IBM Spectrum Protect
8.1.12

使用應用程式設計介面

附註：

在使用本資訊及其支援的產品之前，請先閱讀第 181 頁的『注意事項』中的資訊。

此版本適用於 8.1.12 版的 IBM Spectrum® Protect（產品編號 5725-W98、5725-W99、5725-X15）及後續所有的版次和修訂，除非新版中另有指示。

目錄

關於此出版品

此出版品提供的資訊可協助您執行下列作業：

- 新增 IBM Spectrum Protect 應用程式介面呼叫到現有的應用程式
- 撰寫具備通用程式介面的程式可取得 IBM Spectrum Protect 的服務。

除了應用程式設計介面 (API) 之外，下列程式也包含在幾個作業系統中：

- 備份保存用戶端程式可以從您的工作站或從檔案伺服器 到儲存體備份和保存檔案，以及還原和擷取本端檔案系統的備份版本和保存副本的檔案。
- Web 備份保存用戶端可讓授權的管理者、支援人員或一般使用者使用遠端機器上的 Web 瀏覽器 來執行備份、還原、保存及擷取等服務程式。
- 管理用戶端程式可讓您從 Web 瀏覽器或指令行來存取。 管理者可以控制及監督伺服器活動、 為備份、保存及空間管理服務程式定義儲存體管理原則，以及設定排程定期執行這些服務程式。

本出版品適用對象

此出版品提供有關如何將 API 呼叫新增至現有應用程式的指示。您應該熟悉 C 程式設計語言和 IBM Spectrum Protect 函數。

出版品

IBM Spectrum Protect 系列產品包括 IBM Spectrum Protect Plus、IBM Spectrum Protect for Virtual Environments、IBM Spectrum Protect for Databases 及 IBM® 提供的數個其他儲存體管理產品。

若要檢視 IBM 產品說明文件，請參閱 IBM Knowledge Center。

本書使用的慣例

此出版品使用下列慣例：

範例	說明
autoexec.ncf hsmgui.exe	一系列小寫字母和副檔名指示程式檔案名稱。
DSMI_DIR	一系列大寫字母指示回覆碼和其他值。
dsmQuerySessInfo	粗體字類型指示在指令行上輸入的指令、函數呼叫的名稱、結構的名稱、結構中的欄位或參數。
timeformat	斜體的粗體字類型指示備份保存用戶端選項。粗體類型用於建立選項，或者在範例中使用。
dateformat	斜體類型,指示選項、選項的值、新項目、您要提供資訊的位置保留元或文字中的特殊強調。
maxcmdretries	等寬類型指示程式的片段或可能出現在顯示畫面上的資訊（例如指令範例）。
加號 (+)	兩個按鍵之間的加號指示同時按下兩個按鍵。

8.1.12 版的新增功能

IBM Spectrum Protect 8.1.12 版引入了新增特性和更新項目。

如需此版次中的新增特性及更新項目清單，請參閱 API 更新項目。

本產品說明文件中任何新的和已變更的資訊由變更左側的垂直線 (|) 指出。

第 1 章 API 概觀

IBM Spectrum Protect 應用程式介面 (API) 可讓應用程式用戶端使用儲存體管理函數。

API 包含 許多函數呼叫，可讓您在應用程式中執行下列作業：

- 啟動或結束階段作業
- 將物件儲存在伺服器上之前，先指派管理類別給物件
- 備份或保存物件到伺服器
- 從伺服器還原或擷取物件
- 查詢伺服器有關儲存物件的資訊
- 管理檔案空間
- 傳送保留事件

身為應用程式開發者，在安裝 API 時，您會收到應用程式的一般使用者所需要的檔案：

- API 共用程式庫。
- 訊息檔案。
- 用戶端選項檔範例。
- 您的應用程式所需的 API 標頭檔的原始程式。
- 範例應用程式的原始程式以及用來建置它的 make 檔。

如果 是 64 位元應用程式，則應使用啟用 64 位元支援的編譯器選項，來執行所有的編譯動作。 例如，在 AIX® 上建置 API 應用程式時應使用 '-q64'，而在 Linux® 上則應使用 '-m64'。 如需相關資訊，請參閱 make 檔範例。

重要: 當您安裝 API 時，請確定所有的檔案都是相同的層級。

如需安裝 API 的相關資訊，請參閱安裝 IBM Spectrum Protect 備份保存檔用戶端。

對 UNIX 及 Linux 的參照包括 AIX、HP-UX、Linux、Mac OS X 及 Oracle Solaris 作業系統。

瞭解配置檔和選項檔

配置檔和選項檔會設定執行階段作業的條件和界限。

您（管理者）或一般使用者可以將選項值設定為：

- 設定對伺服器的連接
- 控制傳送到伺服器的物件以及它們相關的管理類別

當您安裝 API 到您的工作站上時，請在一或兩個檔案中定義選項。

在 UNIX 和 Linux 作業系統，選項是位於二個選項檔中：

- dsm.opt - 用戶端選項檔
- dsm.sys - 用戶端系統選項檔

在其他的作業系統，用戶端選項檔 (dsm.opt) 包含所有的選項。

限制: API 不支援下列備份保存用戶端選項：

- `autofsrename`
- `changingretries`
- 網域
- `eventlogging`
- `groups`

- subdir
- 使用者
- virtualmountpoint

您也可以在 **dsmInitEx** 函數呼叫時指定選項。 請使用選項字串參數或 API 配置檔參數。

相同的選項可能衍生自一個以上的配置來源。 發生這種情況時，具有較高優先順序的來源優先。 第 2 頁的表 1 列出優先順序。

表 *1.* 按遞減優先順序的次序配置來源

優先順序	UNIX 和 Linux	Windows	說明
1	dsm.sys 檔 （用戶端系統選項）	不適用	這個檔案包含系統管理者僅針對 UNIX 和 Linux 所設定的選項。 **提示:** 如果您的 dsm.sys 檔案包含伺服器段落，請確保 passwordaccess 選項在每個段落中指定相同的值（prompt 或 generate）。
2	選項字串 （用戶端選項）	選項字串 （所有選項）	這其中的一個選項會在當作參數傳遞給 **dsmInitEx** 呼叫時生效。 清單可以包含用戶端選項，如 compressalways、servername（僅適用 UNIX 和 Linux）或 tcpserveraddr（非 UNIX）。 應用程式用戶端可以利 用 API 選項字串來變更 API 配置檔和用戶端選項檔中的選項值。 例如，您的應用程式 可能會詢問使用者是否需要壓縮。 根據使用者的回應，您可以利用這個選項來建構 API 選項字串，然後傳遞給 **dsmInitEx** 的呼叫。 如需 API 選項字串格式的相 關資訊，請參閱第 96 頁的『 dsmInitEx 』。 您也可以將此參數設定為 NULL。 這樣表示此階段作業無 API 選項字串。
3	API 配置檔 （用戶端選項）	API 配置檔 （所有選項）	您在 API 配置檔 中設定的值會置換在用戶端選項檔中所設定的值。請以適合使用者 IBM Spectrum Protect 階段作業的值，來設定 API 配置檔中的選項。當 API 配置檔名稱當作 **dsmInitEx** 呼叫中的參數來傳送時，這些值就會生效。 您也可以將此參數設定為 NULL。 這樣表示此階段作業無 API 配置檔。
4	dsm.opt 檔 (dsm.opt file) （用戶端選項）	dsm.opt 檔 (dsm.opt file) （所有選項）	在 UNIX 和 Linux 作業系統上，dsm.opt 檔僅包含使用者選項。 在其他的作業系統，dsm.opt 檔包含所有選項。 如果要置換這些檔案中的選項，請遵循本表所說明的方法。

相關概念
處理選項

設定 API 環境

API 會使用唯一的環境變數來尋找檔案。 您可以對 API 應用程式使用與備份保存用戶端不同的檔案。 應用程式可以使用 **dsmSetup** 函數呼叫來置換 環境變數所設定的值。

提示: 在 Windows 上，預設安裝目錄為：%SystemDrive%\Program Files\Common Files\Tivoli\TSM\api

第 3 頁的表 2 按作業系統列出 API 環境變數。

表 *2. API* 環境變數

變數	UNIX 和 Linux	Windows
DSMI_CONFIG	用戶端選項檔 (dsm.opt) 的完整名稱。	用戶端選項檔 (dsm.opt) 的完整名稱。
DSMI_DIR	指向包含 dsm.sys、en_US 子目錄以及任何其他國家語言支援 (NLS) 語言的路徑。en_US 子目錄必須包含 dsmclientV3.cat。	指向包含 dscenu.txt 以及任何 NLS 訊息檔的路徑。
DSMI_LOG	指向 dsierror.log 檔的路徑。	指向 dsierror.log 檔的路徑。 如果設定了用戶端 **errorlogname** 選項，則透過該選項指定的位置會置換透過 DSMI_LOG 指定的目錄。

第 2 章 建置和執行範例 API 應用程式

API 套件包含範例應用程式，以示範環境定義中的 API 函數呼叫。 請安裝此範例應用程式並檢閱原始碼，以瞭解如何使用函數呼叫。

請選取下列其中一個範例 API 應用程式套件：

- 交談式、單緒應用程式套件（ dapi*）
- 多執行緒應用程式套件 (callmt*)
- 邏輯物件分組測試應用程式 (dsmgrp*)
- 以事件為基礎的保留原則範例應用程式 (callevnt)
- 刪除保留範例應用程式 (callhold)
- 資料保留保護範例應用程式 (callret)
- IBM Spectrum Protect 資料緩衝區範例程式 (callbuff)

為協助您開始進行，請檢閱程序，依您的平台建置範例 dapismp 範例應用程式：

- 如果是 UNIX 或 Linux 應用程式，請參閱第 5 頁的『UNIX 或 Linux 範例應用程式原始檔』。
- 對於 Windows 應用程式，請參閱第 7 頁的『Windows 64 位元範例應用程式』。

在備份或保存物件時，dapismp 範例應用程式會建立自己的資料串流。 它不會讀取物件或將物件寫入本端磁碟檔案系統。 物件名稱不會對應到您工作站上的任何檔案。 您發出的 "seed string" 會產生一個型樣，以便在還原或擷取物件時加以驗證。 在您編譯範例應用程式並執行 **dapismp** 來啟動它之後，請遵循顯示於畫面上的指示。

UNIX 或 Linux 範例應用程式原始檔

如果要建置及執行範例 UNIX 或 Linux 範例應用程式，您必須確定您擁有某些原始檔。在建置範例應用程式之後，您就可以編譯及執行它。

第 5 頁的表 3 中所列出的檔案包含建置 API 套件中的範例應用程式時， 所需要的原始檔和其他檔案。

表 *3.* 建置 *UNIX* 或 *Linux API* 範例應用程式所需的檔案

檔名		說明
README_api_enu		README 檔
dsmrc.h dsmapitd.h dsmapips.h dsmapifp.h release.h		回覆碼標頭檔 通用類型定義標頭檔 作業系統特定的類型定義標頭檔 函數原型標頭檔 版次值標頭檔
dapibkup.c dapidata.h dapiinit.c dapint64.h dapint64.c dapipref.c dapiproc.c dapiproc.h	dapipw.c dapiqry.c dapirc.c dapismp.c dapitype.h dapiutil.h dapiutil.c	指令行範例應用程式的模組
makesmp[64].*xxx*		用來為您的作業系統建置 dapismp 的 make 檔。 *xxx* 是表示作業系統。

表 3. 建置 *UNIX* 或 *Linux API* 範例應用程式所需的檔案 *(*繼續*)*

檔名	說明
callmt1.c callmt2.c	多緒範例檔
callmtu1.c callmtu2.c	多執行緒 Unicode 範例檔
libApiDS.xx libApiDS64.xx 或 libApiTSM64.xx	共用媒體庫（字尾是根據平台而定）
dsmgrp.c callevnt.c callhold.c callret.c callbuff.c dpsthread.c	分組範例檔 以事件為基礎的保留原則範例原始程式 刪除保留範例原始程式 資料保留保護範例原始碼

建置 UNIX 或 Linux 範例應用程式

利用您作業系統適用的編譯器來建置 **dapismp** 範例 API 應用程式。

關於這項作業

您必須安裝下列編譯器，才能建置 UNIX 或 Linux API 範例應用程式：

- IBM AIX - IBM Visual Age 編譯器第 6 版或更新版本
- HP-IA64 - aCC 編譯器 A.05.50 或更新版本
- Linux - GCC 編譯器 3.3.3 版或更新版本
- Mac OS X - GCC 編譯器 4.0 版或更新版本
- Oracle Solaris - Oracle Studio C++ 編譯器第 11 版或更新版本

程序

1. 如果要建置 API 範例，請執行下列指令：

   ```
   gmake -f makesmp[64].xxx
   ```

   其中 *xxx* 是表示作業系統。
2. 建置範例後，請設定您的環境變數（包括 DSMI_DIR），以及選項檔。 如需相關資訊，請參閱第 1 頁的『瞭解配置檔和選項檔』。
3. 第一次登入時，請以 root 使用者登入以登錄您的密碼。

   提示: 將 compressalways 選項設為 no，可能不會重新傳送未經壓縮的物件。 此行為視應用程式功能而定。

   如果要在 AIX 上指定「共用記憶體」通訊方法，IBM Spectrum Protect API 用戶端使用者必須符合下列其中一種情況：

   - 必須以 root 使用者身分登入。
   - 必須具有與執行 IBM Spectrum Protect 伺服器之程序相同的 UID。

   如需相關資訊，請參閱應用程式說明文件。

4. 執行 **dapismp** 指令來啟動應用程式。
5. 從顯示的選項清單中選擇。 請確定您已執行登入動作，然後再執行任何其他的動作。

   需求: 在輸入檔案空間、高階和低階名稱時，請一律在這些名稱前面加上正確的路徑定界字元 (\)，例如：`\myfilespace`。 即使您指定星號 (*) 萬用字元，也必須使用這個字首。

相關概念

環境變數系統（UNIX 和 Linux 系統）

Windows 64 位元範例應用程式

如果要建置和執行 Microsoft Windows 64 位元系統的範例應用程式，您必須安裝 IBM Spectrum Protect API 並確定您擁有某些原始檔。

限制:

- 如果要取得最好的結果，請使用動態載入。 例如，請參閱 `dynaload.c` 檔以及範例程式碼中的實作。
- 範例應用程式的檔案位在下列目錄中：

  api64\obj
  包含 API 範例程式目的檔。

  api64\samprun
  包含範例程式 **dapismp**。 範例程式包含執行目錄。
- DLL tsmapi64.dll 為 64 位元 DLL。
- 請使用 Microsoft C/C++ 第 15 版編譯器和 makefile（即 `makesmp64.mak`），來編譯 API 範例應用程式 **dapismp**。 您可能需要調整 make 檔以符合您的環境（特別是程式庫或併入目錄）。
- 在編譯應用程式之後，請從 `api64\samprun` 目錄發出指令 **dapismp**，來執行範例應用程式。
- 從顯示的選項清單中選擇。 請確定您已執行登入動作，然後再執行任何其他的動作。
- 在輸入檔案空間、高階和低階名稱時，請一律在這些名稱前面加上正確的路徑定界字元 (\)，例如：`\myfilespace`。 即使您指定星號 (*) 萬用字元，也必須使用這個字首。

在 Windows 作業系統，第 7 頁的表 4 列出了建置範例應用程式所必須具備的原始檔。 範例應用程式包含在 API 套件中。 為方便您使用，另含括了經過前置編譯的執行檔 (dapismp.exe)。

表 *4.* 用於建置 *Windows 64* 位元 *API* 範例應用程式的檔案

檔名	說明
`api.txt`	README 檔
`tsmapi64.dll`	API DLL
`dsmrc.h`	回覆碼標頭檔
`dsmapitd.h`	通用類型定義標頭檔
`dsmapips.h`	作業系統特定的類型定義標頭檔
`dsmapifp.h`	函數原型標頭檔
`dsmapidl.h`	以動態方式載入的函數原型標頭檔
`release.h`	版次值標頭檔

`dapidata.h`
`dapint64.h`
`dapitype.h`
`dapiutil.h`
	原始程式標頭檔

表 *4.* 用於建置 *Windows 64* 位元 *API* 範例應用程式的檔案 *(*繼續*)*

檔名	說明
tsmapi64.lib	隱含的媒體庫
dapibkup.c dapiinit.c dapint64.c dapipref.c dapiproc.c dapiproc.h dapipw.c dapiqry.c dapirc.c dapismp64.c dapiutil.c dynaload.c	dapismp.exe 的原始程式檔
makesmpx64.mak (Windows x64) makesmp64.mak (Windows IA64)	建置範例應用程式的 make 檔
callmt1.c callmt2.c callmtu164.c callmtu264.c	多執行緒範例檔
dpsthread.c	範例檔原始碼
callevnt.c callhold.c callret.c callbuff.c	以事件為基礎的保留原則原始碼 刪除保留範例原始程式 資料保留保護範例原始碼 共用的緩衝區（無副本）範例原始碼。

第 3 章 設計應用程式的注意事項

當您設計應用程式時，必須廣泛地瞭解 API 的許多方面。

如果要瞭解 API，請檢閱下列主題：

- 第 11 頁的『決定大小限制』
- 第 12 頁的『維護 API 版本控制』
- 第 13 頁的『使用多執行緒』
- 第 14 頁的『信號與信號處理程式』
- 第 14 頁的『啟動或結束階段作業』
- 第 18 頁的『物件名稱和 ID』
- 第 17 頁的『控制對密碼檔的存取權』
- 第 19 頁的『以階段作業擁有者的身分存取物件』
- 第 20 頁的『跨節點和跨擁有者存取物件』
- 第 20 頁的『管理檔案空間』
- 第 23 頁的『使物件與管理類別產生關聯』
- 第 24 頁的『到期/刪除保留和釋放』
- 第 26 頁的『查詢 IBM Spectrum Protect 系統』
- 第 30 頁的『傳送資料到伺服器』
- 第 46 頁的『備份和保存流程圖範例』
- 第 51 頁的『檔案分組』
- 第 60 頁的『IBM Spectrum Protect API 狀態圖摘要』

當您設計應用程式時，請檢閱第 9 頁的表 5 中的注意事項。 使用 **memset** 欄位的開始結構，在後續版本中可能會改變。 `stVersion` 值會隨著每一個產品加強功能而增量。

表 5. 設計應用程式的 *API* 注意事項

設計項目	注意事項
設定語言環境	呼叫 API 之前，應用程式必須設定語言環境。 如果要將語言環境設為預設值，請新增下列程式碼到應用程式： `setlocale(LC_ALL,"");` 如果要將語言環境設為另一個值，請使用相同的呼叫，並在第二個參數中使用適當的語言環境。 請查看您使用的各作業系統之說明文件中的細節。

表 5. 設計應用程式的 *API* 注意事項 (繼續)

