน

ภายในฟังก์ชัน NPP_Destroy นี้เน็ตสเคปปลั๊กอินสามารถที่จะเก็บข้อมูลที่จะทำลายไว้บางส่วนหรือทั้งหมดได้โดยทำการย้ายไปเก็บไว้ในหน่วยความจำของเน็ตสเคป ข้อมูลที่เก็บไว้นี้จะถูกส่งกลับมาให้เน็ตสเคปปลั๊กอินผ่านทางพารามิเตอร์ saved ของฟังก์ชัน NPP_New เมื่อข้อมูลที่ไม่ได้ถูกใช้งานนี้ถูกนำกลับมาใช้ใหม่ ขั้นตอนในการย้ายข้อมูลไปเก็บไว้ในหน่วยความจำของเน็ตสเคปมีดังนี้

1. ทำการจองหน่วยความจำสำหรับโครงสร้างข้อมูล NPSavedData ขึ้นมาโดยใช้ฟังก์ชัน NPN_MemAlloc จากนั้นเก็บตำแหน่งของหน่วยความจำนี้ไว้ในตัวชี้ที่ถูกชี้โดยพารามิเตอร์ save ของฟังก์ชัน NPP_Destroy โครงสร้างของ NPSavedData มีดังนี้

```
typedef struct _NPSavedData
{
        int32   len;
        void    *buf;
} NPSavedData;
```

2. ทำการจองหน่วยความจำที่ใช้สำหรับเก็บข้อมูลโดยใช้ฟังก์ชัน NPN_MemAlloc เก็บจำนวนไบต์ของหน่วยความจำนี้ไว้ใน len และเก็บตำแหน่งของหน่วยความจำนี้ไว้ใน buf จากนั้นทำการสำเนาข้อมูลต่างๆที่ต้องการเก็บลงในหน่วยความจำนี้

อนึ่งถึงแม้เราสามารถที่จะเก็บข้อมูลไว้ในหน่วยความจำของเน็ตสเคปได้แต่ข้อมูลที่เก็บนี้ไม่ควรจะมีความสำคัญมาก เนื่องจากเน็ตสเคปสามารถนำหน่วยความจำนี้ไปใช้งานอย่างอื่นได้ถ้ามีความจำเป็น (นั่นคือข้อมูลที่เก็บไว้จะไม่ถูกส่งกลับมาให้เมื่อมีการเรียกฟังก์ชัน NPP_New) และข้อมูลที่เก็บนี้ไม่ควรจะมีตัวชี้ที่ชี้ไปยังหน่วยความจำอื่นๆอีก ทั้งนี้เนื่องจากเน็ตสเคปไม่มีความรู้เกี่ยวกับโครงสร้างข้อมูลที่เราเก็บไว้ ดังนั้นจึงไม่สามารถติดตามตัวชี้ต่างๆเพื่อไปปลดปล่อยหน่วยความจำเหล่านี้ได้

รายละเอียดของฟังก์ชันทั้งหมดที่อยู่ในกลุ่มนี้มีดังนี้

NPError NPP_New(NPMIMEType pluginType,
NPP instance,
uint16 mode,
int16 argc,
char *argn[],
char *argv[],
NPSavedData *saved)

จุดประสงค์: ใช้สำหรับสร้างข้อมูลที่เน็ตสเคปปลั๊กอินจำเป็นต้องใช้งานขึ้นมาและเชื่อมโยงข้อมูลนี้เข้าเป็นส่วนหนึ่งของอินสแตน (ผ่านทาง pdata)

พารามิเตอร์:

pliginType	= ตัวชี้ไปยังสายอักขระ (string) ที่บอกถึงชนิดข้อมูลซึ่งเขียนตามรูปแบบเอ็มไอเอ็มอี
instance	= ตัวชี้ไปยังอินสแตนที่ถูกสร้างขึ้นมา
mode	= วิธีที่เรียกใช้งานเน็ตสเคปปลั๊กอิน ค่าที่เป็นไปได้ของ mode คือ NP_FULL: เรียกใช้งานแบบเต็มหน้า NP_EMBED: เรียกใช้งานแบบฝังหรือแบบซ่อน
argc	= จำนวน attribute ในรหัสคำสั่ง <EMBED>
argn	= แถวลำดับ (array) ของตัวชี้ที่ชี้ไปยังสายอักขระของส่วน name ของ attribute ทั้งหมดในรหัสคำสั่ง <EMBED>
argv	= แถวลำดับของตัวชี้ที่ชี้ไปยังสายอักขระของส่วน value ของ attribute ทั้งหมดในรหัสคำสั่ง <EMBED>
saved	= ตัวชี้ไปยังโครงสร้างข้อมูล NPSavedData ที่เก็บไว้ในหน่วยความจำของเน็ตสเคปโดยฟังก์ชัน NPP_Destroy

ค่าส่งกลับ:
NPERR_NO_ERROR: แสดงว่าไม่มีข้อผิดพลาดเกิดขึ้น
NPERR_GENERIC_ERROR: แสดงว่ามีข้อผิดพลาดเกิดขึ้น
NPERR_OUT_OF_MEMORY_ERROR: แสดงว่าหน่วยความจำไม่พอ
NPERR_INVALID_INSTANCE_ERROR: แสดงว่าค่าของ instance ไม่ถูกต้องเช่นเป็น NULL

NPError NPP_Destroy(NPP instance, NPSavedData **save)

จุดประสงค์: ใช้สำหรับทำลายข้อมูลในอินสแตนส่วนที่ใช้งานโดยเน็ตสเคปปลั๊กอิน (ส่วนที่ชี้โดย pdata) และทำการเก็บข้อมูลที่ต้องการไว้ในหน่วยความจำของเน็ตสเคป

พารามิเตอร์: instance = ตัวชี้ไปยังอินสแตนที่ต้องการทำลาย

save = ตัวชี้ไปยังตัวชี้ที่ชี้ไปยังโครงสร้างข้อมูล NPSavedData ถ้าเน็ตสเคปปลั๊กอินต้องการเก็บข้อมูลไว้ในหน่วยความจำของเน็ตสเคป หลังจากที่ทำตามขั้นตอนที่กล่าวมาแล้ว ตำแหน่งของโครงสร้างข้อมูล NPSavedData จะต้องมาเก็บไว้ที่ตัวชี้ที่ถูกชี้โดย save (นั่นคือ *save)

ค่าส่งกลับ: NPERR_NO_ERROR: แสดงว่าไม่มีข้อผิดพลาดเกิดขึ้น

NPERR_GENERIC_ERROR: แสดงว่ามีข้อผิดพลาดเกิดขึ้น

NPERR_OUT_OF_MEMORY_ERROR: แสดงว่าหน่วยความจำไม่พอ

NPERR_INVALID_INSTANCE_ERROR: แสดงว่าค่าของ instance ไม่ถูกต้องเช่นเป็น NULL

เอพีไอที่ทำหน้าที่เกี่ยวกับการรับข้อมูลของเน็ตสเคปปลั๊กอิน

จากหัวข้อที่ผ่านมาอินสแตนจะถูกสร้างขึ้นก่อนที่เน็ตสเคปจะทำการส่งข้อมูลที่ได้มาให้กับเน็ตสเคปปลั๊กอิน หลังจากที่อินสแตนถูกสร้างขึ้นมาอย่างสมบูรณ์แล้ว (นั่นคือหลังจากมีการเรียกใช้ฟังก์ชัน NPP_New แล้ว) เน็ตสเคปก็จะทำการส่งข้อมูลไปให้กับเน็ตสเคปปลั๊กอิน แต่ก่อนที่การส่งข้อมูลจะเริ่มขึ้นเน็ตสเคปปลั๊กอินจะทำการสร้างสตรีม (stream) ขึ้นมาก่อน สตรีมคือโครงสร้างข้อมูลที่จำเป็นต้องใช้ในระหว่างการส่งข้อมูลจากเน็ตสเคปไปยังเน็ตสเคปปลั๊กอิน ซึ่งมีโครงสร้างดังนี้

```
typedef struct _NPStream
{
    void        *pdata;
    void        *ndata;
    const char  *url;
    uint32      end;
    uint32      lastmodified;
} NPStream;
```

การใช้งาน pdata และ ndata จะเหมือนกับอินสแตนคือ pdata เป็นตัวชี้ไปยังข้อมูลที่ถูกใช้โดยเน็ตสเคปปลั๊กอินในระหว่างรับข้อมูล และ ndata เป็นตัวชี้ไปยังข้อมูลที่ถูกใช้โดยเน็ตสเคปในระหว่างส่งข้อมูล url เป็นตัวชี้ไปยังสายอักขระของยูอาร์แอลที่ทำให้เกิดการส่งข้อมูล end คือจำนวนไบต์ที่จะมีการส่ง (ค่าของ end อาจเป็น 0 ได้ถ้า

เน็ตสเคปไม่สามารถหาขนาดของข้อมูลได้) lastmodified คือเวลาการแก้ไขครั้งล่าสุดของข้อมูลในยูอาร์แอล (นับเป็นวินาทีตั้งแต่เที่ยงคืนของวันที่ 1 มกราคม ค.ศ. 1970 ตามเวลามาตราฐาน)

ทุกครั้งที่เน็ตสเคปสร้างสตรีมขึ้นมาใหม่ข้อมูลทุกส่วนใน NPStream จะถูกใส่มาให้แล้วยกเว้น pdata ซึ่งมีค่าเป็น NULL เนื่องจากเป็นข้อมูลที่ถูกใช้โดยเน็ตสเคปปลั๊กอิน ดังนั้นเน็ตสเคปจึงต้องทำการเรียกฟังก์ชัน NPP_NewStream ทุกครั้งที่สร้างสตรีมขึ้นมาเพื่อให้โอกาสเน็ตสเคปปลั๊กอินได้ทำการสร้างข้อมูลที่ต้องการใช้ขึ้นมาแล้วตั้งค่า pdata ให้ชี้ไปยังข้อมูลนั้น หน้าที่ของ NPP_NewStream นอกจากใช้สำหรับสร้างข้อมูลที่จำเป็นในการรับข้อมูลจากเน็ตสเคปแล้วยังใช้กำหนดวิธีในการส่งข้อมูลของเน็ตสเคปด้วยโดยการส่งค่ากลับไปยังเน็ตสเคปผ่านทางพารามิเตอร์ stype ซึ่งมีค่าได้มีดังนี้

1. NP_NORMAL: วิธีนี้เน็ตสเคปจะส่งข้อมูลที่ได้มาผ่านทางฟังก์ชัน NPP_WriteReady และ NPP_Write

2. NP_ASFILE: วิธีนี้จะเหมือนกับ NP_NORMAL แต่จะทำการเก็บข้อมูลทั้งหมดลงในแฟ้มข้อมูลด้วย จากนั้นจะส่งชื่อของแฟ้มข้อมูลนั้นมาให้ผ่านทางฟังก์ชัน NPP_StreamAsFile

3. NP_SEEK: วิธีนี้จะทำการส่งข้อมูลผ่านทางฟังก์ชัน NPP_WriteReady และ NPP_Write ก็ต่อเมื่อมีการระบุช่วงของข้อมูลที่ต้องการผ่านทางฟังก์ชัน NPN_RequestRead ก่อน

การส่งข้อมูลนี้จะเริ่มขึ้นเมื่อสตรีมถูกสร้างขึ้นมาอย่างสมบูรณ์แล้ว (นั่นคือหลังจากมีการเรียกใช้ฟังก์ชัน NPP_NewStream แล้ว) แต่มีสิ่งที่ควรรู้เกี่ยวกับพฤติกรรมการส่งข้อมูลของเน็ตสเคปผ่านทางฟังก์ชัน NPP_WriteReady และ NPP_Write ดังนี้

1. ข้อมูลทั้งหมดที่จะส่งไปให้กับเน็ตสเคปปลั๊กอินอาจจะถูกแบ่งออกเป็นส่วนๆ โดยทยอยส่งทีละส่วน

2. การส่งข้อมูล 1 ส่วนของเน็ตสเคปจะเรียกใช้ทั้งฟังก์ชัน NPP_WriteReady และ NPP_Write โดยจะทำการเรียก NPP_WriteReady ก่อน NPP_Write เสมอ

และเมื่อเน็ตสเคปทำการส่งข้อมูลทั้งหมดให้กับเน็ตสเคปปลั๊กอินแล้ว หรือมีการยกเลิกโดยผู้ใช้ หรือมีปัญหาเกี่ยวกับเครือข่าย เน็ตสเคปจะทำการทำลายสตรีมของข้อมูลนั้น (ยกเว้นวิธีในการส่งข้อมูลแบบ NP_SEEK และ