設計項目	注意事項
階段作業控制	套用下列準則到階段作業控制： · 指派唯一的節點名稱給每一個 IBM Spectrum Protect 備份保存用戶端和 IBM Spectrum Protect API 用戶端產品。下列產品是這些用戶端的範例： – 適用於郵件的 IBM Spectrum Protect – 或 IBM Spectrum Protect HSM for Windows · 在備份及還原程序中，請使用一致的擁有者名稱。 · 使用 passwordaccess 選項來管理受保護密碼檔的存取。 · 請確定在作業完成時，資料移動的階段作業結束，讓伺服器上的裝置可以被釋放供其他階段作業使用。 · 如果要允許不需 LAN 資料傳送，請使用 **dsmSetup** 函數呼叫並設定開啟 multithread 旗標。 · 在 AIX 上，當您使用多執行緒應用程式或不需 LAN 時（尤其是在配備多個處理器的機器上執行時），請在啟動應用程式前，先於環境中將環境變數 AIXTHREAD_SCOPE 設為 S，以便獲得較佳的效能和較連續而完整的排程。 例如： `EXPORT AIXTHREAD_SCOPE=S` 將 AIXTHREAD_SCOPE 設為 S，以預設屬性所建立的使用者執行緒，會被置於全系統的競爭範圍內。 使用者執行緒連結到核心執行緒並由該核心執行緒排程。基礎核心執行緒不與其他任何使用者執行緒共用。 如需相關資訊，請參閱第 13 頁的『使用多執行緒』。 · 請確定階段作業中隨時都只有一個執行緒呼叫任何的 API 函數。 使用具有相同階段作業控點的多個執行緒的應用程式，必須同步處理 API 呼叫。 例如，使用 **mutex** 來同步處理 API 呼叫： getTSMMutex() issue TSM API call releaseTSMMutex() 只有執行緒共用 handle 時才使用這種方式。 如果各個呼叫都有不同的階段作業控點，您可以平行呼叫 API 函數。
階段作業控制項（續）	· 實作資料移動的執行緒消費者/生產者模型。 API 呼叫為同步處理，並且 **dsmGetData 函數**和 **dsmSendData 函數**的呼叫會封鎖到完成為止。 藉由使用消費者/生產者模型，應用程式可以在等待網路的期間，讀取下一個緩衝區。 當發生網路瓶頸或延遲情況時，這也可以藉由減少資料讀寫與網路的依存性來提升效能。 一般而言，下列情況成立： `Data thread <---> shared queue of buffers <---> communication thread (issue calls to the IBM Spectrum Protect API)` · 對多個作業使用相同的階段作業，以免造成額外負擔。 對於處理許多小物件的應用程式，請實作階段作業儲存區，讓多個小作業之間可以使用相同的階段作業。 額外的負擔是與開啟和關閉 IBM Spectrum Protect 伺服器的階段作業有關。 即使是在多執行緒應用程式中，dsmInit/dsmInitEX 呼叫一樣會被序列化，隨時都只有一個執行緒可以登入。 在登入期間，API 也會對伺服器傳送一些一次性查詢，讓伺服器可以執行所有作業。 這些查詢包含原則、選項、檔案空間和本端配置。

表 5. 設計應用程式的 *API* 注意事項 (繼續)

設計項目	注意事項
作業順序	在某些作業期間，IBM Spectrum Protect 伺服器會鎖定檔案空間資料庫登錄。 以下是設計 IBM Spectrum Protect API 應用程式時所適用的規則： · 查詢會在整個異動期間鎖定檔案空間。 · 查詢鎖定可與其他查詢作業共用，所以相同檔案空間上的多個查詢作業便可同時執行。 · 下列作業用來修改 IBM Spectrum Protect 伺服器資料庫 (**DB Chg**)：傳送、取得、更名、更新和刪除。 · 資料庫變更期間，當交易結束時，需要鎖定檔案空間，以完成 **DB Chg** 作業。 · 相同檔案空間上的多個 **DB Chg** 作業可同時執行。 序列在等待結束交易時鎖定的過程中，可能會有延遲現象發生。 · 查詢鎖定無法與 **DB Chg** 作業共用。 在相同檔案空間上，**DB Chg** 作業會造成查詢延後開始執行，所以應用程式的設計應該將要在相同檔案空間上執行的查詢與 **DB Chg** 作業分開執行並使查詢序列化。
物件命名	當您命名物件時，請考量下列因素： · 特定的物件名稱是高階和低階物件名稱。 如果名稱中有專用的 ID（例如日期戳記），則備份物件一律為作用中。 只有在使用 **dsmDeleteObj** 函數呼叫，將其標示為非作用中時，物件才會到期。 · 物件的還原方法決定名稱的格式化方式，以便能輕鬆查詢。 如果您要使用部分物件 還原 (POR)，請勿使用壓縮。 如果要停用壓縮，請使用 **dsmSendObj objAttr objCompressed=bTrue** 函數。
物件分組	利用檔案空間對物件進行邏輯分組。 檔案空間是伺服器上的儲存器，提供物件的分組種類。 在起始登入期間以及查詢期間，此 API 會查詢所有的檔案空間，所以必須限制檔案空間的數目。 合理的假設是應用程式會為每個節點設定 20 - 100 個檔案空間。 此 API 可以提供更多的檔案空間，但是每一個檔案空間都會造成階段作業的額外負擔。 如果要建立更細的分隔，請在應用程式中使用 `directory` 物件。
物件處理	請不要儲存 `objectID` 值以供日後還原使用。 在物件生命週期中，這些值並不一定會持續存在。 在還原期間，請特別注意還原順序。 在查詢之後，請先排序此值，然後再進行還原。 如果您是使用多種類型的序列媒體，請在個別的階段作業中存取不同類型的媒體。 如需詳細資訊，請參閱下列主題： 第 54 頁的『依照還原順序來選取及排序物件』
管理類別	考慮對於與應用程式物件相關的管理類別，應用程式需要有何種程度的控制。 您可以定義 include 陳述式，或在 **dsmSendObj** 函數呼叫中指定名稱。
物件大小	IBM Spectrum Protect 需要了解每個物件的大小預估值。請考量應用程式如何估計物件的大小。 高估了物件大小會比低估好。

決定大小限制

某些 API 中的資料結構或欄位有大小限制。 這些結構通常是名稱，或其他 無法超過預定長度的文字欄位。

下列欄位是有大小限制的資料結構範例：

· 應用程式類型
· 保存說明
· 副本群組目的地
· 副本群組名稱

· 檔案空間資訊
· 管理類別名稱
· 物件擁有者名稱
· Password

這些限制是定義為標頭檔 dsmapitd.h 中的常數。任何儲存體配置是根據這些常數，而不是您所輸入的數目。 如需相關資訊，請參閱第 133 頁的『附錄 B API 類型定義原始檔』。

維護 API 版本控制

所有的 API 都有某種形式的版本控制。您在應用程式中所使用的 API 版本，必須與安裝在使用者工作站上的 API 程式庫版本相容。

dsmQueryApiVersionEx 應為使用 API 時，您所輸入的第一個 API 呼叫。 這個呼叫會執行下列作業：

· 確認已安裝了 API 程式庫，而且可在一般使用者的系統上使用
· 傳回應用程式所存取的 API 程式庫版本層次

API 的設計是可以向上相容的。 當您執行較新的版本時，針對舊版本或版次的 API 程式庫所撰寫的應用程式都可以正常作業。

因為部份版次可能有不同的記憶體需求和資料結構定義，決定 API 程式庫的版次是非常重要的。 舊版相容性則不然。 請參閱第 12 頁的表 6 以取得關於平台的資訊。

表 6. 平台相容性資訊

平台	說明
Windows	訊息檔所使用的層次必須與程式庫 (DLL) 的相同。
UNIX 或 Linux	API 程式庫及訊息檔必須是相同的層級。

dsmQueryApiVersionEx 呼叫會傳回安裝在一般使用者工作站上所安裝的 API 程式庫版本。 然後，您可以比較傳回值，以及應用程式用戶端所使用的 API 版本。

應用程式用戶端的 API 版本號碼是以 dsmapitd.h 中所定義四個一組的常數輸入到編譯過的目的碼中:

```
DSM_API_VERSION
DSM_API_RELEASE
DSM_API_LEVEL
DSM_API_SUB_LEVEL
```

請參閱第 133 頁的『附錄 B API 類型定義原始檔』。

應用程式用戶端的 API 版本應小於或等於使用者系統中所安裝的 API 程式庫。 請注意任何其他的條件。 不論是否啟動了 API 階段作業，您都可隨時輸入 **dsmQueryApiVersionEx** 呼叫。

API 所使用的資料結構也有本身的版本控制資訊。 結構的第一個欄位中會有版本資訊。 隨著結構功能的加強，版本號碼也隨之增加。 在起始設定版本欄位時， 請使用 dsmapitd.h 中所定義的結構 Version 值。

第 13 頁的圖 1 示範了 dsmapitd.h 標頭檔中， **dsmApiVersionEx** 結構的類型定義。範例中還定義了 名稱為 ***apiLibVer*** 的廣域變數。 它也示範了您要如何在呼叫 **dsmQueryApiVersionEx** 時使用它，以便傳回一般使用者的 API 程式庫版本。 最後，傳回值會與應用程式用戶端的 API 版本號碼作比較。

```
typedef struct
{
        dsUint16_t     stVersion;           /* 結構版本                */
        dsUint16_t version;       /* API 版本                        */
        dsUint16_t release;       /* API 版次                        */
        dsUint16_t level;         /* API 層級                          */
        dsUint16_t subLevel;      /* API 子層                      */
} dsmApiVersionEx;

dsmApiVersionEx apiLibVer;

memset(&apiLibVer,0x00,sizeof(dsmApiVersionEx));
dsmQueryApiVersionEx(&apiLibVer);

/* 檢查相容性問題 */
dsInt16_t appVersion= 0, libVersion = 0;
  appVersion=(DSM_API_VERSION * 10000)+(DSM_API_RELEASE * 1000) +
               (DSM_API_LEVEL * 100) + (DSM_API_SUBLEVEL);
  libVersion = (apiLibVer.version * 10000) + (apiLibVer.release * 1000) +
                  (apiLibVer.level * 100) + (apiLibVer.subLevel);
   if (libVersion < appVersion)
   {
      printf("\n*********************************************************\n");
      printf("The IBM Spectrum Protect API library is lower than the application version\n");
      printf("Install the current library version.\n");
      printf("*********************************************************\n");
      return 0;
   }

printf("* API Library Version = %d.%d.%d.%d  *\n",
     apiLibVer.version,
     apiLibVer.release,
     apiLibVer.level,
     apiLibVer.subLevel);
```

圖 *1.* 取得 *API* 的版本層次範例

使用多執行緒

多執行緒 API 允許應用程式在同一個程序內，建立多個 IBM Spectrum Protect 伺服器階段作業。 您可以再次輸入該 API。 您可以在不同的緒中同時執行任何呼叫。

提示: 當您要執行使用多執行緒 API 的應用程式時，請使用 **dsmQueryAPIVersionEx** 呼叫。

如果要以多執行緒模式來執行 API，請在 **dsmSetUp** 呼叫中，將 *mtflag* 值設定為 DSM_MULTITHREAD。 **dsmSetUp** 呼叫必須是 **dsmQueryAPIVersionEx** 呼叫之後的第一個呼叫。 此呼叫必須在任何緒呼叫 **dsmInitEx** 呼叫前傳回。 當所有的緒完成處理，請輸入呼叫至 **dsmCleanUp**。 在所有的緒完成處理前，主要的程序應未結束。 請參閱範例應用程式中的 callmt1.c 。

限制：API 的預設值是單一執行緒模式。 如果應用程式未呼叫 **dsmSetUp** 而且將 *mtflag* 值設定為 DSM_MULTITHREAD，API 僅容許每個處理程序中有一個階段作業。

在順利完成 **dsmSetUp** 後，應用程式可以開始進入多緒，而且輸入 多個 **dsmInitEx** 呼叫。 每個 **dsmInitEx** 呼叫 會傳回該階段作業的 handle。 該緒中任何對於該階段作業的任何後續呼叫必須使用 handle 值。 部份值是適用於處理的環境變數（設定在 **dsmSetUp** 的值）。 每個 **dsmInitEx** 呼叫會再次剖析選項。 您可以在 **dsmInitEx** 呼叫中 指定改寫檔或選項字串，讓每個緒可以用不同的選項來執行。 這樣可以讓不同的緒前往不同的伺服器， 或使用不同的節點名稱。

建議： 在 HP 上，請將執行緒堆疊設定為 64K 或以上。 執行緒堆疊預設值 (32K) 可能不夠使用

如果要讓應用程式使用者能夠使用不需 LAN 的階段作業，請在您的應用程式中使用 **dsmSetUp** *mtFlag DSM_MULTITHREAD*。 即使應用程式是單一執行緒，您也必須這麼做。 這個旗標會啟動 IBM Spectrum Protect 不需 LAN 介面所需要的執行緒。

信號與信號處理程式

應用程式會處理來自使用者或作業系統的信號。 如果使用者輸入了 **CTRL+C** 按鍵序列，應用程式必須捕捉該信號，然後對每一個作用中的執行緒傳送 **dsmTerminate** 呼叫。 然後，呼叫 **dsmCleanUp** 以便跳出。 如果未適當地關閉階段作業，伺服器上可能會發生非預期的結果。

應用程式需要信號處理程式（例如 SIGPIPE 和 SIGUSR1），以處理會導致應用程式結束的信號。 然後應用程式會從 API 接收回覆碼。 例如，如果要忽略 SIGPIPE，請在應用程式中新增下列指令：`signal(SIGPIPE, SIG_IGN)`。 新增此資訊之後，應用程式不會在管道中斷時結束，而會傳回適當的回覆碼。

啟動或結束階段作業

IBM Spectrum Protect 是一項階段作業型產品，而且必須在 IBM Spectrum Protect 階段作業內執行所有活動。為了啟動階段作業，應用程式會啟動 **dsmInitEx** 呼叫。 您必須在 **dsmQueryApiVersionEx** 、**dsmQueryCliOptions** 或 **dsmSetUp** 以外的任何 API 呼叫前，執行此呼叫。

dsmQueryCliOptions 函數只可以在 **dsmInitEx** 呼叫之前進行呼叫。 函數會傳回選項檔、壓縮設定和通訊參數等重要選項的值。 **dsmInitEx** 呼叫會使用呼叫所傳送或選項檔中定義之參數指示的伺服器來設定階段作業。

用戶端節點名稱、擁有者名稱和密碼參數會傳送至 **dsmInitEx** 呼叫。 擁有者名稱會區分大小寫，但是節點名稱和密碼則不分。 應用程式用戶端節點必須在啟動階段作業之前登錄至伺服器。

每當 API 應用程式用戶端在伺服器上啟動階段作業時，用戶端應用程式類型會登錄至伺服器。 請一律指定作業系統縮寫作為應用程式類型值，因為此值會輸入到伺服器上的 platform 欄位中。 字串長度上限為 DSM_MAX_PLATFORM_LENGTH。

dsmInitEx 函數呼叫會使用應用程式用戶端的 API 配置檔和選項清單建立 IBM Spectrum Protect 階段作業。 應用程式用戶端可以使用 API 配置檔和選項清單來設定一些 IBM Spectrum Protect 選項。 這些值會置換安裝期間在使用者配置檔中所設定的值。 使用者無法變更管理者定義的選項。如果應用程式用戶端沒有特定的配置檔和選項清單，您可以將這些參數設定為空值。 如需配置檔的相關資訊，請參閱下列主題：

第 1 頁的『瞭解配置檔和選項檔』

dsmInitEx 函數呼叫會利用允許延伸驗證的參數來建立 IBM Spectrum Protect 階段作業。

檢查 **dsmInitEx** 函數呼叫和 **dsmInitExOut** 資訊回覆碼。 如果回覆碼正確 (RC=ok) 而且資訊回覆碼 (infoRC) 是 DSM_RC_REJECT_LASTSESS_CANCELED，管理者會取消前次階段作業。如果要立即結束現行階段作業，請呼叫 **dsmTerminate**。

dsmQuerySessOptions 呼叫會傳回與 **dsmQueryCliOptions** 呼叫相同的欄位。 只能在階段作業內傳送呼叫。 該值會反映該階段作業期間、來自選項檔和 **dsmInitEx** 呼叫所置換的有效用戶端選項。

在啟動階段作業之後，應用程式可將呼叫傳送至 **dsmQuerySessInfo** ，以判定為此階段作業設定的伺服器參數。 原則領域和交易限制等項目會傳回具有此呼叫的應用程式。

以 **dsmTerminate** 呼叫結束階段作業。 與伺服器的任何連線都會關閉，而且會釋放與此階段作業相關聯的所有資源。

如需啟動和結束階段作業的範例，請參閱下列主題：

第 16 頁的圖 2

範例定義了對 **dsmInitEx** 和 **dsmTerminate** 的呼叫中所使用的一些廣域和區域變數。

dsmInitEx 呼叫使用 dsmHandle 的指標作為參數。**dsmTerminate** 呼叫使用 dsmHandle 的指標作為參數。

第 16 頁的圖 3 中的範例會顯示 **rcApiOut** 的詳細資料。 **rcApiOut** 函數會呼叫 API 函數 **dsmRCMsg**，將回覆碼轉譯為訊息。

然後，**rcApiOut** 呼叫會為使用者列印此訊息。 **rcApiOut** 的版本是內含於 API 範例應用程式。 **dsmApiVersion** 函數是標頭檔 dsmapitd.h 中的類型定義。

階段作業安全

以階段作業為基礎的 IBM Spectrum Protect 系統所具有的安全性元件，可以容許應用程式以安全的方式來啟動階段作業。這些安全性措施可禁止對伺服器進行未經授權的存取，並且協助確保系統的完整性。

伺服器所啟動的每一個階段作業，都必須使用密碼完成登入程序。 在連接伺服器時，同時使用密碼和用戶端的節點名稱，可確保正確的授權。 應用程式用戶端可將此密碼提供至 API 以啟動階段作業。

您可使用兩種方法來處理密碼：*passwordaccess=prompt* 或 *passwordaccess=generate*。 如果您要使用 *passwordaccess=prompt* 選項，您必須在每一個 **dsmInitEx** 呼叫中併入密碼值。 或您可以將節點名稱和擁有者名稱提供給 **dsmInitEx** 呼叫。

密碼有與本身有關的期限時間。 如果 **dsmInitEx** 呼叫因為密碼到期 回覆碼 (DSM_RC_REJECT_VERIFIER_EXPIRED) 而失敗，應用程式用戶端 必須輸入 **dsmChangePW** 呼叫並使用 **dsmInitEx** 傳回的 handle。 這樣會在順利建立階段作業前更新密碼。 第 17 頁的圖 4 中的範例會示範使用 **dsmChangePW** 來變更密碼的程序。 登入擁有者必須使用 root 使用者 ID 或授權使用者 ID 來變更密碼。

第二種方法 *passwordaccess=generate* 會加密，並且將密碼值儲存在檔案中。 **dsmInitEx** 呼叫無法提供節點名稱和擁有者名稱，而會使用系統預設值。 這樣可保護密碼檔的安全性。 當密碼到期時，`generate` 參數會建立新的密碼，並且自動更新密碼檔。

秘訣:

1. 如果兩部實體機器採用相同 IBM Spectrum Protect 節點名稱，或使用數個伺服器段落在一個節點定義多個路徑，*passwordaccess=generate* 就只能用於在密碼到期之後最先使用的段落。 在最先用戶端-伺服器聯 絡期間，會要求使用者個別地為每一個伺服器段落輸入相同的密碼，而且對於每一個段段來說，會 個別儲存密碼複本。 當密碼到期時，會針對連接最先用戶端-伺服器聯絡的段落產生新密碼。 之後，透過其他伺服器段落來連接的所有後續嘗試都會失敗，因為它們的個別舊密碼副本之間沒有邏輯鏈結，而且在密碼到期之後會先使用此段落產生的更新副本。 在此情況下，您必須在密碼到期之前或之後更新密碼作為復原的措施，如下所示：
   a. 在伺服器執行 **dsmadmc** 並更新密碼。
   b. 執行 **dsmc -servername=stanza1** 並使用新密碼來產生適當的項目。
   c. 執行 **dsmc -servername=stanza2** 並使用新密碼來產生適當的項目。
2. 若為 UNIX 或 Linux：只有 root 使用者或授權使用者，可以在使用 *passwordaccess=prompt* 時變更密碼。只有 root 使用者或授權使用者，可以在使用 *passwordaccess=generate* 時啟動密碼檔。

   限制: 無法辨識 `users` 和 `groups` 選項。

應用程式可以利用其他方式限制使用者存取權，如設定存取過濾器。

使用多重 IP 連線來連接單一 IBM Spectrum Protect 伺服器的應用程式，應該對每個階段作業使用相同的節點名稱和 IBM Spectrum Protect 用戶端密碼。請遵循下列步驟來啟用這項支援：

1. 在 dsm.sys 檔中定義一個 IBM Spectrum Protect 伺服器段落.
2. 若連線不是使用預設 IP 位址，請在 **dsmInitEx** 呼叫上指定 *TCPserver* 位址 和 *TCPport* 。