ได้รับข้อมูลทั้งหมด) แต่ก่อนที่จะทำลาย เน็ตสเคปจะเรียกใช้ฟังก์ชัน NPP_DestroyStream เพื่อให้เน็ตสเคปปลั๊กอินทำการทำลายข้อมูลที่ชี้โดย pdata ก่อน

ที่กล่าวมาข้างต้นทั้งหมดในหัวข้อนี้เป็นกระบวนการรับข้อมูลที่เกิดขึ้นจากการกระทำของผู้ใช้ เช่นมีการระบุยูอาร์แอลของเอกสารเอชทีเอ็มแอลที่มีรหัสคำสั่ง <EMBED> อยู่เป็นต้น แต่เราสามารถทำให้กระบวนการรับข้อมูลนี้เกิดขึ้นได้เองโดยไม่ต้องอาศัยผู้ใช้โดยการเรียกใช้ฟังก์ชัน NPN_GetURL ฟังก์ชันนี้ใช้สำหรับดึงข้อมูลจากยูอาร์แอลที่กำหนดมาให้แก่เน็ตสเคปปลั๊กอินหรือเน็ตสเคป ในกรณีที่ดึงมาให้กับเน็ตสเคปปลั๊กอินก็จะเกิดกระบวนการรับข้อมูลที่กล่าวมาแล้วขึ้น การจะระบุว่าข้อมูลที่ดึงมานั้นจะให้ใครจะกำหนดโดยพารามิเตอร์ window ของฟังก์ชัน NPN_GetURL พารามิเตอร์นี้สามารถมีค่าดังต่อไปนี้

1. ตัวชี้ไปยังสายอักขระที่เป็นชื่อพิเศษ ซึ่งจะทำให้ข้อมูลที่ดึงมาถูกส่งไปให้กับเน็ตสเคปเพื่อทำการแสดงผล สถานที่ที่ใช้ในการแสดงผลนี้ขึ้นอยู่กับค่าของชื่อพิเศษ ซึ่งมีด้วยกันดังนี้

1.1. _blank หรือ _new: เน็ตสเคปจะสร้างหน้าต่างขึ้นมาใหม่แล้วแสดงผลบนหน้าต่างนั้น

1.2. _self หรือ _current: แสดงผลบนหน้าต่างของเน็ตสเคปหรือเฟรม (frame) (Netscape Communications, 1996) ที่อินสแตนที่เรียกใช้ฟังก์ชันนี้แสดงผลอยู่

1.3. _parent: แสดงผลบนเฟรมที่เป็นพ่อ (parent) ของเฟรมที่อินสแตนที่เรียกใช้ฟังก์ชันนี้แสดงผลอยู่ ถ้าอินสแตนแสดงผลอยู่ในหน้าต่างหรือเฟรมที่ไม่มีพ่อผลที่ได้จะเหมือนกับ _top

1.4. _top: แสดงผลบนหน้าต่างของเน็ตสเคปที่อินสแตนที่เรียกใช้ฟังก์ชันนี้แสดงผลอยู่

2. ตัวชี้ไปยังสายอักขระที่เป็นชื่อเฟรม ซึ่งจะทำให้ข้อมูลที่ดึงมาถูกส่งไปให้กับเน็ตสเคปเพื่อทำการแสดงผลในเฟรมที่กำหนด

3. NULL ซึ่งจะทำให้ข้อมูลที่ดึงมาถูกส่งไปให้กับเน็ตสเคปปลั๊กอินผ่านทางกระบวนการรับข้อมูลที่กล่าวมาแล้ว

รายละเอียดของฟังก์ชันทั้งหมดที่อยู่ในกลุ่มนี้มีดังนี้

```
NPError NPP_NewStream(NPP instance,
                      NPMIMEType type,
                      NPStream *stream,
```

NPBool seekable,

uint16 *stype)

จุดประสงค์: ใช้สำหรับสร้างข้อมูลที่เน็ตสเคปปลั๊กอินจำเป็นต้องใช้ในการรับข้อมูลขึ้นมาพร้อมกับเชื่อมโยงข้อมูลนี้เข้าเป็นส่วนหนึ่งของสตรีม (ผ่านทาง pdata) และกำหนดวิธีการส่งข้อมูลของเน็ตสเคป

พารามิเตอร์:

instance = ตัวชี้ไปยังอินสแตนที่จะรับข้อมูล

type = ตัวชี้ไปยังสายอักขระที่บอกถึงชนิดของข้อมูลที่จะได้รับ ซึ่งเขียนตามรูปแบบเอ็มไอเอ็มอี

stream = ตัวชี้ไปยังสตรีมที่ถูกสร้างขึ้นมา

seekable = เป็นตัวแสดงว่าข้อมูลนี้สามารถเข้าถึงโดยสุ่ม (random access) ได้หรือไม่ ถ้าค่าเป็น TRUE แสดงว่าได้ซึ่งทำให้เราสามารถใช้วิธีการส่งแบบที่ 3 ได้ ถ้าค่าเป็น FALSE แสดงว่าไม่ได้ ถ้าเราใช้วิธีการส่งแบบที่ 3 กับข้อมูลที่เข้าถึงโดยสุ่มไม่ได้ เน็ตสเคปจะทำการเก็บข้อมูลทั้งหมดลงในแฟ้มข้อมูลก่อนจากนั้นจึงทำการสุ่มจากแฟ้มข้อมูล

stype = ตัวชี้ไปยังข้อมูลที่ใช้กำหนดวิธีในการส่งข้อมูลของเน็ตสเคป ค่าที่เป็นไปได้ของข้อมูลที่ถูกชี้โดย stype คือ

NP_NORMAL

NP_ASFILE

NP_SEEK

ค่าส่งกลับ:

NPERR_NO_ERROR: แสดงว่าไม่มีข้อผิดพลาดเกิดขึ้น

NPERR_GENERIC_ERROR: แสดงว่ามีข้อผิดพลาดเกิดขึ้น

NPERR_OUT_OF_MEMORY_ERROR: แสดงว่าหน่วยความจำไม่พอ

NPERR_INVALID_INSTANCE_ERROR: แสดงว่าค่าของ instance ไม่ถูกต้องเช่นเป็น NULL

---

int32 NPP_WriteReady(NPP instance, NPStream *stream)