這些值會置換 IP 連線資訊，但階段作業仍會使用相同 dsm.sys 段落的節點和密碼資訊。

註: 叢集中的節點共用單一密碼。

```
dsmApiVersionEx * apiApplVer;
char           *node;
char           *owner;
char           *pw;
char           *confFile = NULL;
char           *options = NULL;
dsInt16_t      rc = 0;
dsUint32_t     dsmHandle;
dsmInitExIn_t  initIn;
dsmInitExOut_t initOut;
char           *userName;
char           *userNamePswd;

memset(&initIn, 0x00, sizeof(dsmInitExIn_t));
memset(&initOut, 0x00, sizeof(dsmInitExOut_t));
memset(&apiApplVer,0x00,sizeof(dsmapiVersionEx));
apiApplVer.version = DSM_API_VERSION; /* 設定應用程式編譯 */
apiApplVer.release = DSM_API_RELEASE;  /* 時間版本。         */
apiApplVer.level   = DSM_API_LEVEL;
apiApplVer.subLevel= DSM_API_SUBLEVEL;

printf("Doing signon for node %s, owner %s, with password %s\n", node,owner,pw);

initIn.stVersion = dsmInitExInVersion;
initIn.dsmApiVersionP = &apiApplVer
initIn.clientNodeNameP = node;
initIn.clientOwnerNameP = owner ;
initIn.clientPasswordP = pw;
initIn.applicationTypeP = "Sample-API AIX";
initIn.configfile = confFile;
initIn.options = options;
initIn.userNameP = userName;
initIn.userPasswordP = userNamePswd;
rc = dsmInitEx(&dsmHandle, &initIn, &initOut);

if (rc == DSM_RC_REJECT_VERIFIER_EXPIRED)
{
   printf("*** Password expired. Select Change Password.\n");
   return(rc);
}
else if (rc)
{
   printf("*** Init failed: ");
   rcApiOut(dsmHandle, rc);   /* 呼叫函數至列印錯誤訊息 */
   dsmTerminate(dsmHandle);    /* 清除記憶體區塊 */
   return(rc);
}
```

圖 *2.* 啟動和結束階段作業範例

```
void rcApiOut (dsUint32_t handle, dsInt16_t rc)
{
     char *msgBuf ;

     if ((msgBuf = (char *)malloc(DSM_MAX_RC_MSG_LENGTH+1)) == NULL)
     {
          printf("Abort:  Not enough memory.\n") ;
          exit(1) ;
      }

      dsmRCMsg(handle, rc, msgBuf);
      printf("
      free(msgBuf) ;
      return;
}
```

圖 *3. rcApiOut* 的明細

```
printf("Enter your current password:");
gets(current_pw);
printf("Enter your new password:");
gets(new_pw1);
printf("Enter your new password again:");
gets(new_pw2);
/* 若新密碼項目不符合，請重試或結束。*/
/* 若不符合，請呼叫 dsmChangePW
。                          */

rc = dsmChangePW
(dsmHandle,current_pw,new_pw1);
if (rc)
{
   printf("*** Password change failed.  Rc =
}
else
{
   printf("*** Your new password has been accepted and updated.\n");
}
return 0;
```

圖 *4.* 變更密碼範例

控制對密碼檔的存取權

若要控制對 UNIX 和 Linux 系統 systems, you can log on as an authorized user and set the passwordaccess option to generate。

程序

將 passwordaccess 設定為 generate 時，請完成下列步驟：

1. 在撰寫應用程式時，加入呼叫會傳遞 *argv[0]* 的 **dsmSetUp**。 *argv[0]* 包含了呼叫 API 的應用程式名稱。 應用程式允許執行授權的使用者；但是，管理者必須決定授權使用者的登入名稱。
2. 將應用程式執行檔的有效使用者 ID 位元（S 位元）設為 On。 然後，應用程式可執行檔的擁有者會成為授權使用者，並可建立密碼檔、更新密碼和執行應用程式。應用程式可執行檔的擁有者必須與執行程式的使用者 ID 相同。在下列範例中，使用者 是 user1，應用程式可執行檔的名稱為 applA，而 user1 具有對 /home/user1 目錄的讀寫許可權。applA 執行檔具有下列許可權：

   ```
   -rwsr-xr-x user1    group1   applA
   ```

3. 指示應用程式的使用者使用授權使用者的名稱來登入。IBM Spectrum Protect 會在容許存取受保護的密碼檔前，驗證登入 ID 與應用程式可執行檔的擁有者是否相符。
4. 在 dsm.sys 檔案中設定 passworddir 以指向這個使用者具有讀寫存取權的目錄。
   例如，在 dsm.sys 檔案的伺服器段落中輸入下列行：

   ```
   passworddir /home/user1
   ```

5. 建立密碼檔，並且確定授權使用者擁有該檔案。
6. 以 user1 登入並執行 app1A。
7. 呼叫 **dsmSetUp** 並且傳遞至 *argv*。

具有用戶端擁有者權限的管理使用者

具有用戶端擁有者權限的管理使用者可以設定 **dsmInitEx** 函數呼叫的參數來啟動階段作業。 此使用者可作為"管理使用者"，具備對已定義的節點進行備份和還原的權限。

程序

如果要獲得用戶端擁有者的權限，請在伺服器完成下列步驟：

1. 定義管理使用者：

```
REGister Admin admin_name password
```

其中：

- *admin_name* 是管理使用者名稱。
- *password* 是管理者密碼。

2. 定義授權層次。 具有系統或原則權限的使用者也有用戶端擁有者的權限。

```
Grant Authority admin_name classes authority node
```

其中：

- *admin_name* 是管理使用者。
- *classes* 是節點。
- *authority* 是下列其中一個權限層次。
  - owner：節點的完整備份和還原權限
  - node：單一節點
  - domain：節點群組

3. 定義存取單一節點。

```
Register Node node_name password userid=user_id
```

其中：

- *node_name* 是用戶端使用者節點
- *password* 是用戶端使用者節點的密碼
- *user_id* 是管理使用者名稱

結果

當應用程式要使用這種類型的管理使用者時，會使用 >userName 和 userNamePswd 參數來呼叫 **dsmInitEx** 函數呼叫。

```
dsmInitEx
         clientNodeName = NULL
         clientOwnerName = NULL
      clientPassword = NULL
      userName = 'administrative user' name
      userNamePswd = 'administrative user' password
```

您可以將 passwordaccess 選項設定為 generate 或 prompt。 只要有其中任一參數， userNamePswd 值可以啟動階段作業。 當階段作業啟動時，該節點可執行任何備份或還原程序。

物件名稱和 ID

IBM Spectrum Protect 伺服器是一種物件儲存體伺服器，它的主要功能是有效地儲存和擷取指名物件。 物件 ID 專屬於每個物件，而除了 使用匯出或匯入以外，此物件 ID 在物件的生命週期中則屬於該物件。

為符合此需求，IBM Spectrum Protect 具有兩個主要儲存區，即資料庫和資料儲存體。

- 資料庫包含了所有 meta 資料，而其中包含了與物件結合的名稱或屬性。
- 資料儲存體包含物件資料。 資料儲存體實際上是系統管理者所定義的儲存體階層。 根據成本和存取的需要，線上或離線的媒體可有效率地儲存和管理資料。

儲存在伺服器上的每個物件會有相關聯的名稱。 用戶端會控制該名稱的下列主要元件：

- 檔案空間名稱
- 高階名稱
- 低階名稱

· 物件類型

在決定應用程式的物件命名時，您可能需要對一般使用者的完整物件名稱使用外部名稱。 特別是一般使用者可能需在執行應用程式時，在 Include 或 Exclude 陳述式中指定物件。 這些陳述式中的物件名稱正確語法會因為平台而有所不同。 在 Windows 作業系統上，與檔案空間（不是檔案空間名稱本身）相關聯的磁碟機代號會用於 Include 或 Exclude 陳述式。

建立物件時所指定的物件 ID 值可能不會和執行還原程序時相同。 應用程式應儲存物件名稱，然後查詢以便在執行還原前，取得現行物件 ID。

檔案空間名稱

檔案空間名稱是其中一個最重要的儲存體元件。 它可以是檔案系統、磁碟機的名稱，或將相關資料分組在一起的任何其他高階限定元。

IBM Spectrum Protect 使用檔案空間來識別資料所在的檔案系統或磁碟機。 透過此方式，您可以在檔案空間中對所有的項目，執行查詢指定檔案空間中的所有物件等動作。 因為檔案空間是 IBM Spectrum Protect 命名慣例的重要元件，您可以使用特別的呼叫以登錄、更新、查詢和刪除檔案空間。

伺服器也有管理指令來查詢 IBM Spectrum Protect 儲存體中任何節點的檔案空間，而且可在需要時刪除它們。 所有應用程式用戶端儲存的資料必須有一個與它相關聯的檔案空間名稱。 請小心選取這個名稱，以便將系統中類似的資料分組在一起。

若要避免被干擾，應用程式用戶端應該選取備份保存用戶端不會使用的檔案空間名稱。 應用程式用戶端應該發佈它的檔案空間名稱，讓使用者在需要時能夠識別 include-exclude 陳述式的物件。

註: 在 Windows 平台上，會有與檔案空間相關聯的磁碟機代號。 當您註冊或更新檔案空間時，您必須提供磁帶機字母。 因為併入/排除清單會參照磁碟機代號，您必須追蹤每個磁碟機代號，以及和其相關聯的檔案空間。 在範例程式 dapismp 中，磁碟機代號是預設為 "G"。

請參閱第 5 頁的『第 2 章 建置和執行範例 API 應用程式』以取得範例程式的相關資訊。

高階與低階名稱

物件名稱的兩個其他元件是高階名稱限定元和低階名稱限定元。 高階名稱限定元是物件所屬的目錄路徑，而低階名稱限定元是在該目錄路徑中的物件實際名稱。

當檔案空間名稱、高階名稱和低階名稱連接在一起時，在執行用戶端的作業系統上這 些名稱必須組成語法正確的名稱。 名稱不一定要像系統上的物件一樣存在，也不必與本端檔案系統上的實際資料類似。 但是，名稱必須符合標準的命名原則，以便讓 **dsmBindMC** 呼叫能正常地處理。 請參閱第 34 頁的『了解備份和保存物件』以便了解與原則管理相關的命名注意事項。

物件類型

物件類型會將物件視為檔案或目錄。 檔案是包含屬性和二進位資料的物件，而目錄是僅包含屬性的物件。

第 19 頁的表 7 依平台顯示應用程式用戶端為物件名稱編寫的程式碼。

表 7. 應用程式物件名稱範例（依平台）

平台	物件名稱的用戶端程式碼
UNIX 或 Linux	/myfs/highlev/lowlev
Windows	"myvol\\highlev\\lowlev" **註:** 在 Windows 平台上，雙反斜線會轉換成單反斜線，因為反斜線是跳出字元。 在 UNIX 或 Linux 平台上，檔案空間名稱以斜線為開頭；但在 Windows 平台上，請不要以斜線為開頭。

以階段作業擁有者的身分存取物件

每個物件都有相關聯的擁有者名稱。 決定存取何物件的規則是根據階段作業啟動時所使用的擁有者名稱。 使用此階段作業擁有者值來控制存取物件。

在呼叫 **dsmInitEx** 時的 *clientOwnerNameP* 參數會設定階段作業擁有者。 如果您以 **dsmInitEx** 擁有者名稱 *NULL* 來啟動階段作業，並且使用 *passwordaccess=prompt*，該階段作業擁有者是由階段作業（root 或授權使用者）權限所處理。如果您以 root 使用者 ID 或授權使用者 ID 來登入，而且使用 *passwordaccess= generate*，此狀況也成立。 以這種方式啟動的階段作業期間，您可以對這個節點所擁有的任何物件執行任何動作，不管該物件的實際擁有者為何。

如果階段作業是以特定的擁有者名稱啟動，階段作業只能對相關聯物件擁有者名稱的物件執行動作。 備份或保存至系統全都必須有相關聯的此擁有者名稱。 任何查詢只會傳回與此擁有者名稱相關聯的值。 物件擁有者值是在 **ObjAttr** 結構中 **Owner** 欄位的 **dsmSendObj** 呼叫期間所設定。 擁有者名稱區分大小寫。 第 20 頁的表 8 彙總了使用者可存取物件的條件。

表 *8.* 使用者存取物件摘要

階段作業擁有者	物件擁有者	使用者存取權
NULL（root、系統擁有者）	" "（空字串）	是
空值	特定名稱	是
特定名稱	" "（空字串）	否
特定名稱	相同名稱	是
特定名稱	不同的名稱	否

跨節點和跨擁有者存取物件

三個函數呼叫支援在相同平台上的跨節點、 跨擁有者存取：**dsmSetAccess**、**dsmDeleteAccess** 和 **dsmQueryAccess**。 這些函數和傳遞至 **dsmInitEx** 的 *-fromnode* 及 *-fromowner* 字串選項，可容許透過 API 完成跨節點查詢、還原和擷取程序。

例如，在節 點 A 的使用者 A 使用 **dsmSetAccess** 函數呼叫，將在 /db 檔案空間下面的他自己的備份之存取權給予來自節點 B 的使用者 B。存取規則會顯示成：

ID	類型	節點	使用者	路徑
1	備份	節點 B	使用者 B	/db/*/*

當使用者 B 登入至節點 B 時， **dsmInitEx** 的選項字串是：

```
-fromnode=nodeA -fromowner=userA
```

這些選項會設定至此階段作業。 任何查詢都會存取檔案空間以及節點 A 的檔案。 不允許備份和保存。 只允許從使用者 B 具有存取權的檔案空間執行查詢、還原和擷取程序。 如果在使用 *-fromnode* 或 *-fromowner* 選項集登入的情況下，應用程式嘗試使用 **dsmBeginTxn** 來執行任何作業（例如，備份或更新），則 **dsmBeginTxn** 會失敗，帶有回覆碼 DSM_RC_ABORT_NODE_NOT_AUTHORIZED。 請參閱單獨的函數呼叫和第 96 頁的『 dsmInitEx 』以便取得其他資訊。

提示: 在 UNIX 和 Linux，您可以在 **dsmInitEx** 函數呼叫傳送的選項字串中指定 *–fromowner=root*。 這樣可在執行了一組存取時，讓 root 以外的使用者來存取 root 所擁有的檔案。

使用 *asnodename* 選項（位於 **dsmInitEx** 選項字串上）並搭配適當的函數， 在 IBM Spectrum Protect 伺服器的目標節點名稱下執行資料的備份、保存、還原、擷取、查詢或刪除。 如需啟用此選項的相關資訊，請參閱第 66 頁的『透過用戶端節點 Proxy 支援來備份多個節點』。

管理檔案空間

因為檔案空間對於系統的作業來說非常重要，所以會使用個別的一組呼叫來登錄、更新和刪除檔案空間 ID。 在將與檔案空間相關聯的物件儲存在系統之前，您必須先 向 IBM Spectrum Protect 登錄檔案空間。

使用 **dsmRegisterFS** 呼叫來完成此作業。 如需物件名稱和 ID 的相關資訊，請參閱第 18 頁的『物件名稱和 ID』。

檔案空間 ID 是三部分名稱階層中的頂層限定元。 在檔案空間內集中相關資料可讓您更容易管理資料。 例如，應用程式用戶端或 IBM Spectrum Protect 伺服器管理者可以刪除檔 案空間以及該檔案空間內的所有物件。

檔案空間也可讓應用程式用戶端將關於檔案空間的資訊提供給管理者可以查詢的伺服器。**qryRespFSData** 結構中的查詢會收到此傳回的資訊，並且包含了下列檔案系統資訊：

類型	定義
fstype	檔案空間類型。 此欄位是應用程式用戶端所設定的字元字串。
fsAttr[platform].fsInfo	用於用戶端特定資料的用戶端資訊欄位。
容量	檔案空間中的空間總量。
佔用空間	檔案空間中目前已佔據的空間數量。
backStartDate	啟動最新備份（以傳送 **dsmUpdateFS** 呼叫所設定）時的時間戳記。
backCompleteDate	最新備份（以傳送 **dsmUpdateFS** 呼叫所設定）完成時的時間戳記。

使用 capacity 和 occupancy 會依應用程式用戶端而有所不同。 部分應用程式可能不需要關於檔案空間大小的資訊，在那種情況下，您可將這些欄位預設為 0。如需查詢檔案空間的相關資訊，請參閱第 26 頁的『查詢 IBM Spectrum Protect 系統』。

在將檔案空間登錄至系統後，您可以隨時備份或保存物件。 如果要在備份或保存作業之後更新檔案空間的 occupancy 和 capacity 欄位，請呼叫 **dsmUpdateFS** 。 這個呼叫可確定檔案系統的 occupancy 和 capacity 值是最新的。 您也可以更新 **fsinfo**、**backupstart** 和 **backupcomplete** 欄位。

如果您要監視最新的備份日期，請在啟動備份前輸入 **dsmUpdateFS** 呼叫。 將更新動作設定至 DSM_FSUPD_BACKSTARTDATE。 這可強制伺服器，將檔案空間的 **backStartDate** 欄位設定為目前的時間。 在完成該檔案空間的備份後，請輸入 **dsmUpdateFS** 呼叫，以及設定於 DSM_FSUPD_BACKCOMPLETEDATE 中的更新動作。 這個呼叫會在備份結束時建立時間戳記。

如果不再需要檔案空間，您可以使用 **dsmDeleteFS** 指令來刪除它。 在 UNIX 或 Linux 作業系統上，只有 root 使用者或授權使用者可以刪除檔案空間。

第 22 頁的圖 5 中的範例，示範如何針對 UNIX 或 Linux 使用這三個檔案空間呼叫。 如需如何使用 Windows 的三個檔案空間呼叫的範例，請參閱系統上安裝的範例程式碼。

```
/*  登錄檔案空間若之前尚未執行。 */

dsInt16               rc;
regFSData    fsData;
char         fsName[DSM_MAX_FSNAME_LENGTH];
char         smpAPI[] = "Sample-API";

strcpy(fsName,"/home/tallan/text");
memset(&fsData,0x00,sizeof(fsData));
fsData.stVersion = regFSDataVersion;
fsData.fsName = fsName;
fsData.fsType = smpAPI;
strcpy(fsData.fsAttr.unixFSAttr.fsInfo,"Sample API FS Info");
fsData.fsAttr.unixFSAttr.fsInfoLength =
       strlen(fsData.fsAttr.unixFSAttr.fsInfo) + 1;
fsData.occupancy.hi=0;
fsData.occupancy.lo=100;
fsData.capacity.hi=0;
fsData.capacity.lo=300;

rc = dsmRegisterFS(dsmHandle,fsData);
if (rc == DSM_RC_FS_ALREADY_REGED) rc = DSM_RC_OK;  /* 已完成 */
if (rc)
{
   printf("Filespace registration failed: ");
   rcApiOut(dsmHandle, rc);
   free(bkup_buff);
   return (RC_SESSION_FAILED);
}
```

圖 *5.* 使用檔案空間範例，第 *1* 部分

```
/* 更新檔案空間。 */

dsmFSUpd     updFilespace;         /* 適用於更新 FS */

updFilespace.stVersion = dsmFSUpdVersion;
updFilespace.fsType = 0;             /* 無變更 */
updFilespace.occupancy.hi = 0;
updFilespace.occupancy.lo = 50;
updFilespace.capacity.hi = 0;
updFilespace.capacity.lo = 200;
strcpy(updFilespace.fsAttr.unixFSAttr.fsInfo,
       "My update for filespace") ;
updFilespace.fsAttr.unixFSAttr.fsInfoLength =
       strlen(updFilespace.fsAttr.unixFSAttr.fsInfo);

updAction = DSM_FSUPD_FSINFO |
            DSM_FSUPD_OCCUPANCY |
            DSM_FSUPD_CAPACITY;

rc = dsmUpdateFS (handle,fsName,&updFilespace,updAction);
printf("dsmUpdateFS
 rc=%d\n", rc);
```

圖 *6.* 使用檔案空間範例，第 *2* 部分

```
/* 刪除檔案空間。 */

printf("\nDeleting file space
rc = dsmDeleteFS (dsmHandle,fsName,DSM_REPOS_ALL);
if (rc)
{
   printf("  FAILED!!! ");
   rcApiOut(dsmHandle, rc);
}
else printf("  OK!\n");
```

圖 *7.* 使用檔案空間範例，第 *3* 部分

使物件與管理類別產生關聯

IBM Spectrum Protect 的主要功能是使用原則（管理類別），來定義如何在 IBM Spectrum Protect 儲存體中儲存和管理物件。 在備份或保存物件時，物件會與管理類別相關聯。

此管理類別可決定：

· 在備份時要保存多少版本的物件

· 保存副本要保留多久

· 要將伺服器上儲存體階層中的物件插入何處

管理類別包含了備份副本群組和保存副本群組。 副本群組是屬性集，而用來定義備份或保存物件的管理原則。 如果執行備份作業，則適用於備份副本群組中的屬性。 如果 正在執行保存作業，則會套用保存副本群組中的屬性。

特定管理類別中的備份或保存副本群組可以是空白或 NULL。 如果物件是連結到 NULL 的備份副本群組，則該物件將無法備份。 如果物件是連結到 NULL 的保存副本群組，則該物件將無法保存。

因為原則的使用是 IBM Spectrum Protect 的一個很重要的元件，所以 API 要求傳給伺服器的 物件必須先使用 **dsmBindMC** 呼叫指定管理類別。 憑藉 IBM Spectrum Protect 軟體，您可以使用併入/排除清單來影響管理類別連結。 **dsmBindMC** 呼叫使用現行的「納入-排除」清單來執行管理類別連結。

Include 陳述式可使特定的管理類別與備份或保存物件相關聯。 Exclude 陳述式可使物件不被備份，而只被保存。

API 必須在備份或保存物件前呼叫 **dsmBindMC** 。 **dsmBindMC** 呼叫會傳回 mcBindKey 結構，其中包含管理類別，以及與物件相關聯副本群組資訊。 在繼續傳送前檢查副本群組的目的地。 當您在單一交易中傳送多個物件時，它們必須有相同的副本群組目的地。 **dsmBindMC** 函數呼叫會傳回以下的資訊：

表 *9. dsmBindMC* 呼叫傳回的資訊

資訊	說明
管理類別	與物件結合的管理類別名稱。 應用程式用戶端可傳送 **dsmBeginQuery** 呼叫以決定此管理類別的所有屬性。
備份副本群組	會在此管理類別中有備份副本群組時通知您。 如果正在執行備份作業並且備份副本不存在，則無法將此物件傳送至儲存體。如果您嘗試使用 **dsmSendObj** 呼叫來傳送它時，您會收到錯誤碼。
備份副本目的地	此欄位識別將資料傳送至的儲存區。如果您執行多重物件備份交易，該交易中的所有副本目的地必須相同。 如果物件的副本目的地與交易中的先前物件不同，結束目前的交易，並且在您傳送物件前開始新交易。 如果您嘗試要在相同的交易中，將物件傳送至不同的副本目的地，您會收到錯誤碼。
保存副本群組	會在此管理類別中有保存副本群組時通知您。 如果正在執行備份作業並且保存副本群組不存在，則無法將此物件傳送至儲存體。如果您嘗試使用 **dsmSendObj** 呼叫來傳送它時，您會收到錯誤碼。
保存副本目的地	此欄位識別將資料傳送至的儲存區。如果您執行多重物件保存交易，該交易中的所有副本目的地必須相同。 如果物件的副本目的地與交易中的先前物件不同，結束目前的交易，並且在您傳送物件前開始新交易。 如果您嘗試要在相同的交易中，將物件傳送至不同的副本目的地，您會收到錯誤碼。

如果採用相同物件名稱來執行後續的備份，但該備份使用與原來不同的管理類別，那麼物件的備份副本可以重新連結到另一個管理類別。 例如，如果您備份 ObjectA 並將它連結到 Mgmtclass1，後來您備份 ObjectA 並將它連結到 Mgmtclass2，那麼最新的備份會將任何非作用中副本重新連結到 Mgmtclass2。 現在定義於 Mgmtclass2 的 參數會控制所有複本。 不過，如果目的地不同的話，就不會移動資料。

您也可以使用 **dsmUpdateObj** 或 **dsmUpdateObjEx** 呼叫和 DSM_BACKUPD_MC 動作，將備份副本重新連結到另一個管理類別。

相關參考
Deduplication 選項

查詢管理類別

應用程式可以查詢管理類別，以便決定指定節點可能使用的管理類別，以及決定管理類別中會有哪些屬性。

您只能使用 **dsmBindMC** 呼叫來結合物件與管理類別。 您可能要用應用程式查詢管理類別屬性，並且將它們顯示給一般使用者。 如需相關資訊，請參閱 第 26 頁的『查詢 IBM Spectrum Protect 系統』。

在第 24 頁的圖 8 中的範例，switch 陳述式是用來在呼叫 **dsmBindMC** 時，區分備份和保存作業。 此呼叫所傳回的資訊是儲存在 **MCBindKey** 結構。

```
dsUint16_t     send_type;
dsUint32_t     dsmHandle;
dsmObjName   objName;     /* 結構包含物件名稱  */
mcBindKey    MCBindKey;   /* 管理類別資訊          */
char         *dest;       /* 儲存目的地值                */

switch (send_type)
{
   case (Backup_Send) :
      rc = dsmBindMC
(dsmHandle,&objName,stBackup,&MCBindKey);
      dest = MCBindKey.backup_copy_dest;
      break;
   case (Archive_Send) :
      rc = dsmBindMC
(dsmHandle,&objName,stArchive,&MCBindKey);
      dest = MCBindKey.archive_copy_dest;
      break;
   default : ;
}

if (rc)
{
   printf("*** dsmBindMC
 failed: ");
   rcApiOut(dsmHandle, rc);
   rc = (RC_SESSION_FAILED);
   return;
}
```

圖 *8.* 使管理類別與物件相關聯範例

到期/刪除保留和釋放

您可以保留特定保存物件的刪除和過期，以回應需要保留特定資料的擱置或進行中動作。在此事件中，起始了可能需要存取資料的動作，該資料的可用性必須持續到動作結束，且處理程序不再需要存取該資料為止。確定不再需要暫緩執行（解除）之後，便會依據原有保留期來回復正常的刪除和過期計時。

開始之前

發出測試的 **dsmRetentionEvent** 呼叫來驗證伺服器已獲授權：

1. 查詢一個要保留的物件並取得其 ID。
2. 發出 **dsmBeginTxn**、**dsmRetentionEvent**（搭配 Hold）和 **dsmEndTxn**。
3. 如果伺服器未被授權，您會收到中斷 (abort) 的 vote 值，原因碼是 DSM_RC_ABORT_LICENSE_VIOLATION。

限制:

1. 您無法在單一異動中發出多個 **dsmRetentionEvent** 呼叫。
2. 您無法保留已在保留下的物件。

程序

1. 如果要保留物件，請完成下列步驟：
   a) 查詢伺服器中您要保留的所有物件。 取得每個物件的物件 ID。
   b) 發出 **dsmBeginTxn** 呼叫，然後發出 **dsmRetentionEvent** 呼叫（並附帶物件清單），後接 **dsmEventType**: eventHoldObj 呼叫。 如果物件數超出 maxObjPerTxn 的值，請使用多重異動。
   c) 使用 **dsmGetNextQObj** 函數呼叫上的 qryRespArchiveData 回應來確認物件是否已經保留。檢查 qryRespArchiveData 中的 objHeld 的值。
2. 如果要從保留中釋放物件，請完成下列步驟：
   a) 查詢伺服器中要從保留中釋放的所有物件。 取得每個物件的物件 ID。
   b) 發出 **dsmBeginTxn** 呼叫，然後發出 **dsmRetentionEvent** 呼叫（並附帶物件清單），後接 **dsmEventType**: eventReleaseObj 呼叫。 如果物件數超出 maxObjPerTxn 的值，請使用多重異動。
   c) 使用 dsmGetNextQObj 函數呼叫上的 **qryRespArchiveData** 回應來確認物件是否已從保留中釋放。檢查 qryRespArchiveData 中的 **objHeld** 的值。

保存資料保留保護

受 IBM Spectrum Protect 控制的資料無法由未獲授權的代理（例如個人或程式）修改。這項保護會延伸，而防止任何代理在保留期過期前刪除資料（例如保存物件）。

關於這項作業

保護保存保留有助於確保個人或程式不會惡意或意外刪除受 IBM Spectrum Protect 控制的資料。傳送至保存保留保護伺服器的保存物件受到意外刪除保護比個具有強制施行的保留期限。保存保留保護存在下列限制：

- 保留保護伺服器上只容許執行保存作業。
- 未經由 **dsmBindMc** 函數呼叫中的值或 Include-Exclude 陳述式明確連結到管理類別的任何物件，將會連結到預設管理類別的明確名稱。 比方說，如果節點原則中的預設管理類別為 MC1，則物件將會明確連結到 MC1 而不是 DEFAULT。在查詢回應上，物件會顯示為連結到 MC1。
- 在啟用保存資料保留保護之後，若嘗試刪除保留期限過期之前的物件，會在結束交易時傳回回覆碼 DSM_RC_ABORT_DELETE_NOT_ALLOWED。

如需設定保存物件的保留保護的指示，請參閱 IBM Spectrum Protect 伺服器說明文件。

程序

如果要設定資料保留保護，請完成下列步驟：

1. 在不含先前的資料的新伺服器安裝中，執行 **SET ARCHIVERETENTIONPROTECTION ON** 指令。
2. 在 **dsmInit** 或 **dsmInitEx** 函數呼 叫的 API 選項字串中，輸入指令：

   ```
   -ENABLEARCHIVERETENTIONPROTECTION=yes
   ```

   您也可以在非 UNIX 系統上的 dsm.opt 檔案或 UNIX 系統上的 dsm.sys 檔案中設定 enablearchiveretentionprotection 選項：

   ```
   SERVERNAME srvr1.ret
      TCPPORT                          1500
      TCPSERVERADDRESS                 node.domain.company.com
      COMMMETHOD TCPIP
      ENABLEARCHIVERETENTIONPROTECTION YES
   ```

   如需此選項的相關資訊，請參閱第 26 頁的『enablearchiveretentionprotection 選項』。
3. 對伺服器發出查詢，確認 IBM Spectrum Protect 伺服器是否已啟用保存保留保護。檢查 dsmQuerySessInfo 結構中 archiveRetentionProtection 欄位的值。

enablearchiveretentionprotection 選項

enablearchiveretentionprotection 選項指定是否要對 此用途專用的 IBM Spectrum Protect 伺服器上之保存物件啟用資料保留保護。您的伺服器管理者必須對尚未儲存物件 （指備份、保存或空間管理物件）的新伺服器啟動資料保留保護。如果 API 應用程式試圖在此伺服器儲存備 份版本或空間管理物件，則系統會發出錯誤訊息。

第 9 頁的『第 3 章 設計應用程式的注意事項』中的附註說明：「請不要儲存 objectID 值以供日後還原使用。在物件生命週期中，這些值並不一定會持續存在。」對於 Archive Manager 應用程式不必擔心，因為 Archive Manager 伺服器不支援匯出或匯入。在物件還原期間，Archive Manager 應用程式可以儲存及使用 objectID 來增進效能。

如果伺服器發出 **SET ARCHIVERETENTIONPROTECTION ON** 指令，則您無法使用 **delete filespace** 指令刪除伺服器中保存的物件，除非指定適當的保存副本群組原則參數。有關如何設定管理類別的資訊，請參閱適當的伺服器說明文件。

以事件為基礎的保留原則

在事件型保留原則中，保存物件的保留時間由商業事件（例如關閉銀行帳號）起始。事件型保留使得 IBM Spectrum Protect 資料保留原則與企業的資料需求更加一致。當事件發生時，應用程式會將該物件的 **eventRetentionActivate** 事件傳送至伺服器，以起始保留。

程序

如果要使用以事件為基礎的保留原則，請完成下列步驟：

1. 在伺服器上，以 EVENT 類型的保存 **copygroup** 建立管理類別。如需相關資訊，請參閱 IBM Spectrum Protect 伺服器說明文件。
2. 查詢該管理類別以確認類別為事件型。如果管理類別為事件型，則 **archDetailCG** 結構中的 **retainInit** 欄位為 ARCH_RETINIT_EVENT。
3. 將物件連結到以事件為基礎的管理類別，方法是使用 include、**archmc**，或明確透過 **dsmSendObj** 函數呼叫上 **ObjAttr** 結構中的 **mcNameP** 屬性。
4. 當您要啟動物件保留時，查詢伺服器中所有受影響的物件。檢查它們是否處於 PENDING 狀態，並取得物件 ID。在擱置狀態中，**qryRespArchiveData** 結構中的 **retentionInitiated** 欄位指示 DSM_ARCH_RETINIT_PENDING。
5. 發出 **dsmBeginTxn** 呼叫，然後發出 **dsmRetentionEvent** 呼叫（並附帶物件清單），後接 **dsmEventType**: eventRetentionActivate 呼叫。 如果物件數超出 maxObjPerTxn 的值，請使用多重異動。

   限制: 針對每一個交易只能發出一次 **dsmRetentionEvent** 呼叫。
6. 查詢物件以確認是否已啟動保留。如果保留已起始，則 **qryRespArchiveData** 結構中的 **retentionInitiated** 欄位的值為 I。

查詢 IBM Spectrum Protect 系統

API 有管理類別查詢等數種查詢可供應用程式使用。

程序

使用 **dsmBeginQuery** 呼叫的所有查詢都遵循下列步驟：

1. 傳送 **dsmBeginQuery** 呼叫與適當的查詢類型：
   - 備份
   - 保存
   - 作用中的備份物件
   - 檔案空間

· 管理類別

dsmBeginQuery 呼叫會通知 API 有關從伺服器傳回的資料格式。您可以在 **dsmGetNextQObj** 呼叫所傳遞的資料結構中放置適當的欄位。 您也可以在開始的查詢呼叫中適當地指定參數，而讓應用程式用戶端設定查詢的範圍。

限制: 在 UNIX 或 Linux 系統上，只有 root 使用者可以查詢作用中的備份物件。此查詢類型稱為「捷徑」。

2. 輸入 **dsmGetNextQObj** 呼叫以便從查詢中取得每個紀錄。 此呼叫會傳遞夠大的緩衝區，以便 保留從查詢傳回的資料。 每個查詢類型都有對應的資料傳回的資料結構。 例如，備份查詢類型有在傳送 **dsmGetNextQObj** 呼叫時所需移入的相關聯 **qryRespBackupData** 結構。
3. **dsmGetNextQObj** 呼叫通常會傳回下列其中一個代碼：

   · DSM_RC_MORE_DATA：再次傳送 **dsmGetNextQObj** 呼叫。

   · DSM_RC_FINISHED：沒有更多資料。傳送 **dsmEndQuery** 呼叫。
4. 傳送 **dsmEndQuery** 呼叫。 已擷取所有查詢資料或不再需要其他查詢資料時，請輸入 **dsmEndQuery** 呼叫來結束查詢程序。API 清除任何來自查詢串流的剩餘資料，並且釋放由查詢所使用的任何資源。

結果

第 27 頁的圖 9 顯示了查詢作業的狀態圖。

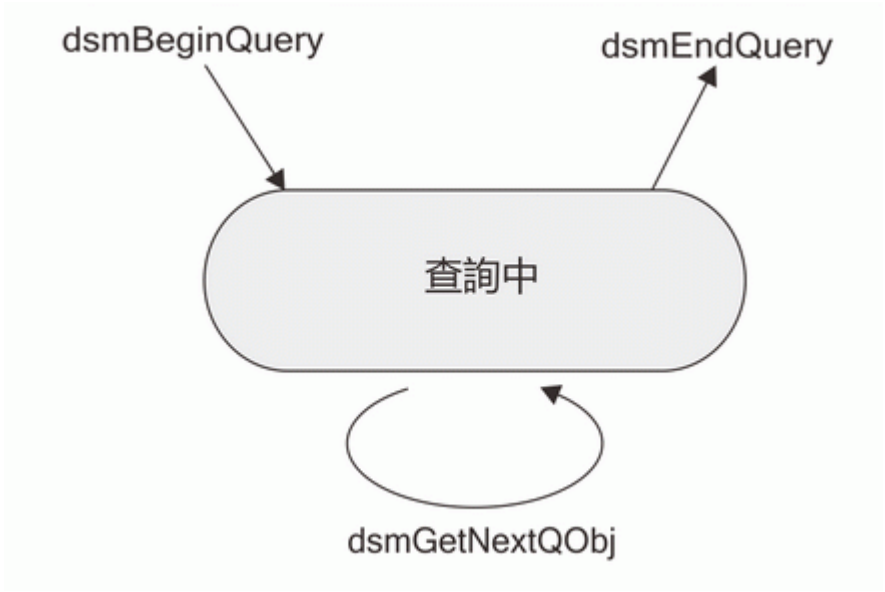


圖 9. 一般查詢的狀態圖

第 28 頁的圖 10 顯示了查詢作業的流程圖。

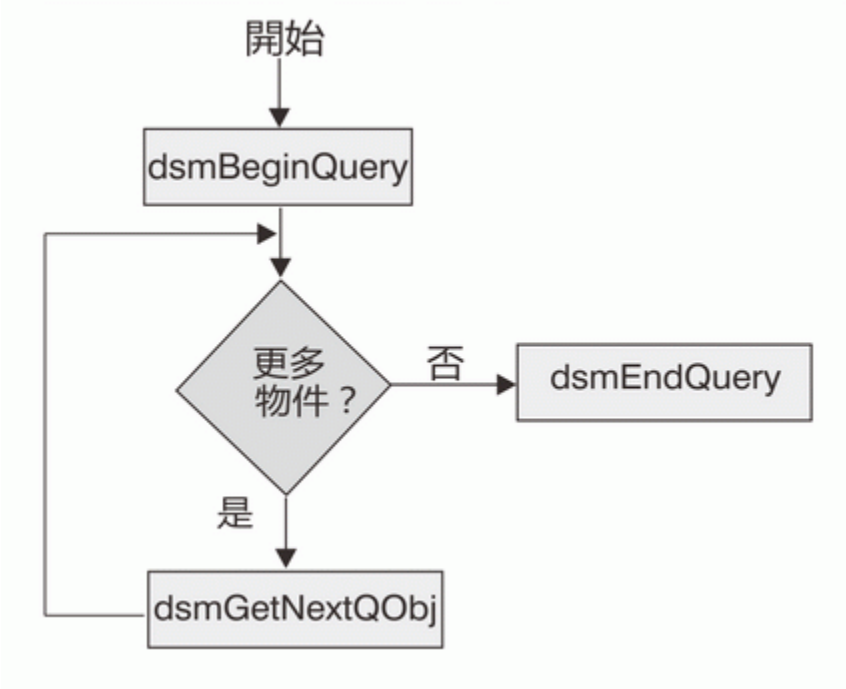


圖 *10.* 一般查詢的流程圖

查詢系統的範例

在本例中，管理類別查詢會列印出特定管理類別所屬備份和保存副本群組中的所有欄位值。

```
dsInt16                rc;
qryMCData             qMCData;
DataBlk               qData;
qryRespMCDetailData   qRespMCData, *mcResp;
char                  *mc, *s;
dsBool_t  done = bFalse;
dsUint32_t               qry_item;

/* 填寫需要的 qMCData 結構和查詢準則 */
qMCData.stVersion = qryMCDataVersion;   /* 結構版本      */
qMCData.mcName    = mc;                 /* 管理類別名稱 */
qMCData.mcDetail  = bTrue;              /* 需要完整詳細資訊？    */

/* 設定使用的資料區塊參數來取得或傳送資料     */
qData.stVersion = DataBlkVersion;
qData.bufferLen = sizeof(qryRespMCDetailData);
qData.bufferPtr = (char *)&qRespMCData;

qRespMCData.stVersion = qryRespMCDetailDataVersion;
```

```
if ((rc = dsmBeginQuery
(dsmHandle,qtMC,(dsmQueryBuff *)&qMCData)))
{
   printf("*** dsmBeginQuery
 failed: ");
   rcApiOut(dsmHandle, rc);
   rc = (RC_SESSION_FAILED);
}
else
{
   done = bFalse;
   qry_item = 0;
   while (!done)
   {
      rc = dsmGetNextQObj
(dsmHandle,&qData);
      if ((   (rc == DSM_RC_MORE_DATA)
           || (rc == DSM_RC_FINISHED))
          && qData.numBytes)
      {
         qry_item++;
         mcResp = (qryRespMCDetailData *)qData.bufferPtr;
         printf("Mgmt. Class
         printf("               Name:
         printf("   Backup CG Name:
            .
            .  /* 其他備份欄位及保存副本群組 */
            .
         printf("   Copy Destination:
      }
      else
      {
         done = bTrue;
         if (rc != DSM_RC_FINISHED)
         {
            printf("*** dsmGetNextQObj
 failed: ");
            rcApiOut(dsmHandle, rc);
         }
      }
      if (rc == DSM_RC_FINISHED) done = bTrue;
   }
   rc = dsmendQuery (dsmHandle);
}
```