จุดประสงค์: ใช้สำหรับสอบถามจำนวนข้อมูลที่เน็ตสเคปปลั๊กอินสามารถจะรับได้ในขณะนั้น เน็ตสเคปจะทำการเรียกฟังก์ชันนี้ก่อนหน้าการส่งข้อมูลให้เน็ตสเคปปลั๊กอินผ่านฟังก์ชัน NPP_Write เสมอ

พารามิเตอร์:

instance = ตัวชี้ไปยังอินสแตนที่จะรับข้อมูล

stream = ตัวชี้ไปยังสตรีมที่ใช้ในการรับข้อมูล

ค่าส่งกลับ: จำนวนไบต์ของข้อมูลที่เน็ตสเคปปลั๊กอินสามารถรับได้ในการรับข้อมูลครั้งต่อไปทาง NPP_Write

int32 NPP_Write(NPP instance, NPStream *stream, int32 offset, int32 len, void *buf)

จุดประสงค์: ใช้สำหรับรับข้อมูลที่ส่งมาจากเน็ตสเคป

พารามิเตอร์:

instance	= ตัวชี้ไปยังอินสแตนที่จะรับข้อมูล
stream	= ตัวชี้ไปยังสตรีมที่ใช้ในการรับข้อมูล
offset	= ตำแหน่งของข้อมูลใน buf เทียบกับจุดเริ่มต้นของข้อมูล
len	= จำนวนไบต์ของข้อมูลที่ส่งมา
buf	= ตัวชี้ไปยังข้อมูลที่ส่งมา

ค่าส่งกลับ: จำนวนไบต์ของข้อมูลที่เน็ตสเคปปลั๊กอินรับไป ซึ่งอาจจะเป็นทั้งหมดหรืออย่างน้อยต้องเท่ากับที่ระบุไว้ใน NPP_WriteReady ถ้าค่านี้ติดลบจะทำให้เกิดการยกเลิกการส่งข้อมูลและทำให้สตรีมถูกทำลาย

---

void NPP_StreamAsFile(NPP instance, NPStream *stream, const char *fname)

จุดประสงค์: ใช้สำหรับระบุชื่อแฟ้มข้อมูลที่เน็ตสเคปใช้เก็บข้อมูลที่ส่งมา ฟังก์ชันนี้จะถูกเรียกต่อเมื่อมีการกำหนดวิธีส่งข้อมูลเป็นแบบ NP_ASFILE

พารามิเตอร์:

instance	= ตัวชี้ไปยังอินสแตนที่จะรับข้อมูล
stream	= ตัวชี้ไปยังสตรีมที่ใช้ในการรับข้อมูล
fname	= ตัวชี้ไปยังสายอักขระที่ระบุชื่อแฟ้มข้อมูล

ค่าส่งกลับ: ไม่มี

NPError NPP_DestroyStream(NPP instance, NPStream *stream, NPError reason)

จุดประสงค์: ใช้สำหรับทำลายข้อมูลในสตรีมส่วนที่ใช้งานโดยเน็ตสเคปปลั๊กอิน (ส่วนที่ชี้โดย pdata) พร้อมกับบอกเหตุผลในการทำลาย

พารามิเตอร์:

instance	= ตัวชี้ไปยังอินสแตนที่ใช้สตรีมที่จะถูกทำลาย
stream	= ตัวชี้ไปยังสตรีมที่จะทำลาย
reason	= เหตุผลในการทำลายสตรีม ซึ่งค่าที่เป็นไปได้มีดังนี้

NPRES_DONE: ข้อมูลถูกส่งไปหมดแล้ว

NPRES_NETWORK_ERR: เครือข่ายมีปัญหา

NPRES_USER_BREAK: ถูกยกเลิกโดยผู้ใช้

ค่าส่งกลับ: NPERR_NO_ERROR: แสดงว่าไม่มีข้อผิดพลาดเกิดขึ้น

NPERR_GENERIC_ERROR: แสดงว่ามีข้อผิดพลาดเกิดขึ้น

NPERR_OUT_OF_MEMORY_ERROR: แสดงว่าหน่วยความจำไม่พอ

NPERR_INVALID_INSTANCE_ERROR: แสดงว่าค่าของ instance ไม่ถูกต้องเช่นเป็น NULL

NPError NPN_RequestRead(NPStream *stream, NPByteRange *rangeList)

จุดประสงค์: ใช้สำหรับดึงกลุ่มข้อมูลที่ต้องการ การกำหนดกลุ่มข้อมูลที่ต้องการจะทำโดยการสร้างรายการเชื่อมโยง (linked list) ของโครงสร้างข้อมูล NPByteRange ขึ้นมา ซึ่งมีโครงสร้างดังนี้

```
typedef struct _NPByteRange
{
    int32           offset;
    uint32          length;
    struct _NPByteRange    *next;
} NPByteRange;
```

offset คือตำแหน่งเริ่มต้นข้อมูลที่ต้องการนับจากจุดเริ่มต้นของข้อมูลถ้าเป็นค่าบวก แต่ถ้าค่าเป็นลบจะนับจากท้ายข้อมูล length คือจำนวนไบต์ของข้อมูลที่ต้องการ next คือตัวชี้ไปยังโครงสร้างข้อมูล NPByteRange ถัดไปถ้าไม่มีให้ใส่ค่า NULL

พารามิเตอร์: stream = ตัวชี้ไปยังสตรีมที่ใช้ในการดึงข้อมูล

rangeList = ตัวชี้ไปยังรายการเชื่อมโยงของโครงสร้างข้อมูล NPByteRange

ค่าส่งกลับ: NPERR_NO_ERROR: แสดงว่าไม่มีข้อผิดพลาดเกิดขึ้น

NPERR_GENERIC_ERROR: แสดงว่ามีข้อผิดพลาดเกิดขึ้น

NPError NPN_GetURL(NPP instance, const char *url, const char *window)

จุดประสงค์: ใช้สำหรับดึงข้อมูลจากยูอาร์แอลที่กำหนดมาให้กับเน็ตสเคปปลั๊กอินหรือเน็ตสเคป อนึ่งการดึงข้อมูลนี้จะเป็นแบบไม่ประสานจังหวะ (asynchronous) คือฟังก์ชันนี้จะคืนการทำงานกลับมาทันทีโดยไม่รอให้ข้อมูลที่ต้องการมาครบ