圖 *11.* 執行系統查詢範例

伺服器效率

當您傳送物件到 IBM Spectrum Protect 伺服器，或從伺服器擷取物件時，請使用這些準則。

- 當您從 IBM Spectrum Protect 伺服器中擷取物件時，請遵循下列準則：
  - 按照 IBM Spectrum Protect 伺服器所提供的還原順序來擷取資料。 此還原順序對於磁帶機尤其重要，因為不照順序擷取資料可能會導致磁帶倒轉和裝載。

- 即使資料是儲存在磁碟裝置上時，按照順序擷取仍可節省時間。
- 請在單一 IBM Spectrum Protect 伺服器階段作業中盡量執行工作。
- 請勿啟動和停止多個階段作業。

· 當您將物件傳送到 IBM Spectrum Protect 伺服器時，請遵循下列準則：

- 在單一交易中傳送多個物件。
- 避免每個交易傳送一個物件，特別是在將資料直接傳送至磁帶機時。 部分磁帶機交易是要確保磁帶的 RAM 緩衝區中的資料會寫入媒體。

相關概念

第 54 頁的『依照還原順序來選取及排序物件』
在執行備份或保存查詢之後，應用程式用戶端必須判斷要還原或擷取哪些物件。

相關資訊

第 14 頁的『啟動或結束階段作業』
IBM Spectrum Protect 是一項階段作業型產品，而且必須在 IBM Spectrum Protect 階段作業內執行所有活動。為了啟動階段作業，應用程式會啟動 **dsmInitEx** 呼叫。 您必須在 **dsmQueryApiVersionEx** 、**dsmQueryCliOptions** 或 **dsmSetUp** 以外的任何 API 呼叫前，執行此呼叫。

傳送資料到伺服器

應用程式用戶端可利用此 API，將資料或指名的物件和它們的關聯資料傳送到 IBM Spectrum Protect 伺服器儲存體。

提示: 您可以備份或保存資料。 在交易中執行所有的傳送作業。

交易模型

在一個交易內完成在備份或保存作業期間傳到 IBM Spectrum Protect 儲存體的資料。 交易模型可以提供高層次資料完整性，但也會造成應用程式用戶端必須加以考慮的某些限制。

呼叫 **dsmBeginTxn** 以啟動階段作業，或呼叫 **dsmEndTxn** 以便結束交易。 單一交易是一個極小的動作。在交易界限內傳送的資料會在結束交易時確定至系統，或在交易提前結束時回復。

交易是由單一物件傳送或多個物件傳送所構成。 如果要藉由減少系統額外需要來增進系統效能，請在多個物件交易中傳送較小的物件。 應用程式用戶端可判斷單一或多重交易比較適合。

在多重物件交易中，將所有的物件傳送至相同的複製目的地。 如果您要將物件傳送至與先前物件不同的目的地，結束目前的交易然後啟動新的交易。 在新的交易中，您可以將物件傳送至新的副本目的地。

註: 沒有包含任何位元資料的物件 (*sizeEstimate=0*)，不會檢查複製目的地一致性。

IBM Spectrum Protect 會限制在多重物件交易中傳送的物件數目。 若要找出此限制，請呼叫 **dsmQuerySessInfo** 並且檢查 **maxObjPerTxn** 欄位。 此欄位會顯示伺服器上所設定的 *TXNGroupmax* 選項值。

應用程式用戶端必須追蹤交易中所傳送的物件，以執行重試程序或交易提前結束時的錯誤處理程序。 伺服器或用戶端可隨時停止交易。 如果交易沒有啟動，應用程式用戶端必須準備處理突然的交易終止。

檔案集成

IBM Spectrum Protect 伺服器可使用名為檔案集合體的函數。 檔案集合體可讓 傳送至單一交易中的所有物件儲存在一起，以節省空間並增進效能。 您仍然可以單獨查詢和還原物件。

若要使用此函數，那麼交易中的所有物件必須採用相同檔案空間名稱。 如果在交易內變更檔案空間名稱，伺服器會關閉現有的聚集物件並開始新的聚集物件。

不需 LAN 的資料移動 (LAN-free data transfer)

如果多執行緒的 **dsmSetUp** 選項為 ON，則 API 可以使用不需 LAN 的資料傳送。 API 會在 **Query Mgmt Class** 回應結構的 **archDetailCG** 或 **backupDetailCG** 欄位 **bLanFreeDest** 中，傳回有不需 LAN 目的地。

您可以在平台上使用受儲存體代理站支援的不需 LAN 作業。 Macintosh 平台並不包含在內。

免用 LAN 資訊在下列輸出結構中提供。 **dsmEndGetData** 的輸出 結構 (**dsmEndGetDataExOut_t**) 包含 **totalLFBytesRecv** 欄位。 這是收到的免用 LAN 總位元組數。 **dsmEndSendObjEx** 的 輸出結構 (**dsmEndSendObjExOut_t**) 包含 **totalLFBytesSent** 欄位。 這是 已傳送的免用 LAN 總位元組數。

相關資訊

不需 LAN 的資料移動：儲存體代理站概觀

同步寫入作業

您可以配置 IBM Spectrum Protect 伺服器儲存區在備份或保存期間同步寫入主要儲存區和副本儲存區。 請使用此配置來建立多個物件副本。

如果同步寫入作業失敗，**dsmEndTxn** 函數的回覆碼可能是 DSM_RC_ABORT_STGPOOL_COPY_CONT_NO，指出寫入其中一個副本儲存區失敗，IBM Spectrum Protect 儲存區選項 COPYCONTINUE 已設為 NO。應用程式終止，IBM Spectrum Protect 伺服器管理者必須解決此問題。

如需設定同步寫入作業的相關資訊，請參閱 IBM Spectrum Protect 伺服器說明文件。

增強 API 效能

您可以使用 tcpbuffsize 和 tcpnodelay 用戶端選項，並使用 **DataBlk** API 參數，來增強 API 效能。

text="可增強效能的備份保存選項與 API 參數" 說明對於增強 API 效能，您可以採取的動作。

表 *10.* 可增強效能的備份保存選項與 *API* 參數

備份保存用戶端選項	**說明**
tcpbuffsize	指定 TCP 緩衝區的大小。 預設值是 31 KB。 如果要增強效能，請將值設為 32 KB。
tcpnodelay	指定是否將小型緩衝區傳到伺服器而不要保留它們。 為了增強效能，請對所有平台，將此選項設為 *yes*。 這個選項僅適用於 Windows 和 AIX。
API 參數	**說明**
DataBlk	這個參數與 **dsmSendData** 函數呼叫搭配使用，可判定應用程式緩衝區大小。為了得到最佳效果，請將參數設為 tcpbuffsize 所指定之 *tcpbuffsize* 值的倍數再減去 4 個位元組。 例如，如果 tcpbuffsize 的值設為 32 KB，請設定此參數值為 28。

每一個 **dsmSendData** 呼叫會同步，而且要等 **dataBlkPtr** 中傳送到 API 的資料寫入網路之後，才會返回。 API 會將 4 位元組額外負擔新增到放在網路的每個交易緩衝區。

例如，當交易緩衝區大小是 32 KB 而且應用程式 **DataBlk** 緩衝區大小是 31 KB 時，每個應用程式 **DataBlk** 緩衝區剛好可放入通訊緩衝區並且可以立即清除。 不過，如果應用程式 **DataBlk** 緩衝區正好是 32 KB，而且因為 API 會在每個交易緩衝區增加 4 個位元組，所以會進行二次清除；一次是 32 KB，另一次是 4 位元組。 另外，如果您將 tcpnodelay 選項設為 no，清除 4 位元組可能最長需要 200 毫秒。

設定 API 以將效能資料傳送至用戶端效能監視器

用戶端效能監視器是「Tivoli® Storage Manager 管理中心」的一個元件， 可用來顯示 API 所收集的效能資料。用戶端效能監視器會記錄並顯示用戶端的備份、保存和還原等作業的效能資料。

啟用效能監視之後，您可以使用效能監視器顯示 API 所收集的效能資料；效能監視器在「Tivoli Storage Manager 管理中心」中提供。從 7.1 版開始，該「管理中心」元件不再包括在 Tivoli Storage Manager 或 IBM Spectrum Protect 發行套件中。如果您的「管理中心」是隨舊版伺服器安裝的，則可以繼續使用它來顯示效能資料。如果您尚未安裝「管理中心」，則可以從 ftp://public.dhe.ibm.com/storage/tivoli-storage-management/maintenance/admincenter/v6r3/ 下載以前發佈的版本。如需使用效能監視器的相關資訊，請參閱 Tivoli Storage Manager 6.3 版伺服器說明文件。

配置用戶端效能監視器選項

您透過在用戶端選項檔案中指定相應的參數，容許 IBM Spectrum Protect 用戶端使用效能監視器。您針對要監視的每一個用戶端指定這些選項。

開始之前

監視 UNIX 和 Linux 電腦上的效能時，透過使用下列指令將開啟檔案描述子限制設為至少 1024：

```
ulimit -n 1024
```

程序

若要配置用戶端效能監視器選項，請完成下列步驟：

1. 針對要監視的每個用戶端，開啟用戶端選項檔案。根據您的配置，用戶端選項在下列其中一個檔案中：
   - dsm.opt
   - dsm.sys
2. 將下列選項新增至用戶端選項檔案：

   ```
   PERFMONTCPSERVERADDRESS
   PERFMONTCPPORT
   PERFMONCOMMTIMEOUT
   ```

PERFMONTCPSERVERADDRESS

PERFMONTCPSERVERADDRESS 選項指定安裝用戶端效能監視器所在系統的主機名稱或 IP 位址。

支援的用戶端

這個選項是獨立於平台之外，而且所有的用戶端都有支援。

選項檔

請在用戶端選項檔（dsm.opt 或 dsm.sys）中設定這個選項。

語法

參數

server
已安裝用戶端效能監視器的系統的伺服器主機名稱或 IP 位址（與執行「管理中心」相同的伺服器）。

範例

選項檔：

```
PERFMONTCPSERVERADDRESS 131.222.10.5
```

指令行：

無法使用指令行來設定這個選項。

PERFMONTCPPORT

用戶端效能監視器接聽來自用戶端之效能資料的埠號。

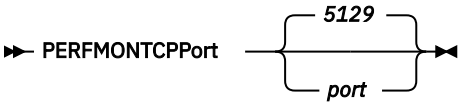
支援的用戶端

這個選項是獨立於平台之外，而且所有的用戶端都有支援。

選項檔

請在用戶端選項檔（dsm.opt 或 dsm.sys）中設定這個選項。

語法



參數

port

用來監視用戶端效能資料的埠。 5129 埠是預設埠。

範例

選項檔：

```
PERFMONTCPPPORT 5000
```

指令行：

無法使用指令行來設定這個選項。

PERFMONCOMMTIMEOUT

指定在階段作業結束之後， dsmTerminate 呼叫等待效能資料抵達的秒數上限。

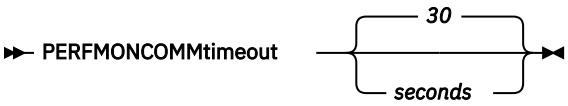
支援的用戶端

這個選項是獨立於平台之外，而且所有的用戶端都有支援。

選項檔

請在用戶端選項檔（dsm.opt 或 dsm.sys）中設定這個選項。

語法



參數

seconds

在結束階段作業之前，要等待剩餘資料到到達的時間。

範例

選項檔：

```
PERFMONCOMMTIMEOUT 60
```

指令行：

無法使用指令行來設定這個選項。

傳送物件到伺服器

應用程式用戶端可以使用 API 備份和保存函數，將資料或指名的物件以及它們的相關資料傳送到 IBM Spectrum Protect 儲存體。 系統的備份和保存元件允許對傳到儲存體的資料使用不同管理程序。

該大小預估屬性是傳送至伺服器的資料物件的預估總大小。 如果應用程式不知道物件的實際大小，請將 *sizeEstimate* 設定為較高的預估值。 如果預估值小於實際大小，IBM Spectrum Protect 伺服器會使用額外的資源，來管理額外的空間配置。

秘訣:

- 當您在預估這個大小時，請盡可能的精準。 伺服器會使用這個屬性，在其儲存體資源中進行有效率的空間配置以及物件放置。
- 如果預估值小於實際大小，具有快取的伺服器不會配置額外的空間，並會停止傳送。

如果 *sizeEstimate* 過大，您可能會遭遇問題。 伺服器可能沒有足夠空間來儲存預估的大小，但是有儲存實際大小的空間；或者伺服器可能使用速度較慢的裝置。

您可以備份或保存大小超過 2 GB 的物件。 物件可以壓縮或不壓縮。

如果要啟動傳送作業，請呼叫 **dsmSendObj** 。 如果資料大小超過一次所能傳送的量，您可以重複呼叫 **dsmSendData** 來轉送其餘的資訊。 呼叫 **dsmEndSendObj** 來完成傳送作業。

了解備份和保存物件

IBM Spectrum Protect 系統的備份元件支援儲存在伺服器的數個版本的指名物件。

任何與來自該用戶端，要備份至伺服器的物件，若具有與已儲存於伺服器的物件相同的名稱，都要接受版本控制。 在伺服器上，物件會被視為作用中或者非作用中的狀態。 伺服器上未被關閉的物件最新副本則為作用中狀態。 不論是否為舊版或關閉的副本，任何同名物件都會被視為非作用中。 管理類別建構並定義不同的管理準則。 它們會被指定為伺服器上的作用中和非作用中物件。

第 34 頁的表 11 列出適用於作用中和非作用中狀態的副本群組欄位：

表 *11.* 備份副本群組欄位

欄位	**說明**
VEREXISTS	作用中版本存在時，非作用中版本的數目。
VERDELETED	作用中版本不存在時，非作用中版本的數目。
RETEXTRA	保留非作用中版本的天數。
RETONLY	作用中版本不存在時，保留最後非作用中版本的天數。

如果備份版本各有專用的名稱（如在名稱中使用時間戳記），則不會自動進行版本化：所有的物件都是在作用中。 作用中物件絕不會到期，所以應用程式會負責以 **dsmDeleteObj** 呼叫來關閉這些物件。 在此狀況中，應用程式會需要關閉物件以便盡可能使它到期。 使用者會以 VERDELETED=0 和 RETONLY=0 來定義備份副本群組。

IBM Spectrum Protect 系統的保存元件允許使用保留或到期期間控制而非版本控制，將物件儲 存在伺服器。 不論物件名稱是否可能與已保存的物件相同，所儲存的每個物件都是唯一的。 保存物件所擁有的說明欄位會與 meta 資料相關聯，而且可在查詢期間用來識別特定的物件。

IBM Spectrum Protect 伺服器上的所有物件都會被指定一個唯一的物件 ID。在物件的生命週期中，原始值並不一定會保持一定 （特別是在匯出或匯入後）。 因此，應用程式不應查詢以及儲存原始物件 ID，以供日後還原使用。 更確切地說，應用程式應儲存物件名稱和插入日期。 您可以在還原時使用此資訊來查詢物件，並且驗證插入日期。 而目前的物件 ID 可用來還原物件。

壓縮

給定節點的配置選項和 **dsmSendObj** `objCompressed` 選項，決定在傳送期間 IBM Spectrum Protect 是否會壓縮物件。 此外，具有 `sizeEstimate` 且小於 `DSM_MIN_COMPRESS_SIZE` 的物件，一律不會被壓縮。

如果物件已被壓縮 (objCompressed=bTrue)，就不會再壓縮。 如果物件未被壓縮，IBM Spectrum Protect 會根據壓縮選項的值來決定是否壓縮此物件，這些壓縮選項值是由管理者所設定，也可以在 API 配置來源中設定。

管理者可以使用登錄節點指令（compression=yes、no 或 client-determined）來變更壓縮臨界值。 如果這個指令是 client-determined，就會由配置來源中的壓縮選項值決定壓縮行為。

已壓縮的資料等部分類型的資料，實際上可能會在以壓縮演算法處理時變大。 在發生此狀況時，則會產生 DSM_RC_COMPRESS_GREW 回覆碼。 如果您知道 這種情況可能會發生，但是又要傳送作業繼續進行，請告訴使用者在選項檔中指定下列選項：

```
   COMPRESSAlways Yes
```

如果在進行啟用壓縮的 **dsmSendData** 函數期間出現 DSM_RC_COMPRESS_GREW 回覆碼，則您應重新開始，在沒有壓縮下重新傳送物件。 如果要強制執行此動作，請將 **dsmSendObj** ObjAttr.objCompressed 設為 bTrue。

dsmEndSendObjEx 呼叫傳回關於在 **dsmSendObj** 呼叫期間的實際壓縮行為之資訊。 objCompressed 指定是否已完成壓縮。 totalBytesSent 是應用程式傳回的位元組數。 totalCompressedSize 是壓縮之後的位元組數。 **dsmEndSendObjEx** 呼叫也有 totalLFBytesSent 欄位，其中包含透過免用 LAN 送的位元組總數。

小心: 如果您的應用程式打算使用部分物件還原或擷取，您就不能在傳送資料時壓縮資料。 如果要強制執行此動作，請將 **dsmSendObj** ObjAttr.objCompressed 設為 bTrue。

壓縮類型

用戶端使用的壓縮類型取決於備份或保存處理期間使用的壓縮及用戶端重複資料刪除組合。

用戶端使用的壓縮演算法由 API 在 **qryRespArchiveData** 和 **qryRespBackupData** 結構的新欄位中報告：

```
dsChar_t        compressAlg[20];  /* compression algorithm name */
```

報告了使用下列壓縮類型：

LZ4
一種較快且效率較高的壓縮方法，在下列任意情況中，用戶端會使用這種壓縮方法：

- 將用戶端已刪除重複資料的物件傳送至 IBM Spectrum Protect 伺服器上的儲存器儲存區。伺服器必須為 7.1.5 版或更新版本。
- 物件未進行用戶端重複資料刪除處理。
- 物件只能進行傳統的伺服器端重複資料刪除處理。

LZW
一種傳統的壓縮類型，當將用戶端已刪除重複資料的物件傳送至伺服器上的傳統（非儲存器）儲存區時，用戶端會使用此壓縮方法。

空白欄位
用戶端不會壓縮物件。 不壓縮物件，因為在備份或保存處理期間，compression 選項設為 *no* 或未指定。 雖然用戶端不會壓縮物件，但伺服器可能會進行壓縮。

壓縮類型不可配置。這由備份保存用戶端在備份或保存處理時所決定。

範例

下列範例顯示來自 64 位元範例應用程式 **dapismp** 中的備份和保存查詢輸出中的 Compresssion Type 欄位：

```
Enter selection ==>1
                        Filespace:\fs1
                        Highlevel:\hl
                         Lowlevel:\ll
               Object Type(D/F/A):f
```

```
   Active(A),Inactive(I),Both(B):a
If root, query all owners? (Y/N):
               Object Owner Name:
   point in time date (MMDDYYYY):
   point in time time     (hhmm):
     Show detailed output? (Y/N):y
On Restore, Wait for mount?(Y/N):
Are the above responses correct (y/n/q)?


Item 1: \fs1\hl\ll
   Object type: File
   Object state: Active
   Insert date: 2016/2/3 10:57:41
   Expiration date: 0/0/0 0:0:0
   Owner:
   Restore order: 0-0-0-0-0
   Object id: 0-40967
   Copy group: 1
   Media class: Fixed
   Mgmt class: DEFAULT
   Object info is        :IBM Spectrum Protect API Verify Data
   Object info length is :73
   Estimated size : 0  4000
   Compression : YES
   Compression Type: LZ4
   Encryption : NO
   Encryption Strength : NONE
   Client Deduplicated : YES
```

緩衝區副本排除

緩衝區副本排除功能可移除應用程式與 IBM Spectrum Protect 伺服器之間的資料緩衝區副本，這會產生較佳的處理器使用量。為達最大效果，請在不需 LAN 環境中使用這種方式。

資料移動的緩衝區是由 IBM Spectrum Protect 所配置，並將指標傳回應用程式。 應用程式將資料放在提供的緩衝區中，然後利用共用記憶體，將該緩衝區透過通訊層而傳送到儲存體代理站。 然後資料移至磁帶機，這樣就免去複製資料。 這項功能可與備份或保存作業一起使用。

小心: 當使用這個方法時，請特別注意緩衝區的處理方式和大小是否適當。 元件之間共用緩衝區，所以因程式設計錯誤而造成的任何記憶體改寫，都會導致嚴重錯誤發生。

備份/保存的整體呼叫順序如下所示：

```
dsmInitEx (UseTsmBuffers = True, numTsmBuffers = [how many IBM Spectrum Protect
           -allocated buffers the application needs to allocate])
dsmBeginTxn
for each object in the txn
  dsmBindMC
    dsmSendObject
      dsmRequestBuffer
      dsmSendBufferData (sends and release the buffer used)
    dsmEndSendObjEx
dsmEndTxn
for each buffer still held
      dsmReleaseBuffer
dsmTerminate
```

dsmRequestBuffer 函數可以多次呼叫，最高次數由 numTsmBuffers 選項所指定的值決定。 應用程式可以有兩個執行緒： 一個是生產者執行緒，用來在緩衝區填入資料；另一個是消費者執行緒，利用 **dsmSendBufferData** 呼叫將這些緩衝區傳送至 IBM Spectrum Protect。當發出 **dsmRequestBuffer** 呼叫並且已經到達 **numTsmBuffers** 時，系統會封鎖 **dsmRequestBuffer** 呼叫直到釋放緩衝區為止。 呼叫用於傳送和釋放緩衝區的 **dsmSendBufferData**，或是呼叫 **dsmReleaseBuffer**，就會發生緩衝區釋放。 如需相關資訊，請參閱 API 範例目錄中的 callbuff.c。

如果傳送過程中任何時候發生失敗狀況，應用程式必須釋放保留的所有緩衝區並終止階段作業。 例如：

```
  If failure
    for each data buffer held by application
      call dsmReleaseBuffer
  dsmTerminate
```

如果應用程式呼叫 **dsmTerminate** 但緩衝區仍被保留，則 API 不會結束。 傳回下列代碼：DSM_RC_CANNOT_EXIT_MUST_RELEASE_BUFFER。 如果應用程式無法釋放緩衝區，應用程式必須結束程序以強制清理。

緩衝區副本排除以及還原和擷取

IBM Spectrum Protect 伺服器會控制要存放在緩衝區中的資料量，這是基於還原和擷取作業的磁帶存取最佳化情況而定。 這個方法對應用程式而言，並不像取得資料的一般方法那樣有助益。 在建立原型期間，檢查緩衝區副本排除方法的效能，並且只在您看到實質改善時，才使用這個方法。

IBM Spectrum Protect 伺服器所傳回之單一緩衝區的最大資料量，為 256 KB 減掉標頭額外負擔的所得值。 因此，這項資料擷取機制只對處理小型緩衝區寫入的應用程式有幫助。 應用程式必須特別留意緩衝區中的位元組數，其會取決於物件大小、網路以及其他邊界條件。 在某些情況下，使用緩衝區副本排除的實際執行效能可能比一般還原方式更差。 API 通常會快取資料並傳回固定長度的資料給應用程式。 然後，應用程式會控制寫回磁碟的資料量。

如果您使用緩衝區副本排除，請對小於偏好的寫入緩衝區大小的緩衝區，建立資料快取機制。 比方說，如果應用程式寫入 64K 資料區塊到磁碟，應用程式必須執行以下動作：

1. 呼叫 **dsmGetBufferData**。
2. 寫出 64K 的區塊。
3. 在最後的區塊上，將剩餘資料複製到 **tempBuff**、發出另一個 **dsmGetBufferData** 呼叫，然後將剩餘資料填入 **tempBuff**。
4. 繼續寫入 64K 的區塊：

```
dsmGetBufferData #1 get 226K                dsmGetBufferData #2 get 240K
Block1 64K - write to disk          Block1 30K - copy to tempbuff-write to disk
Block2 64K - write to disk          Block2 64K - write to disk
Block3 64K - write to disk          Block3 64K - write to disk
Block4 34K - copy to tempbuff       Block4 64K - write to disk
Block5 18K - write to tempbuff        etc
```

在此範例中，六個磁碟寫入是直接，1 個是快取。

整體的還原和擷取呼叫順序如下：

dsmInitEx（UseTsmBuffers = True numTsmBuffers = 應用程式要配置的緩衝區數目）。

```
dsmBeginGetData
While obj id
  dsmGetObj (no data restored on this call- buffer set to NULL)
 While data to read
   dsmGetBufferData (returns the data in the data buffer)
  ...process data...
  dsmReleaseBuffer
dsmEndGetObj
dsmEndGetData
```

對每一個 **dsmGetBufferData** 呼叫實作 **dsmReleaseBuffer** 呼叫。 **dsmGetBufferData** 和 對應的 **dsmReleaseBuffer** 不需要連續。 應用程式可以先發出多個 **dsmGetBufferData** 呼叫來取得數個緩衝區，稍後再發出對應的 **dsmReleaseBuffer** 呼叫。 如需使用這個函數的範例程式碼，請參閱 API 範例目錄中的 callbuff.c。

限制: 因為此 API 提供緩衝區的目的是要將處理器使用率降至最低，因此不允許在緩衝區中進行其他的資料處理程序。應用程式無法將加密和壓縮與緩衝區副本排除搭配使用，因為這兩項作業都需要處理和複製資料。

請同時實作一般資料移動路徑和緩衝區副本排除，讓使用者依其需求來切換使用這兩個路徑。 如果使用者必須壓縮或加密資料，則使用現有的機制。 如果有處理器限制，則使用新機制。這兩個機制為互補關係，並無法完全取代對方。

API 加密

有兩種方法可用於加密資料：應用程式管理的加密和 IBM Spectrum Protect 用戶端加密。

僅選取並使用這些方法之一來加密資料。這兩種方法是互斥的，如果同時使用這兩種方法來加密，將無法還原或擷取部分資料。例如，假設應用程式使用應用程式管理的加密來加密物件 A，然後使用 IBM Spectrum Protect 用戶端加密來加密物件 B。在還原作業期間，如果應用程式設定選項以使用 IBM Spectrum Protect 用戶端加密並且它嘗試還原兩個物件，則只能還原物件 B；無法還原物件 A，因為它是使用應用程式管理的加密而不是用戶端加密進行加密的。

無論使用哪種加密方法，IBM Spectrum Protect 都必須啟用密碼鑑別。依預設，伺服器會使用 SET AUTHENTICATION ON。

API 使用 AES 128 位元或 AES 256 位元加密。AES 256 位元資料加密提供的資料加密形式比 AES 128 位元資料加密層次更高。使用 AES 256 位元加密來備份的檔案，無法經由更舊版本的用戶端來還原。啟用加密時可以使用或不使用壓縮。 如果您使用加密，則無法使用局部物件還原及擷取和緩衝區副本排除功能。

應用程式管理的加密

透過應用程式管理的加密，應用程式會提供金鑰密碼給 API（使用金鑰 DSM_ENCRYPT_USER），而管理金鑰密碼是應用程式的責任。

小心: 如果未儲存加密金鑰，而且您已忘記金鑰，您的資料將無法復原。

應用程式在 **dsmInitEx** 呼叫中提供密碼鎖密碼，在還原時它必須提供正確的密碼鎖密碼。

小心: 如果遺失金鑰密碼，那就無法還原資料了。

在進行同一物件的備份和還原（或保存和擷取）時，必須使用相同的密碼鎖密碼。此方法沒有 IBM Spectrum Protect 伺服器層級的相依關係。如果要設定此方法，應用程式需要執行下列步驟：

1. 在 **dsmInitEx** 呼叫中將 bEncryptKeyEnabled 變數設為 bTrue，並設定 encryptionPasswordP 變數來指向具有加密金鑰密碼的字串。
2. 對要加密的物件設定 include.encrypt。 例如，如果要加密所有資料，請設定：

   ```
   include.encrypt /.../* (UNIX)
   ```

   及

   ```
   include.encrypt *\...\* (Windows)
   ```

   若要加密物件 /FS1/DB2/FULL，請設定：

   ```
   include.encrypt /FS1/DB2/FULL
   ```

3. 在 Windows 的 **dsmInitEx** 呼叫中，於傳遞給 API 的選項字串中設定 ENCRYPTKEY=PROMPT|SAVE。這個選項也可以在 dsm.opt (Windows) 或 dsm.sys（UNIX 或 Linux）中設定。

依預設，encryptkey 選項是設定為 prompt。 此設定可確保金鑰不會被自動儲存。如果將 encryptkey 指定為 save，則 IBM Spectrum Protect 會將金鑰儲存在本端機器上，但如此一來，只有一個金鑰可適用於相同節點名稱的所有 IBM Spectrum Protect 作業。

在傳送物件之後，**dsmEndSendObjEx** 會指定物件是否已加密以及指定所使用的方法。*encryptionType* 欄位的可能值如下：

- DSM_ENCRYPT_NO
- DSM_ENCRYPT_USER
- DSM_ENCRYPT_CLIENTENCRKEY

下表列出 API 加密類型、必備條件和可用功能。

表 *12. API* 加密類型、必備條件和可用功能		
類型	**必備項目**	**可用功能**
ENCRYPTIONTYPE	無	在 Windows 的 **dsmInitEx** 呼叫中，於傳遞給 API 的選項字串中設定 ENCRYPTIONTYPE。依預設，ENCRYPTIONTYPE=AES128。
EncryptKey=save	無	API 和備份保存
EncryptKey=prompt	無	API 和備份保存
EncryptKey=generate	無	API 和備份保存
EnableClientEncryptKey	無	僅 API

註: 建議伺服器開啟鑑別。如果鑑別關閉，則不會加密金鑰，但仍會加密資料。 然而，建議不要使用此項。

第 39 頁的表 13 顯示授權使用者和非授權使用者如何在備份或還原 作業期間，根據指定給 passwordaccess 選項的值來加密或解密資料。若要執行下列授權使用者和非授權使用者，TSM.KDB、TSM.sth 和 TSM.IDX 檔案必須存在。授權使用者將 encryptkey 選項設定為 **save**，將 passwordaccess 選項設定為 **generate**。

表 *13.* 在 *UNIX* 或 *Linux* 上使用應用程式管理的金鑰來加密或解密資料			
作業	**passwordaccess 選項**	**encryptkey 選項**	**結果**
授權使用者備份	generate	save	資料加密。
	generate	prompt	如果 encryptionPasswordP 包含加密密碼，則資料加密。
	prompt	save	如果 encryptionPasswordP 包含加密密碼，則資料加密。
	prompt	prompt	如果 encryptionPasswordP 包含加密密碼，則資料加密。
授權使用者還原	generate	save	資料加密。
	generate	prompt	如果 encryptionPasswordP 包含加密密碼，則資料加密。
	prompt	save	如果 encryptionPasswordP 包含加密密碼，則資料加密。
	prompt	prompt	如果 encryptionPasswordP 包含加密密碼，則資料加密。
非授權使用者備份	generate	save	資料加密。
	generate	prompt	如果 encryptionPasswordP 包含加密密碼，則資料加密。
	prompt	save	如果 encryptionPasswordP 包含加密密碼，則資料加密。
	prompt	prompt	如果 encryptionPasswordP 包含加密密碼，則資料加密。
非授權使用者還原	generate	save	資料加密。
	generate	prompt	如果 encryptionPasswordP 包含加密密碼，則資料加密。
	prompt	save	如果 encryptionPasswordP 包含加密密碼，則資料加密。
	prompt	prompt	如果 encryptionPasswordP 包含加密密碼，則資料加密。

IBM Spectrum Protect 用戶端加密

IBM Spectrum Protect 用戶端加密使用透過 DSM_ENCRYPT_CLIENTENCRKEY 值管理的金鑰來保護您的資料。用戶端加密對於使用 API 的應用程式而言是通透的，例外的是對於已加密或已壓縮的物件，不能執行部分還原作業和擷取作業。

對於 IBM Spectrum Protect 用戶端加密和應用程式管理的加密而言，加密密碼指的是用於產生實際加密金鑰的字串值。加密密碼選項的值的長度是 1-63 個字元，但從它產生的金鑰長度一定是 8 位元組（適用於 56 DES）、16 位元組（適用於 128 AES）和 32 位元組（適用於 256 AES）。

小心: 如果無法使用加密金鑰，則無法還原或擷取資料。當使用 ENABLECLIENTENCRYPTKEY 進行加密時，加密金鑰是儲存在伺服器資料庫上。對於使用這個方法的物件，伺服器資料庫必須存在而且物件必須具有適當值來進行適當的還原。請確定您經常備份伺服器資料庫以防止資料流失。

這是比較簡單的實作方法，它會在每一個階段作業產生一個隨機加密金鑰，該金鑰將與物件一起儲存在 IBM Spectrum Protect 伺服器的伺服器資料庫中。在還原期間，儲存的密碼鎖用於解密。 使用這個方法時，管理金鑰是 IBM Spectrum Protect 的責任，應用程式完全不需要處理金鑰。因為金鑰儲存在伺服器資料庫中，則必須具有已加密物件的還原作業的有效 IBM Spectrum Protect 資料庫。金鑰在 API 和伺服器之間傳輸時，也會加密。金鑰的傳輸是安全的，並且當金鑰儲存在 IBM Spectrum Protect 伺服器資料庫中時，將會加密。唯有當節點資料在伺服器之間匯出時，金鑰才會以明文的方式隨附於匯出資料串流。

若要啟用 IBM Spectrum Protect 用戶端加密，請完成下列步驟：

1. 在傳遞給 dsmInitEx 呼叫上的 API 之選項字串中指定 -ENABLECLIENTENCRYPTKEY=YES，或在系統選項檔 dsm.opt (Windows) 或 dsm.sys（UNIX 或 Linux）中設定該選項。
2. 對要加密的物件設定 include.encrypt。 例如，如果要加密所有資料，請設定：

   ```
   include.encrypt /.../* (UNIX)
   ```

   及

   ```
   include.encrypt *\...\* (Windows)
   ```

   若要加密物件 /FS1/DB2/FULL，請設定：

   ```
   include.encrypt /FS1/DB2/FULL
   ```

刪除重複資料

刪除重複資料是一種藉由刪除冗餘的資料，來減少儲存需求的方法。

概觀

IBM Spectrum Protect 有兩種類型的刪除重複資料：用戶端刪除重複資料和伺服器端刪除重複資料。

用戶端重複資料刪除是一項刪除重複資料技術，在備份保存用戶端上進行在備份和保存處理期間，於資料傳送至 IBM Spectrum Protect 伺服器之前，移除冗餘的資料。使用用戶端刪除重複資料，可以減少透過區域網路所傳送的資料量。

伺服器端刪除重複資料 是指由伺服器所執行的刪除重複資料技術。 IBM Spectrum Protect 管理者可以在 **REGISTER NODE** 或 **UPDATE NODE** 伺服器指令中使用 **DEDUP** 參數指定刪除重複資料位置（用戶端或伺服器）。

加強功能

透過用戶端重複資料刪除，您可以：

- 從重複資料刪除中排除用戶端上的特定檔案。
- 啟用可減少用戶端及伺服器之間網路資料流量的重複資料刪除快取。 快取包含在先前的增量備份作業中傳送至伺服器的範圍。 用戶端查詢其快取，而不查詢伺服器中是否存在範圍。

指定用戶端快取的大小及位置。 如果在伺服器及本端快取之間偵測到不一致，則會移除並重新移入本端快取。

註: 針對使用 IBM Spectrum Protect API 的應用程式，不得使用重複資料刪除快取，因為快取與 IBM Spectrum Protect 伺服器不同步可能導致備份失敗。 如果配置多個並行備份保存用戶端階段作業，則必須針對每個階段作業配置個別快取。

- 啟用用戶端重複資料刪除和壓縮來減少伺服器儲存的資料量。 在將每個範圍傳送至伺服器之前都進行壓縮。 在儲存體節約與壓縮用戶端資料所需要的處理能力之間進行平衡。 一般而言，如果壓縮並刪除用戶端系統上的重複資料，則使用的處理能力大約為單獨刪除重複資料的兩倍。

  伺服器可以處理刪除重複資料的壓縮資料。 此外，早於 6.2 版的備份保存用戶端可以還原刪除重複資料的壓縮資料。

用戶端刪除重複資料使用下列處理程序：

- 用戶端建立範圍。 延伸範圍 是檔案中的部分，可與其他檔案延伸範圍相互比較，以識別出重複資料。
- 用戶端與伺服器協力合作識別重複的延伸範圍。 用戶端傳送非重複的延伸範圍給伺服器。
- 後續的用戶端刪除重複資料作業會建立新範圍。 部分或全部的那些延伸範圍可能會與在先前的刪除重複資料作業中所建立的並且傳送到伺服器的延伸範圍相符。 相符的延伸範圍不會再度傳送到伺服器。

好處

用戶端重複資料刪除提供數個優點：

- 它可以減少透過區域網路 (LAN) 傳送的資料量。
- 識別重複資料所需要的處理能力會從伺服器卸載到用戶端節點。 伺服器端重複資料刪除一律針對啟用重複資料刪除的儲存區啟用。 但位於啟用重複資料刪除之儲存區中的檔案及由用戶端刪除重複資料的檔案，不需要其他處理。
- 將刪除移除伺服器上重複資料所需要的處理能力，可立即節約伺服器上的空間。

用戶端重複資料刪除可能有一個缺點。 伺服器沒有用戶端檔案的完整副本，除非您將包含用戶端範圍的主要儲存區備份到不刪除重複資料的副本儲存區。 （範圍 是在重複資料刪除期間建立的檔案的一部分。） 在儲存區備份到不刪除重複資料的儲存區期間，用戶端範圍會重新組合到連續檔案中。

依預設，必須將針對刪除重複資料設定的主要循序存取儲存區備份到不刪除重複資料的副本儲存區，才能收回這些儲存區及移除重複資料。 預設值確保伺服器在主要儲存區或副本儲存區中一直具有全部檔案的副本。

重要: 如果要進一步減少資料，您可以同時啟用用戶端刪除重複資料和壓縮。 在將每個範圍傳送至伺服器之前都進行壓縮。 壓縮可以節省空間，但是它會增加用戶端工作站上的處理時間。

以下是與刪除重複資料相關的選項：

- 刪除重複資料
- Dedupcachepath
- Dedupcachesize
- Enablededupcache
- Exclude.dedup
- Include.dedup

API 用戶端刪除重複資料

API 會在備份保存用戶端上使用用戶端刪除重複資料，在備份和保存處理期間，於資料傳送至 IBM Spectrum Protect 伺服器之前，移除冗餘的資料。

API 會使用用戶端刪除重複資料，在備份和保存處理期間，於資料傳送至 IBM Spectrum Protect 伺服器之前，移除冗餘的資料。使用用戶端刪除重複資料，可以減少透過區域網路所傳送的資料量。 使用用戶端刪除重複資料，也可以減少 IBM Spectrum Protect 伺服器儲存體空間。

在對用戶端啟用用戶端刪除重複資料，並且執行備份或保存作業時，資料會傳送至伺服器作為延伸範圍。 下次執行備份或保存作業時，用戶端和伺服器會識別哪些資料延伸範圍已經備份或保存，然後只傳送唯一延伸範圍的資料給伺服器。

要進行用戶端刪除重複資料，伺服器和 API 必須是 6.2 版或更新版。

在使用用戶端刪除重複資料來備份或保存檔案之前，系統必須符合下列需求：

- 用戶端必須啟用 deduplication 選項。
- 伺服器必須在 **REGISTER NODE** 或 **UPDATE NODE** 指令上使用 **DEDUP=CLIENTORSERVER** 參數，對用戶端啟用用戶端刪除重複資料。
- 資料的儲存區目的地必須是一個已啟用刪除重複資料的儲存區。 啟用刪除重複資料的儲存區只是檔案裝置類型。
- 確定檔案連結到正確的管理類別。
- 檔案可以排除在用戶端的刪除重複資料處理程序之外。 依預設，會包括所有檔案。
- 檔案大小必須大於 2 KB。
- 伺服器可以藉由在伺服器上設定 CLIENTDEDUPTXNLIMIT 選項， 來限制刪除重複資料的交易大小上限。 如需此選項的相關資訊，請參閱伺服器說明文件。