พารามิเตอร์: instance = ตัวชี้ไปยังอินสแตนที่ทำการดึงข้อมูล

url = ตัวชี้ไปยังสายอักขระของยูอาร์แอลที่ต้องการดึงข้อมูลมา

window = NULL หรือตัวชี้ไปยังสายอักขระที่เป็นชื่อพิเศษหรือชื่อเฟรมที่ใช้สำหรับแสดงผลข้อมูลที่ดึงมา

ค่าส่งกลับ: NPERR_NO_ERROR: แสดงว่าไม่มีข้อผิดพลาดเกิดขึ้น
NPERR_GENERIC_ERROR: แสดงว่ามีข้อผิดพลาดเกิดขึ้น
NPERR_OUT_OF_MEMORY_ERROR: แสดงว่าหน่วยความจำไม่พอ
NPERR_INVALID_INSTANCE_ERROR: แสดงว่าค่าของ instance ไม่ถูกต้องเช่นเป็น NULL

เอพีไอที่ทำหน้าที่เกี่ยวกับการส่งข้อมูลจากเน็ตสเคปปลั๊กอิน

ในหัวข้อนี้จะกล่าวถึงการส่งข้อมูลจากเน็ตสเคปปลั๊กอินไปให้กับเน็ตสเคปหรือยูอาร์แอล กระบวนการส่งข้อมูลจากเน็ตสเคปปลั๊กอินไปยังเน็ตสเคปจะคล้ายกับกระบวนการรับข้อมูลคือทุกครั้งที่จะมีการส่งข้อมูลจะต้องทำการสร้างสตรีมขึ้นมาก่อนโดยใช้ฟังก์ชัน NPN_NewStream หลังจากสตรีมถูกสร้างขึ้นมาแล้ว การส่งข้อมูลจะใช้ฟังก์ชัน NPN_Write และเมื่อหมดข้อมูลที่จะส่งแล้วจะทำลายสตรีมโดยใช้ฟังก์ชัน NPN_DestroyStream ในกรณีที่เน็ตสเคปปลั๊กอินต้องการส่งข้อมูลไปให้กับยูอาร์แอลจะทำได้โดยใช้ฟังก์ชัน NPN_PostURL

รายละเอียดของฟังก์ชันทั้งหมดที่อยู่ในกลุ่มนี้มีดังนี้

NPError NPN_NewStream(NPP instance, NPMIMEType type, const char *target, NPStream **stream)

จุดประสงค์: ใช้สำหรับสร้างสตรีมที่จะใช้ในการส่งข้อมูลจากเน็ตสเคปปลั๊กอินไปให้กับเน็ตสเคป ฟังก์ชันนี้ใช้กับเน็ตสเคปรุ่น 2 ไม่ได้

พารามิเตอร์:
- instance = ตัวชี้ไปยังอินสแตนที่จะส่งข้อมูล
- type = ตัวชี้ไปยังสายอักขระที่บอกถึงชนิดของข้อมูลที่จะส่ง ซึ่งเขียนตามรูปแบบเอ็มไอเอ็มอี
- target = ตัวชี้ไปยังสายอักขระที่เป็นชื่อพิเศษหรือชื่อเฟรมที่ใช้สำหรับแสดงผลข้อมูลที่ส่งไป (ค่าเหมือนกับ NPN_GetURL)
- stream = ตัวชี้ไปยังตัวชี้ที่ชี้ไปยังโครงสร้างข้อมูล NPStream โครงสร้างนี้เน็ตสเคปจะทำการสร้างให้และใส่ตำแหน่งของโครงสร้างนี้ใน *stream

ค่าส่งกลับ: NPERR_NO_ERROR: แสดงว่าไม่มีข้อผิดพลาดเกิดขึ้น
NPERR_GENERIC_ERROR: แสดงว่ามีข้อผิดพลาดเกิดขึ้น
NPERR_OUT_OF_MEMORY_ERROR: แสดงว่าหน่วยความจำไม่พอ
NPERR_INVALID_INSTANCE_ERROR: แสดงว่าค่าของ instance ไม่ถูกต้องเช่นเป็น NULL

int32 NPN_Write(NPP instance, NPStream *stream, int32 len, void *buf)

จุดประสงค์: ใช้สำหรับส่งข้อมูลจากเน็ตสเคปปลั๊กอินไปให้กับเน็ตสเคป ฟังก์ชันนี้ใช้กับเน็ตสเคปรุ่น 2 ไม่ได้

พารามิเตอร์:
- instance = ตัวชี้ไปยังอินสแตนที่จะส่งข้อมูล
- stream = ตัวชี้ไปยังสตรีมที่สร้างจากฟังก์ชัน NPN_NewStream
- len = จำนวนไบต์ที่จะส่ง
- buf = ตัวชี้ไปยังข้อมูลที่จะส่ง

ค่าส่งกลับ: จำนวนไบต์ที่รับโดยเน็ตสเคป ถ้าค่าเป็นลบแสดงว่ามีข้อผิดพลาดเกิดขึ้นและเน็ตสเคปปลั๊กอินควรจะทำลายสตรีมโดยใช้ฟังก์ชัน NPN_DestroyStream

---

NPError NPN_DestroyStream(NPP instance, NPStream *stream, NPError reason)

จุดประสงค์: ใช้สำหรับทำลายสตรีมที่สร้างจากฟังก์ชัน NPP_NewStream และ NPN_NewStream ฟังก์ชันนี้ใช้กับเน็ตสเคปรุ่น 2 ไม่ได้

พารามิเตอร์:
- instance = ตัวชี้ไปยังอินสแตนที่จะทำลายสตรีม
- stream = ตัวชี้ไปยังสตรีมที่จะทำลาย
- reason = เหตุผลในการทำลายสตรีม (เหมือนฟังก์ชัน NPP_DestroyStream)

ค่าส่งกลับ:
- NPERR_NO_ERROR: แสดงว่าไม่มีข้อผิดพลาดเกิดขึ้น
- NPERR_GENERIC_ERROR: แสดงว่ามีข้อผิดพลาดเกิดขึ้น
- NPERR_OUT_OF_MEMORY_ERROR: แสดงว่าหน่วยความจำไม่พอ
- NPERR_INVALID_INSTANCE_ERROR: แสดงว่าค่าของ instance ไม่ถูกต้องเช่นเป็น NULL