如果未符合這其中的任何需求，則會正常處理資料，但是不使用用戶端刪除重複資料。

以下是一些刪除重複資料限制：

- 不需 LAN 的資料移動與用戶端刪除重複資料互斥。 如果您同時啟用不需 LAN 的資料移動和用戶端刪除重複資料，則不需 LAN 的資料移動作業會完成，但是會忽略用戶端刪除重複資料。
- 加密與用戶端刪除重複資料互斥。 如果您同時啟用加密和用戶端刪除重複資料，則加密作業會完成，但是會忽略用戶端刪除重複資料。 加密的檔案及可進行用戶端刪除重複資料的檔案，可以在同一個作業中處理，但是會在個別的交易中執行。

  需求:

  1. 在任何交易中，必須在刪除重複資料中併入或排除所有的檔案。 如果交易具有混合的檔案，則交易會失敗，並且 API 會傳回回覆碼 DSM_RC_NEEDTO_ENDTXN。
  2. 將儲存裝置加密與用戶端刪除重複資料一起使用。 由於 SSL 與用戶端刪除重複資料合併使用，因此不需要進行用戶端加密。
- 進行用戶端刪除重複資料時，無法使用下列功能：
  - IBM Spectrum Protect for Space Management (HSM) 用戶端
  - API 共用緩衝區
  - NAS
  - 子檔案備份
- 緩衝區副本排除不能與壓縮、加密和刪除重複資料之類的資料轉換一起使用。
- 如果您使用用戶端複製，在傳送資料至伺服器期間，API 會偵測到伺服器上已標示為過期的檔案範圍備份，而無法複製 (RC=254)。 如果您要重試作業，就需要在發出呼叫的應用程式中包含該程式設計。
- 在伺服器上進行的同步寫入作業優先於用戶端刪除重複資料。 如果已啟用同步寫入作業，則用戶端刪除重複資料不會發生。

  限制: 當已啟用用戶端刪除重複資料時，即使有定義下一個儲存區，API 也無法從伺服器已耗盡目的地儲存區上的儲存體的狀態回復。 系統會傳回停止原因碼 DSM_RS_ABORT_DESTINATION_POOL_CHANGED，同時作業會失敗。 有兩種方法可以讓您從這種狀況回復：

  1. 要求管理者，在原始檔案儲存區中加入更多暫存磁區。
  2. 在停用刪除重複資料的情況下，重試此作業。

如果要節省更多的頻寬，您可以對本端快取啟用刪除重複資料。 本端快取會使查詢不必送至 IBM Spectrum Protect 伺服器。 ENABLEDEDUPCACHE 的預設值為 NO，這樣快取就不會與伺服器不同步。 如果快取與伺服器不同步，則應用程式會重新傳送所有的資料。 如果您的應用程式可以重試失敗的交易，而且您想要使用本端快取，請在 dsm.opt (Windows) 或 dsm.sys (UNIX) 檔中將 ENABLEDEDUPCACHE 選項設為 YES。

在還原結束時，如果透過 API 還原了所有的 資料，並且用戶端已對物件刪除重複資料，就會進行端對端的摘要計算，並將它與在備份時間所計算的值相比較。 如果那些值不符，則會傳回錯誤 DSM_RC_DIGEST_VALIDATION_ERROR。 如果應用程式收到此錯誤，表示資料已毀損。 網路上發生暫時性錯誤也可能導致此錯誤，因此請重試還原或擷取。

以下是顯示刪除重複資料資訊的查詢階段作業指令範例：

```
dsmQuerySessInfo Values:
Server Information:
Server name: SERVER1
Server Host: AVI
Server port: 1500
Server date: 2009/10/6 20:48:51
Server type: Windows
Server version: 6.2.0.0
Server Archive Retention Protection : NO
Client Information:
Client node type: API Test1
Client filespace delimiter: :
Client hl & ll delimiter: \
Client compression: Client determined (3u)
Client archive delete: Client can delete archived objects
Client backup delete: Client CANNOT delete backup objects
Maximum objects in multiple object transactions: 4096
Lan free Enabled: NO
Deduplication : Client Or Server
General session info:
Node: AVI
Owner:
API Config file:
```

以下是顯示刪除重複資料資訊的查詢管理類別指令範例：

```
Policy Information:
Domain name: DEDUP
Policyset name: DEDUP
Policy activation date: 0/0/0 0:0:0
Default management class: DEDUP
Backup retention grace period: 30 days
Archive retention grace period: 365 days
Mgmt. Class 1:
Name: DEDUP
Description: dedup - values like standard
Backup CG Name: STANDARD
Frequency: 0
Ver. Data Exists: 2
Ver. Data Deleted: 1
Retain Extra Ver: 30
Retain Only Ver: 60
Copy Destination: AVIFILEPOOL
Lan free Destination: NO
Deduplicate Data: YES

Archive CG Name: STANDARD
Frequency: 10000
Retain versions: 365
Copy Destination: AVIFILEPOOL
Lan free Destination: NO
Retain Init : CREATE
Retain Minimum : 65534
Deduplicate Data: YES
```

相關參考

Deduplication 選項

將檔案排除在刪除重複資料之外

您可以選擇從刪除重複資料排除備份或保存檔案。

如果要從刪除重複資料處理程序中排除檔案，請遵循下列步驟：

1. 對要排除的物件設定 `exclude.dedup` 選項。

比方說，如果要針對 UNIX 系統排除所有刪除重複資料，請設定：

```
exclude.dedup /.../*
```

2. 如果要針對 Windows 系統排除所有刪除重複資料，請設定：

```
exclude.dedup *\...\*
```

重要: 如果某物件被傳送至刪除重複資料儲存區，則會在伺服器上進行刪除重複資料（即使該物件被排除在用戶端刪除重複資料之外也一樣）。

併入檔案以進行刪除重複資料

您可以選擇併入備份或保存檔案以進行刪除重複資料。

如果要修正要併入的檔案清單，`include.dedup` 選項可以與 `exclude.dedup` 選項合併使用。

依預設，會併入所有合格的物件進行刪除重複資料。

以下是部分的 UNIX 和 Linux 範例：

```
exclude.dedup /FS1/.../*
include.dedup /FS1/archive/*
```

以下是部分的 Windows 範例：

```
exclude.dedup E:\myfiles\...\*
include.dedup E:\myfiles\archive\*
```

伺服器端刪除重複資料

伺服器端刪除重複資料是指由伺服器所執行的刪除重複資料。

IBM Spectrum Protect 管理者可以在 **REGISTER NODE** 或 **UPDATE NODE** 伺服器指令中使用 **DEDUP** 參數指定刪除重複資料位置（用戶端或伺服器）。

在啟用刪除重複資料的儲存區（檔案儲存區）中，只會保留一個資料延伸範圍實例。 同一個資料延伸範圍的其他實例會取代為指向保留實例的指標。

如需伺服器端刪除重複資料的相關資訊，請參閱 IBM Spectrum Protect 伺服器說明文件。

應用程式失效接手

中斷導致 IBM Spectrum Protect 伺服器變得無法使用時，使用 API 的應用程式可自動失效接手到次要伺服器以進行資料回復。

用戶端和 API 在一般正式作業程序期間期間連接到的 IBM Spectrum Protect 伺服器稱為主要伺服器 *(primary server)*。當設定主要伺服器以進行節點抄寫時，該伺服器也稱為來源抄寫伺服器。可將來源抄寫伺服器上的用戶端節點資料抄寫到目標抄寫伺服器。此伺服器也稱為 次要伺服器，並且是主要伺服器失敗時用戶端自動失效接手至的伺服器。

必須對用戶端和 API 進行配置以進行自動化用戶端失效接手，並且它們必須連接到抄寫用戶端節點資料的 7.1 版（或更新版本）伺服器。API 的配置與備份保存用戶端的配置相同。

在正常作業期間，進行登入處理程序時，會自動將次要伺服器的連線資訊從主要伺服器傳送至用戶端。次要伺服器資訊會自動儲存至用戶端選項檔案。

用戶端應用程式每一次登入 IBM Spectrum Protect 伺服器時，它都會嘗試與主要伺服器聯絡。如果主要伺服器無法使用，則應用程式會使用用戶端選項檔案中的次要伺服器資訊，自動失效接手至次要伺服器。在失效接手模式中，應用程式可以查詢次要伺服器並還原或擷取抄寫的資料。

您必須將應用程式至少備份一次到主要伺服器。僅當用戶端節點已從主要伺服器抄寫到次要伺服器時，API 才能失效接手至次要伺服器以回復資料。

相關概念
自動化用戶端失效配置和使用

失效接手狀態資訊

API 提供狀態資訊，供應用程式用於確定失效接手狀態以及次要伺服器上抄寫的用戶端資料的狀態。

抄寫狀態指示是否已將最近的備份抄寫至次要伺服器。 如果 API 上最近備份作業的時間戳記與次要伺服器上備份的時間戳記相符，則抄寫狀態是最新的。如果兩個時間戳記不符，則抄寫狀態不是最新的，抄寫的資料可能已過時。

在 **qryRespFSData** 結構中 **dsmGetNextQObj** 函數呼叫的 **query filespace** 回應上會傳回下列抄寫狀態資訊：

表 *14. API* 報告的抄寫狀態資訊

狀態資訊	**類型**	**定義**
前次抄寫的開始時間	**lastReplStartDate**	前次抄寫的開始時間。
前次抄寫的結束時間	**lastReplCmpltDate**	前次抄寫的完成時間（即使失敗了也不例外）。
前次備份儲存日期（伺服器）	**lastBackOpDateFromServer**	前次在伺服器上儲存的時間戳記。
前次備份儲存日期（本端）	**lastBackOpDateFromLocal**	前次在用戶端上儲存的時間戳記。

透過 **dsmInitExOut_t** 結構中的 **bIsFailOverMode** 欄位報告的失效接手狀態。

如需 API 的結構和類型定義，請參閱第 133 頁的『附錄 B API 類型定義原始檔』。

DSM_RC_SIGNON_FAILOVER_MODE 回覆碼指示用戶端和 API 已失效接手至次要伺服器，並且正在以失效接手模式執行。

在失效接手期間登入的範例

下列範例輸出是在失效接手期間登入伺服器的範例：

```
signon
Doing signon for node khoyt, owner , with password khoytpass
ANS2106I Connection to primary IBM Spectrum Protect server 123.45.6.78 failed

ANS2107I Attempting to connect to secondary server TARGET at 123.45.6.79 : 1501

ANS2108I Connected to secondary server TARGET.
Handle on return = 1

************************************************************
After dsmInitEx:
Server TARGET ver/rel/lev 7/1/0/0
userNameAuthorities      : Owner
Replication Server name  : TARGET
Home Server name         : MINE
Connected to replication server
************************************************************
```

查詢階段作業指令的範例

下列範例輸出是 **query session** 指令的範例，顯示了次要（抄寫）伺服器資訊：

```
query session
dsmQuerySessInfo Values:
  Server Information:
    Server name   : TARGET
    Server Host   : 123.45.6.79
    Server port: 1500
    Server date   : 2013/5/21  14:13:32
    Server type: Windows
    Server version: 7.1.0.0
    Server Archive Retention Protection : NO
  Replication Server Infomation
    Home Server name        : MINE
    Replication Server name : TARGET
       Host                 : 123.45.6.79
       Port: 1501
    Fail over status        : Connected to replication server
  Client Information:
    Client node type          : Unix
    Client filespace delimiter: /
    Client hl & ll delimiter  : /
    Client compression: Client determined (3u)
    Client archive delete: Client can delete archived objects
    Client backup delete: Client CANNOT delete backup objects
    Maximum objects in multiple object transactions: 4096
    Lan free Enabled: NO
    Deduplication             : Server Only
  General session info:
    Node           : KHOYT
    Access Node    :
    Owner:
    API Config file:
  Policy Information:
    Domain name                   : STANDARD
    Policyset name                : STANDARD
    Policy activation date: 0/0/0 0:0:0
    Default management class      : STANDARD
    Backup retention grace period: 30 days
    Archive retention grace period: 365 days
```

查詢檔案空間指令的範例

下列範例輸出是 **query filespace** 指令的範例，顯示了次要伺服器上檔案空間的抄寫狀態：

```
filespace query
Filespace pattern to query:*
Are the above responses correct (y/n/q)?

Filespace Name  Type            Occupancy  Capacity   Start         End
--------------- --------------- ---------- ---------- ------------- -------------
/fs             API:Sample             100        300   0/0/0 0:0:0   0/0/0 0:0:0

     Start of last Replication :  2013/5/21 21:3:2
     End of last Replication   :  2013/5/21 21:3:3
                                  Server                    Local
     Last backup store date    :  2013/5/21 21:18:25        2013/5/21 21:18:25
     Last archive store date   :  0/0/0 0:0:0               0/0/0 0:0:0
     Last HSM store date       :  0/0/0 0:0:0               0/0/0 0:0:0
     FSINFO : Sample API FS Info
```

相關參考

第 88 頁的『dsmGetNextQObj』

dsmGetNextQObj 函數呼叫從先前的 **dsmBeginQuery** 呼叫取得下一個查詢回應，並將回應置於呼叫者緩衝區中。

備份和保存流程圖範例

API 的設計是針對簡易邏輯流程，並且在應用程式用戶端的多種狀態間清除轉移。 此清除狀態轉移會抓取邏輯流程以及開發循環早期的程式錯誤，而大幅增進系統的品質和可靠性。

例如，除非已啟動交易而且先針對您所備份的物件進行了 **dsmBindMC** 呼叫，否則您不能進行 **dsmSendObj** 呼叫。

第 47 頁的圖 12 會顯示在交易中執行備份或保存作業的狀態圖。 從"傳送物件"指向 **dsmEndTxn** 的箭頭是指出在呼叫 **dsmSendObj** 或 **dsmSendData** 後，您可以啟動 **dsmEndTxn** 呼叫。 如果傳送物件時發生錯誤狀況而您想要停止整個作業，則您可以這麼做。 在此狀況中，您必須使用 DSM_VOTE_ABORT 的票選機制。 但是在一般情況中，請在結束交易前呼叫 **dsmEndSendObj** 。

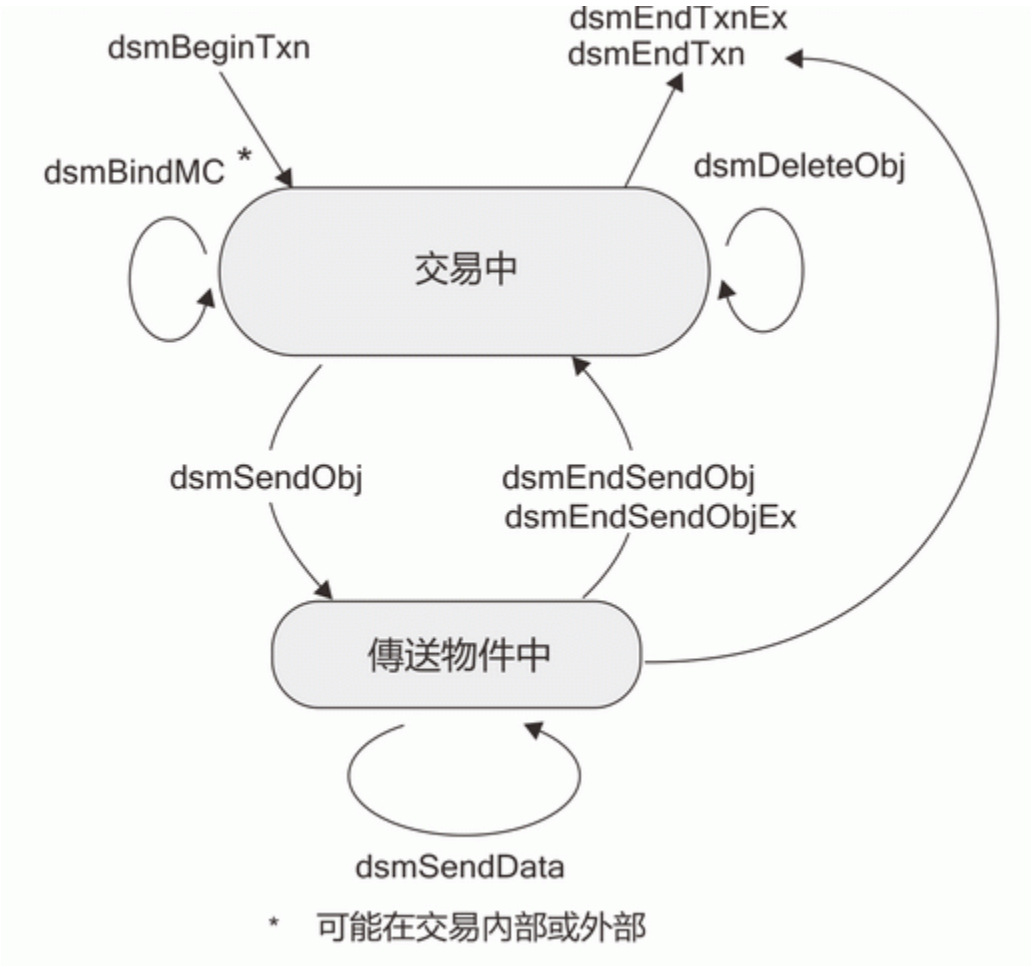


圖 *12.* 備份和保存作業的狀態圖

第 48 頁的圖 13 會顯示在交易中執行備份或保存作業的流程圖。

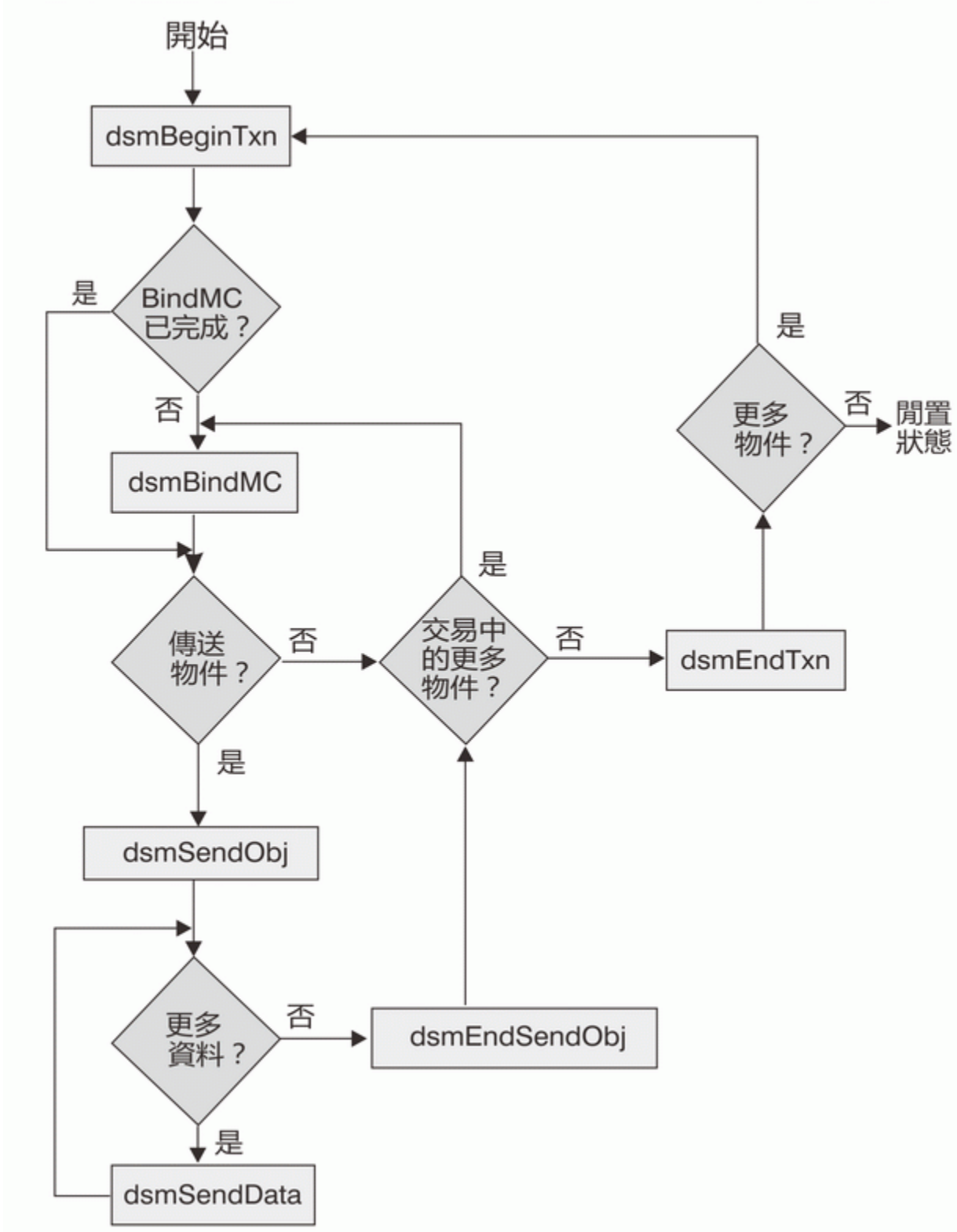


圖 *13.* 備份和保存作業的流程圖

這兩個圖解的主要特性在於異動內進行的下列 API 呼叫之間的迴圈：

- **dsmBindMC**
- **dsmSendObj**

- **dsmSendData**
- **dsmEndSendObj**

dsmBindMC 呼叫相當特殊，而您可在交易界限以內或以外啟動它。 您也可以在需要時從不同的交易啟動它。 **dsmBindMC** 呼叫的唯一需求是必須在備份或保存物件前執行它。 如果您正在備份或保存的物件並未結合管理類別，則 **dsmSendObj** 會傳回錯誤碼。 在此狀況中，呼叫 **dsmEndTxn** 以便結束交易（此錯誤狀況不會顯示在流程圖中）。