NPError NPN_PostURL(NPP instance,
const char *url,
const char *window,
uint32 len,
const char *buf,
NPBool file)

จุดประสงค์: ใช้สำหรับส่งข้อมูลจากเน็ตสเคปปลั๊กอินไปที่ยูอาร์แอลที่กำหนดพร้อมกับรับผลลัพท์ของการส่งด้วย อนึ่งการส่งข้อมูลนี้จะเป็นแบบไม่ประสานจังหวะ (asynchronous) คือฟังก์ชันนี้จะคืนการทำงานกลับมาทันทีโดยไม่รอให้ข้อมูลที่ต้องการถูกส่งหมด ฟังก์ชันนี้ใช้กับเน็ตสเคปรุ่น 2 ไม่ได้

พารามิเตอร์: instance = ตัวชี้ไปยังอินสแตนที่จะส่งข้อมูล

url = ตัวชี้ไปยังสายอักขระของยูอาร์แอลที่จะรับข้อมูล

window = NULL หรือตัวชี้ไปยังสายอักขระที่เป็นชื่อพิเศษหรือชื่อเฟรมที่ใช้สำหรับแสดงผลลัพท์ของการส่งข้อมูล (ค่าเหมือนกับ NPN_GetURL)

len = จำนวนไบต์ที่จะส่ง

buf = ตัวชี้ไปยังข้อมูลที่จะส่ง

file = TRUE ถ้าข้อมูลที่จะส่งอยู่ในแฟ้มข้อมูล ในกรณีนี้ buf จะเป็นตัวชี้ไปยังชื่อแฟ้มข้อมูลนั้นและ len คือความยาวของชื่อแฟ้มข้อมูล ถ้า file มีค่าเป็น FALSE ข้อมูลที่จะส่งจะอยู่ในหน่วยความจำ ในกรณีนี้ buf จะเป็นตัวชี้ไปยังข้อมูลและ len คือจำนวนไบต์ของข้อมูล

ค่าส่งกลับ: NPERR_NO_ERROR: แสดงว่าไม่มีข้อผิดพลาดเกิดขึ้น

NPERR_GENERIC_ERROR: แสดงว่ามีข้อผิดพลาดเกิดขึ้น

NPERR_OUT_OF_MEMORY_ERROR: แสดงว่าหน่วยความจำไม่พอ

NPERR_INVALID_INSTANCE_ERROR: แสดงว่าค่าของ instance ไม่ถูกต้องเช่นเป็น NULL

เอพีไอที่ใช้จองและปลดปล่อยหน่วยความจำ

เอพีไอในกลุ่มนี้ใช้สำหรับการจองหน่วยความจำและปลดปล่อยหน่วยความจำของเน็ตสเคป รายละเอียดของฟังก์ชันทั้งหมดที่อยู่ในกลุ่มนี้มีดังนี้

---

void *NPN_MemAlloc(uint32 size)

จุดประสงค์: ใช้สำหรับจองหน่วยความจำของเน็ตสเคป หน่วยความจำนี้สามารถนำไปใช้งานได้เหมือนกับหน่วยความจำปกติ จุดประสงค์อีกอย่างหนึ่งของฟังก์ชันนี้คือเอาไว้ใช้งานกับ NPP_Destroy ในกรณีที่ต้องการเก็บข้อมูลบางอย่างไว้เพื่อใช้งานเมื่อมีการสร้างอินสแตนขึ้นมาใหม่

พารามิเตอร์: size = จำนวนไบต์ของหน่วยความจำที่ต้องการจอง

ค่าส่งกลับ: ตัวชี้ไปยังหน่วยความจำที่จองไว้ให้ หรือ NULL ถ้าไม่สามารถจองหน่วยความจำให้ได้

---

void NPN_MemFree(void *ptr)

จุดประสงค์: ใช้สำหรับปลดปล่อยหน่วยความจำที่จองไว้โดยฟังก์ชัน NPN_MemAlloc

พารามิเตอร์: ptr = ตัวชี้ไปยังหน่วยความจำที่ถูกจองโดยฟังก์ชัน NPN_MemAlloc
ค่าส่งกลับ: ไม่มี

เอพีไอที่ทำหน้าที่อื่นๆ

เอพีไอในกลุ่มนี้ใช้สำหรับสอบถามข้อมูลต่างๆจากเน็ตสเคป แสดงข้อความแสดงสถานภาพ พิมพ์ข้อมูล และรับหน้าต่างที่ใช้ในการแสดงผล รายละเอียดของฟังก์ชันทั้งหมดที่อยู่ในกลุ่มนี้มีดังนี้

NPError NPP_SetWindow(NPP instance, NPWindow *window)

จุดประสงค์: ใช้สำหรับส่งแฮนเดิล (handle) ของหน้าต่างที่เน็ตสเคปปลั๊กอินสามารถใช้ในการแสดงผล แฮนเดิลนี้จะอยู่ในโครงสร้างข้อมูล NPWindow ซึ่งมีโครงสร้างดังนี้

```
typedef struct _NPRect
{
        uint16      top;
        uint16      left;
        uint16      bottom;
        uint16      right;
} NPRect;
typedef struct _NPWindow
{
        void        *window;
        uint32      x;
        uint32      y;
        uint32      width;
        uint32      height;
        NPRect      clipRect;
} NPWindow;
```

window คือแฮนเดิล (handle) ของหน้าต่าง x และ y คือตำแหน่งมุมบนซ้ายของหน้าต่าง (อ้างอิงกับหน้าต่างของเน็ตสเคป) หน่วยเป็นพิกเซล width และ height คือความกว้างและความสูงของหน้าต่างหน่วยเป็นพิกเซล ส่วน clipRect ไม่ได้ใช้งานในไมโครซอฟท์วินโดส์ ถ้าพารามิเตอร์

window หรือ window ในโครงสร้างข้อมูล NPWindow มีค่าเป็น NULL เน็ตสเคปปลั๊กอินจะต้องไม่แสดงผลใดๆในหน้าต่างนี้และต้องทำลายทุกอย่างที่เกี่ยวข้องกับหน้าต่างนี้

พารามิเตอร์: instance = ตัวชี้ไปยังอินสแตนที่ได้รับหน้าต่างนี้

window = ตัวชี้ไปยังโครงสร้าง NPWindow

ค่าส่งกลับ: NPERR_NO_ERROR: แสดงว่าไม่มีข้อผิดพลาดเกิดขึ้น

NPERR_GENERIC_ERROR: แสดงว่ามีข้อผิดพลาดเกิดขึ้น

NPERR_OUT_OF_MEMORY_ERROR: แสดงว่าหน่วยความจำไม่พอ

NPERR_INVALID_INSTANCE_ERROR: แสดงว่าค่าของ instance ไม่ถูกต้องเช่นเป็น NULL

---

void NPP_Print(NPP instance, NPPrint *platformPrint)

จุดประสงค์: ใช้สำหรับพิมพ์ข้อมูลที่เน็ตสเคปปลั๊กอินต้องการออกทางเครื่องพิมพ์ ฟังก์ชันนี้จะถูกเรียกเมื่อผู้ใช้สั่งพิมพ์ผ่านทางเน็ตสเคปโดยเน็ตสเคปจะส่งข้อมูลที่ใช้ในการพิมพ์มาให้ในโครงสร้าง NPPrint ซึ่งมีโครงสร้างดังนี้

```
typedef struct _NPFullPrint
{
        NPBool          pluginPrinted;
        NPBool          printOne;
        void            *platformPrint;
} NPFullPrint;
typedef struct _NPEmbedPrint
{
        NPWindow        window;
        void            *platformPrint;
} NPEmbedPrint;
typedef struct _NPPrint
{
        uint16          mode;
        union
        {
                NPFullPrint     fullPrint;
                NPEmbedPrint    embedPrint;
```

```
        } print;
    } NPPrint;
```

mode บอกถึงรูปแบบการพิมพ์ที่ต้องการมีค่าได้ดังนี้ NP_EMBED (พิมพ์เป็นส่วนหนึ่งของเอกสาร) ในกรณีนี้ค่าใน print.embedPrint.window จะบอกตำแหน่งและขอบเขตของการพิมพ์มีหน่วยเป็น 1/1440 นิ้ว ค่าใน print.embedPrint.platformPrint จะเก็บดีไวซ์คอนเท็ก (device context) ที่ใช้ในการพิมพ์

ในกรณีที่ผู้ใช้สั่งพิมพ์เมื่อใช้งานเน็ตสเคปปลั๊กอินแบบเต็มหน้าอยู่ ค่าของพารามิเตอร์ platformPrint จะเป็น NULL

พารามิเตอร์: instance = ตัวชี้ไปยังอินสแตนที่จะทำการพิมพ์
platformPrint = ตัวชี้ไปยังโครงสร้างข้อมูล NPPrint

ค่าส่งกลับ: ไม่มี

---

void NPN_Status(NPP instance, const char *message)

จุดประสงค์: ใช้สำหรับเขียนข้อความลงบนพื้นที่แสดงสถานภาพของเน็ตสเคป

พารามิเตอร์: instance = ตัวชี้ไปยังอินสแตนที่ต้องการเขียนข้อความ
message = ตัวชี้ไปยังสายอักขระที่ใช้เป็นข้อความ

ค่าส่งกลับ: ไม่มี

---

const char *NPN_UserAgent(NPP instance)

จุดประสงค์: ใช้สำหรับสอบถามค่าของเขตข้อมูล (field) user agent ของเน็ตสเคป user agent เป็นเขตข้อมูลหนึ่งในส่วนหัว (header) ของโปรโตคอลเอชทีทีพีซึ่งใช้ระบุถึงโปรแกรมเว็บเบราเซอร์ที่ทำการส่งข้อมูล

พารามิเตอร์: instance = ตัวชี้ไปยังอินสแตนที่ต้องการสอบถาม

ค่าส่งกลับ: ตัวชี้ไปยังสายอักขระที่เป็นค่าของเขตข้อมูล user agent

---

void NPN_Version(int *plugin_major, int *plugin_minor, int *netscape_major, int *netscape_minor)

จุดประสงค์: ใช้สำหรับสอบถามรุ่นของเอพีไอที่อยู่ในเน็ตสเคปและในเน็ตสเคปปลั๊กอิน ค่าของรุ่นนี้จะใช้สำหรับตรวจสอบความเทียบเคียงกันได้ (compatible) ของเน็ตสเคปและเน็ตสเคปปลั๊กอิน เพื่อหาว่ามีคุณสมบัติไหนที่ใช้ได้บ้าง

พารามิเตอร์: plugin_major = ตัวชี้ไปยังตัวแปรที่จะเก็บส่วนเลขรุ่นหลัก (major) ของเอพีไอที่อยู่ในเน็ตสเคปปลั๊กอิน

plugin_minor = ตัวชี้ไปยังตัวแปรที่จะเก็บส่วนเลขรุ่นรอง (minor) ของเอพีไอที่อยู่ในเน็ตสเคปปลั๊กอิน

netscape_major = ตัวชี้ไปยังตัวแปรที่จะเก็บส่วนเลขรุ่นหลักของเอพีไอที่อยู่ในเน็ตสเคป

netscape_minor = ตัวชี้ไปยังตัวแปรที่จะเก็บส่วนเลขรุ่นรองของเอพีไอที่อยู่ในเน็ตสเคป

ค่าส่งกลับ: ไม่มี

ขั้นตอนการพัฒนาเน็ตสเคปปลั๊กอิน

1. จัดหาชุดพัฒนาเน็ตสเคปปลั๊กอินรุ่น 2.0 ชุดพัฒนานี้จะถูกอัดแน่น (compress) อยู่ในแฟ้มข้อมูลชื่อ NSPI20.ZIP

2. สร้างสารบบย่อย (sub directory) ขึ้นมาหนึ่งสารบบ

3. ทำการขยายข้อมูลที่ถูกอัดแน่นไว้ในแฟ้มข้อมูล NSPI20.ZIP ลงไปที่สารบบย่อยที่ได้จากขั้นตอนที่ 2 ภายหลังจากการขยายข้อมูลแล้วจะมีสารบบย่อยชื่อ NPAVI และ NPSHELL เกิดขึ้นภายใต้สารบบย่อยที่ได้จากขั้นตอนที่ 2 สารบบย่อย NPAVI จะเก็บตัวอย่างการพัฒนาเน็ตสเคปปลั๊กอินที่ใช้ในการแสดงผลภาพยนตร์ที่จัดเก็บในรูปแบบเอวีไอ ส่วนสารบบย่อย NPSHELL จะเก็บแฟ้มข้อมูลต่างๆที่จำเป็นสำหรับการพัฒนาเน็ตสเคปปลั๊กอิน ผู้พัฒนาจะต้องทำการแก้ไขแฟ้มข้อมูลบางแฟ้มในสารบบย่อยนี้และอาจเพิ่มเติมแฟ้มข้อมูลอื่นๆที่จำเป็นลงไปได้ ตารางที่ 3- 3 แสดงแฟ้มข้อมูลต่างๆที่ปรากฏอยู่ในสารบบย่อย NPSHELL และหน้าที่ของแต่ละแฟ้มข้อมูล

แฟ้มข้อมูล	หน้าที่
NPAPI.H	เป็นที่ประกาศสัญลักษณ์ (symbol) โครงสร้างข้อมูล และโปรโตไทป์ (prototype) ของฟังก์ชันต่างๆ ที่จำเป็นสำหรับการพัฒนาเน็ตสเคปปลั๊กอิน
NPUPP.H	เป็นที่ประกาศแมคโคร (macro) ที่ใช้ในการเรียกฟังก์ชันต่างๆ
NPSHELL.CPP	เป็นที่นิยามฟังก์ชันที่อยู่ในเอพีไอที่เน็ตสเคปปลั๊กอินเป็นผู้จัดเตรียมไว้ (นั่นคือฟังก์ชันที่ขึ้นต้นด้วย NPP)
NPWIN.CPP	เป็นที่นิยามฟังก์ชันที่อยู่ในเอพีไอที่เน็ตสเคปเป็นผู้จัดเตรียมไว้ (นั่นคือฟังก์ชันที่ขึ้นต้นด้วย NPN) และฟังก์ชันต่างๆที่ถูกใช้งานโดยเน็ตสเคปเอง
NPDLL16.DEF	เป็นที่ประกาศฟังก์ชันในเน็ตสเคปปลั๊กอินขนาด 16 บิต ที่เน็ตสเคปสามารถเรียกใช้งานได้
NPDLL16.RC	เป็นที่นิยามทรัพยากรต่างๆที่ใช้โดยเน็ตสเคปปลั๊กอินขนาด 16 บิต
NPDLL16.RC2	เป็นที่นิยามชนิดข้อมูลที่เน็ตสเคปปลั๊กอินขนาด 16 บิตจะจัดการ
NPWIN16.MAK	เป็นแฟ้มข้อมูลของไมโครซอฟท์วิชวลซีพลัสพลัส (Microsoft Visual C++) รุ่น 1.5 ที่ใช้ในการสร้างเน็ตสเคปปลั๊กอินขนาด 16 บิต
NPDLL32.DEF	เป็นที่ประกาศฟังก์ชันในเน็ตสเคปปลั๊กอินขนาด 32 บิต ที่เน็ตสเคปสามารถเรียกใช้งานได้
NPDLL32.RC	เป็นที่นิยามทรัพยากรต่างๆที่ใช้โดยเน็ตสเคปปลั๊กอินขนาด 32 บิต
NPDLL32.RC2	เป็นที่นิยามชนิดข้อมูลที่เน็ตสเคปปลั๊กอินขนาด 32 บิตจะจัดการ
NPWIN32.MAK	เป็นแฟ้มข้อมูลของไมโครซอฟท์วิชวลซีพลัสพลัสรุ่น 2.1 ที่ใช้ในการสร้างเน็ตสเคปปลั๊กอินขนาด 32 บิต

ตารางที่ 3- 3 รายชื่อแฟ้มข้อมูลในสารบบย่อย NPSHELL พร้อมหน้าที่

4. ทำการแก้ไขแฟ้มข้อมูล NPSHELL.CPP ภายในแฟ้มข้อมูลนี้จะเป็นที่นิยามฟังก์ชันที่อยู่ในเอพีไอที่เน็ตสเคปปลั๊กอินเป็นผู้จัดเตรียมไว้ (นั่นคือฟังก์ชันที่ขึ้นต้นด้วย NPP) โดยทางเน็ตสเคปจะสร้างโครงของฟังก์ชันไว้ให้แล้ว ผู้พัฒนาเพียงแต่ทำการแก้ไขโครงเหล่านี้เพื่อให้ทำงานตามหน้าที่ที่กำหนดไว้

5. ทำการแก้ไขแฟ้มข้อมูล NPDLL16.RC2 ภายในแฟ้มข้อมูลนี้จะเป็นที่นิยามชนิดข้อมูลที่เน็ตสเคปปลั๊กอินจะจัดการส่วนที่ต้องแก้คือบรรทัดต่อไปนี้

```
VALUE "MIMEType",          "fake/x-mime-type\0"
VALUE "FileExtents",       "fak\0"
```

VALUE "FileOpenName", "Fake MIME (*.fak)\0"

แก้ fake/x-mime-type เป็นชนิดข้อมูลที่ต้องการ แก้ fak เป็นนามสกุลของแฟ้มข้อมูลที่เก็บข้อมูลชนิดที่ต้องการ แก้ Fake MIME (*.fak) เป็นข้อความที่ต้องการให้ปรากฏในกล่องโต้ตอบ (dialog box) ที่ใช้ในการเปิดแฟ้มข้อมูล

6. สร้างแฟ้มข้อมูลอื่นๆที่จำเป็นในการพัฒนาขึ้นมาแล้วรวมเข้าเป็นส่วนหนึ่งของโครงการ (project)

7. ใช้ตัวแปลโปรแกรมภาษาซีทำการสร้างเน็ตสเคปปลั๊กอินขึ้นมาจาก NPWIN16.MAK อนึ่งชื่อคลังโปรแกรมเชื่อมโยงแบบพลวัตที่ได้จะต้องขึ้นต้นด้วย NP ไม่เช่นนั้นเน็ตสเคปจะไม่ถือว่าคลังโปรแกรมนี้คือเน็ตสเคปปลั๊กอิน

8 . ทำการสำเนาคลังโปรแกรมเชื่อมโยงแบบพลวัตที่ได้ไปไว้ในสารบบย่อย PLUGINS ที่อยู่ภายใต้สารบบของเน็ตสเคป