流程圖會說明應用程式如何使用多重物件交易。 它會顯示在何處放置決策點，以便決定在交易中傳送的物件是否適合，或是否要啟動新的交易。

將資料傳送到 IBM Spectrum Protect 儲存體的 API 函數程式碼範例

本範例示範如何使用 API 函數，將資料傳送到 IBM Spectrum Protect 儲存體。 **dsmSendObj** 呼叫會出現在 switch 陳述式中，以便根據是否為執行備份或保存作業，來呼叫不同的參數。

dsmSendData 呼叫是在迴圈中執行，而且會重複地傳送資料，直到設定了旗號，讓程式執行跳出迴圈。 完整的傳送作業是在交易中執行。

dsmSendObj 呼叫的第三個參數是包含保存說明的緩衝區。 因為備份物件沒有說明，此參數在備份物件時為空值。

第 24 頁的圖 8 所顯示的範例會顯示如何使用 **dsmBindMC** 函數呼叫。

```
if ((rc = dsmBeginTxn(dsmHandle)) )       /* API 階段作業 handle  */
{
   printf("*** dsmBeginTxn failed: ");
   rcApiOut(dsmHandle, rc);
   return;
}

/*  呼叫 dsmBindMC
 若先前尚未完成 */
objAttr.sizeEstimate.hi = 0;       /* 預估        */
objAttr.sizeEstimate.lo = 32000;   /* 物件大小    */
switch (send_type)
{
   case (Backup_Send) :
      rc = dsmSendObj
(dsmHandle,stBackup,
      NULL,&objName,&objAttr,NULL);
      break;
   case (Archive_Send) :
      archData.stVersion = sndArchiveDataVersion;
      archData.descr = desc;
      rc = dsmSendObj
(dsmHandle,stArchive,
   &archData,&objName,&objAttr,NULL);
      break;
   default : ;
}
if (rc)
{
   printf("*** dsmSendObj
 failed: ");
   rcApiOut(dsmHandle, rc);
   return;
}
done = bFalse;
while (!done)
{
   dataBlk.stVersion = DataBlkVersion;
   dataBlk.bufferLen = send_amt;
   dataBlk.numBytes = 0;
   dataBlk.bufferPtr = bkup_buff;
   rc = dsmSendData
(dsmHandle,&dataBlk);
   if (rc)
   {
      printf("*** dsmSendData
 failed: ");
      rcApiOut(dsmHandle, rc);
      done = bTrue;
   }
   /* 調整適用於下一個傳送的 dataBlk 緩衝區 */
}
rc = dsmEndSendObj
(dsmHandle);
if (rc)
{
   printf("*** dsmEndSendObj
 failed: ");
   rcApiOut(dsmHandle, rc);
}
txn_reason = 0;
rc = dsmEndTxn
(dsmHandle,           /* API 階段作業 handle       */
            DSM_VOTE_COMMIT,      /* Commit 異動       */
            &txn_reason);         /* 若 txn 中止的原因    */
if (rc || txn_reason)
{
   printf("*** dsmEndTxn
 failed: rc = ");
   rcApiOut(dsmHandle, rc);
   printf("   reason =
}
```

圖 *14.* 傳送資料到伺服器範例

檔案分組

IBM Spectrum Protect API 具有邏輯檔案分組通訊協定，可以將數個個別物件關聯在一起。您可以在伺服器上以邏輯群組方式來參考和管理這些群組。 邏輯群組需要所有群組成員和群組前導屬於伺服器上的同一個節點和檔案空間。

每個邏輯群組都有一個群組前導。 如果群組前導被刪除，則群組也就被刪除。 您不能刪除屬於群組一部分的成員。 群組中所有成員的到期與否是由群組前導決定。 比方說，如果某個成員標示為到期，只有在群組前導到期時，該成員才會到期。 然而，如果成員未被標示到期，但是群組前導已到期，則所有的成員都會到期。

檔案群組只包含備份資料，而且不能包含保存資料。 保存物件可以使用**保存說明**欄位，在應用程式要求情況下，可讓某一類型的分組更為容易。

dsmGroupHandler 呼叫會分組作業。 **dsmGroupHandler** 函數必須在交易內呼叫。 大部分的群組錯誤狀況都會在 **dsmEndTxn**l 或 **dsmEndTxnEx** 呼叫中被攔截。

dsmEndTxnEx 的 out 結構中包含新欄位 **groupLeaderObjId**。 如果在該交易中開啟群組，此欄位會包含群組前導的物件 ID。 您可以建立跨越一個以上的交易的群組。 在尚未關閉之前，不會在伺服器上確定或儲存群組。 **dsmGroupHandler** 是可以接受五種不同作業 的介面。 包括：

- DSM_GROUP_ACTION_OPEN
- DSM_GROUP_ACTION_CLOSE
- DSM_GROUP_ACTION_ADD
- DSM_GROUP_ACTION_ASSIGNTO
- DSM_GROUP_ACTION_REMOVE

第 51 頁的表 15 列出 **dsmGroupHandler** 函數呼叫動作：

表 *15. dsmGroupHanlder* 函數

動作	**說明**
OPEN	OPEN 動作會建立群組。 傳送的下一個物件會變成群組前導。 群組前導不能有內容。 第一個物件之後的所有物件都成為加入群組的成員。 如果要建立群組，請開啟群組並 傳入唯一的字串以識別群組。 此唯一識別字可允許開啟數個相同名稱的群組。 群組開啟之後，要傳送的下一個物件是群組前導。 傳送的所有其他物件都是 群組成員。
CLOSE	CLOSE 動作可確定並儲存開啟的群組。 如果要關閉群組，請傳入物件名稱和使用於開啟作業中的 唯一字串。 應用程式必須檢查是否有開啟的群組，如果需要，請關閉或刪除群組。 在群組關閉之前，並不會確定或儲存群組。 在下列情況下，CLOSE 動作會失敗： · 您嘗試關閉的群組與現存開啟群組名稱相同。 · 目前關閉的群組與將要關閉的同名新群組之間，有管理類別不相容的情形。 在此情況下，請完成下列步驟： 1. 查詢前一個關閉的群組。 2. 如果現有關閉之群組的管理類別，不同於目前開啟之群組相關聯的管理類別，請發出類型為 DSM_BACKUPD_MC 的 **dsmUpdateObject**。 這個指令會將現有群組更新為新的管理類別。 3. 發出 CLOSE 動作。
ADD	ADD 動作可附加物件至群組。 ADD 動作之後傳送的所有物件 都會指定至群組。
ASSIGNTO	ASSIGNTO 動作允許用戶端，將存在於伺服器上的物件指定給宣告的同層級群組。 此交易會設定 PEER 群組關係。 ASSIGNTO 動作類似 ADD 動作，但有下列的例外情形： · ADD 動作適用於進行中交易內的物件。 · ASSIGNTO 動作適用於伺服器上的物件。

表 *15. dsmGroupHanlder* 函數 *(*繼續*)*	
動作	**說明**
REMOVE	REMOVE 動作可移除群組中的一個成員或一份成員清單。 不能移除群組的群組前導。 群組成員必須移除之後才能被刪除。

請為群組支援使用下列的查詢類型：

- **qtBackupGroups**
- **qtOpenGroups**

qtBackupGroups 會查詢關閉的群組，而 **qtOpenGroups** 會查詢開啟的群組。 新類型的查詢緩衝區有 **groupLeaderObjId** 和 **objType** 欄位。 根據這兩個欄位值的不同會執行不同的查詢。 下表包括了一些可能的 查詢：

表 *16.* 查詢範例

groupLeaderObjId.hi	**groupLeaderObjId.lo**	**objType**	**結果**
0	0	空值	傳回所有群組前導清單
grpLdrObjId.hi	grpLdrObjId.lo	0	對已指派給指定的群組前導 (**grpLdrObjId**) 的所有群組成員傳回清單。
grpLdrObjId.hi	grpLdrObjId.lo	objType	利用 **BackQryRespEnhanced3** 對已指派給指定群組前導 (**grpLdrObjId**) 且符合物件類型 (**objType**) 的每個群組成員傳回清單。

dsmGetNextQObj 的回應結構 (**qryRespBackupData**) 包括二個欄位供群組支援用：

- **isGroupLeader**
- **isOpenGroup**

這些欄位是布林旗標。 下例範例顯示在 IBM Spectrum Protect 伺服器上建立群組、新增成員到群組以及關閉群組來確定群組。

```
dsmBeginTxn
    dsmGroupHandler (PEER, OPEN, leader, uniqueId)
    dsmBeginSendObj
        dsmEndSendObj
dsmEndTxnEx (With objId of leader)
Loop for multiple txns
{
    dsmBeginTxn
        dsmGroupHandler (PEER, ADD, member, groupLeaderObjID)
        Loop for multiple objects
        {
            dsmBeginSendObj
            Loop for data
            {
                dsmSendData
            }
            dsmEndSendObj
        }
    dsmEndTxn
}
dmBeginTxn
    dsmGroupHandler(CLOSE)
dsmEndTxn
```

圖 *15.* 用來建立群組的虛擬碼範例

如需程式碼範例，請參閱內含在 API `sampsrc` 目錄中的範例群組程式 `dsmgrp.c`。

從伺服器接收資料

應用程式用戶端可以使用還原和擷取功能，從 IBM Spectrum Protect 儲存體接收資料或指名的物件以及它們的相關資料。還原功能可存取先前備份的物件，而擷取功能可存取先前保存的物件。

限制: API 只能還原或擷取以 API 呼叫來備份或保存的物件。

還原和擷取功能都是以查詢作業為開端。 查詢會根據資料原來是被備份或保存，而傳回不同的資訊。 例如，對於備份物件的查詢會傳回物件是否為作用中或非作用中的資訊，而對於保存物件的查詢 則會傳回物件說明等資訊。 這兩種查詢都會傳回物件 ID（用於識別伺服器上的物件）。

部分物件還原或擷取

應用程式用戶端可只接收物件的部份。 這稱為部份物件還原或部份物件擷取。

小心: 如果部分還原或擷取已壓縮或加密的物件，會產生無法預期的結果。

註: 如果您將應用程式編碼為使用部分物件還原或擷取，您不能在傳送資料時壓縮資料。 若要強制執行此動作，請將 *ObjAttr.objCompressed* 設定為 *bTrue*。

如果要執行部分物件還原或擷取，請將下列兩個資料欄位關聯到每個物件的 **GetList** 項目：

offset
在傳回資料時的物件位元組偏移。

length
傳回的物件位元組數目。

使用 DSM_MAX_PARTIAL_GET_OBJ，決定針對特定的 **dsmBeginGetData** 清單，可執行部分物件還原或擷取的最大物件數。

以下使用於 **dsmBeginGetData** 呼叫的資料欄位可決定要還原或擷取物件的哪個部份：

- 如果偏移和長度為零，則會從 IBM Spectrum Protect 儲存體還原或擷取整個物件。
- 如果偏移大於零但長度為零，則會從偏移至結尾的部分來還原或擷取物件。
- 如果長度大於零，則只會從偏移開始的指定長度部分來還原或擷取物件。

還原或擷取資料

在執行查詢且建立與 IBM Spectrum Protect 伺服器的階段作業之後，您可以執行程序來還原或擷取資料。

程序

若要還原或擷取資料，請完成下列步驟：

1. 查詢 IBM Spectrum Protect 伺服器是否有備份或保存資料。
2. 決定要從伺服器還原或擷取的物件。
3. 在 **Restore Order** 欄位中將物件排序。
4. 傳送 **dsmBeginGetData** 呼叫以及您要存取的物件清單。
5. 傳送 **dsmGetObj** 呼叫以便從系統取得每個物件。 您可能需對每個物件進行多重 **dsmGetData** 呼叫，以便取得所有的物件相關資料。 在取得物件的所有資料後傳送 **dsmEndGetObj** 呼叫。
6. 在取得所有物件的所有資料或結束接收作業後，傳送 **dsmEndGetData** 呼叫。

查詢伺服器

在開始任何還原或擷取作業之前，請先查詢 IBM Spectrum Protect 伺服器以判斷可以從儲存體接收什麼物件。

如果要傳送查詢，應用程式必須輸入 **dsmBeginQuery** 呼叫的參數清單和結構。 結構必須包含查詢檢查的檔案空間，以及高階和低階名稱欄位的型樣相符項目。 如果在起始設定階段作業時擁有者名稱為空值，則您

不需要指定擁有者欄位。 但是，在階段作業是以明確的擁有者名稱來進行起始設定時，只會傳回與該擁有者名稱相關聯的物件。

時間點 **BackupQuery** 查詢提供特定時間的系統 Snapshot。 您可以指定有效的日期來查詢備份至該時間為止的所有檔案。 時間點會置換物件狀態，所以不論物件是否有日期較新的作用中備份，仍會傳回先前的非作用中副本。 如需相關資訊，請參閱下列範例：pitDate。

查詢會傳回與物件一起儲存的所有資訊，除了下表中的資訊之外。

表 *17.* 查詢伺服器傳回資訊

欄位	說明
copyId	copyIdHi 和 copyIdLo 值提供一個 8 位元組數字，此數字可唯一識別 IBM Spectrum Protect 儲存體中這個節點的這個物件。 請使用這個 ID 來對儲存體要求特定的物件，以便進行還原或擷取處理。
restoreOrderExt	restoreOrderExt 值提供一種儘量以最有效的方式，從 IBM Spectrum Protect 儲存體接收物件的機制。 請將這個值中要還原的物件排序，以確保磁帶只會裝載一次，而且是從前面開始往後面讀取。

您必須保留部分或全部查詢資訊，以供日後處理之用。 因為實際還原作業需要 copyId 和 restoreOrderExt 欄位，請保留它們。 您還必須保留開啟資料檔或識別目的地時所需要的任何其他資訊。

呼叫 **dsmEndQuery** 以便完成查詢作業。

依照還原順序來選取及排序物件

在執行備份或保存查詢之後，應用程式用戶端必須判斷要還原或擷取哪些物件。

然後依照遞增順序（由低至高）排序物件。 此排序對於還原作業的效能非常重要。 在 **restoreOrderExt** 欄位中排序物件可確定以 最有效率的次序，從伺服器讀取資料。

磁碟上的所有資料都會先被還原，之後會還原需要容體裝載（例如磁帶）的媒體類別上的資料。 **restoreOrderExt** 欄位 也可確保在讀取磁帶上的資料時，是從磁帶的開頭處理至結尾。

在 **restoreOrderExt** 欄位中適當地排序可避免 重複裝載磁帶和不必要的磁帶倒轉。

restoreOrderExt.top 欄位中的非零值與 IBM Spectrum Protect 伺服器上的唯一序列存取裝置有相互關係。 因為一個階段作業/裝載點一次只能使用一個序列存取裝置，因此如果應用程式使用多個階段作業，則應該確定它沒有使用相同的 **restoreOrderExt**.top 值來同時進行還原。 否則，雖然第一個階段作業可以存取物件，但其他階段作業必須等到第一個階段作業終止且裝置可用之後，才能存取物件。

下列範例顯示如何使用 **Restore Order** 欄位來排序物件。

圖 *16.* 使用 *Restore Order* 欄位來排序物件

```
typedef struct {
dsStruct64_t      objId;
dsUint160_t      restoreOrderExt;

}  SortOrder;                /* 適用於排序的已使用結構 */

=====================================================================
/* 這個程式碼從這裡開始排序 */
dsmQueryType      queryType;
qryBackupData     queryBuffer;
DataBlk           qDataBlkArea;
qryRespBackupData qbDataArea;
dsInt16_t   rc;
dsBool_t  done = bFalse;
int i = 0;
int qry_item;
SortOrder  sortorder[100]; /* 現在只可以最多完成 100 種
                                        項目的排序。設定適當的
                                        陣列大小來符合您的需要 */
/*--------------------------------------------------------------------+
```

```
| 附註：請確定已適當地起始設定了 queryType、queryBuffer、          |
| qDataBlkAre 和 qbDataArea。
|
-------------------------------------------------------------------*/

   qDataBlkArea.bufferPtf = (char*) &qbDataArea;

   rc = dsmBeginQuery(dsmHandle, queryType, (void *) &queryBuffer);

   /*---------------------------------------+
   | Make sure to check rc from dsmBeginQuery
   +---------------------------------------*/
   while (!done)
   {
      rc = dsmGetNextQObj(dsmHandle, &qDataBlkArea);
 if ((rc == DSM_RC_MORE_DATA) ||
          (rc == DSM_RC_FINISHED))
           &&( qDataBlkArea.numBytes))
      {
         /*****************************************/
         /* 轉送 restoreOrderExt 和 objId          */
         /*****************************************/
         sortorder[i].restoreOrderExt = qbDataArea.restoreOrderExt;
         sortorder[i].objId = qbDataArea.objId;

      } /*  if ((rc == DSM_RC_MORE_DATA) || (rc == DSM_RC_FINISHED)) */
      else
      {
          done = bTrue;
          /***************************/
          /* 採取適當的動作。 */
          /***************************/
      }

      i++;
      qry_item++;

   } /* while (!完成) */
   rc = dsmEndQuery(dsmHandle);
  /*檢查 rc  */
   /*****************************************************/
   /* 使用 qsort 排序陣列。在呼叫後，                    */
   /* restoreOrderExt 欄位會排序 sortorder              */
   /*****************************************************/

   qsort(sortorder, qry_item, sizeof(SortOrder), SortRestoreOrder);

 /*-------------------------------------------------------------------+
| 附註：請確定取回排序的物件 ID 並且將它們以所要的資料結構來儲存。 |
-------------------------------------------------------------------*/

/*-------------------------------------------------------------------+
| int SortRestoreOrder(SortOrder *a, SortOrder *b)
|
| 此函數會比較兩種結構的 restoreOrder 欄位。
| if (a > b)
|   return(GREATERTHAN);
|| if (a < b)
|   return(LESSTHAN);
|| if (a == b)
|   return(EQUAL);
|+-------------------------------------------------------------------*/
int SortRestoreOrder(SortOrder *a, SortOrder *b)
{
    if (a->restoreOrderExt.top > b->restoreOrderExt.top)
       return(GREATERTHAN);
    else if (a->restoreOrderExt.top < b->restoreOrderExt.top)
       return(LESSTHAN);
    else if (a->restoreOrderExt.hi_hi > b->restoreOrderExt.hi_hi)
       return(GREATERTHAN);
    else if (a->restoreOrderExt.hi_hi < b->restoreOrderExt.hi_hi)
       return(LESSTHAN);
    else if (a->restoreOrderExt.hi_lo > b->restoreOrderExt.hi_lo)
       return(GREATERTHAN);
    else if (a->restoreOrderExt.hi_lo < b->restoreOrderExt.hi_lo)
       return(LESSTHAN);
     else if (a->restoreOrderExt.lo_hi > b->restoreOrderExt.lo_hi)
       return(GREATERTHAN);
    else if (a->restoreOrderExt.lo_hi < b->restoreOrderExt.lo_hi)
       return(LESSTHAN);
    else if (a->restoreOrderExt.lo_lo > b->restoreOrderExt.lo_lo)
```

```
        return(GREATERTHAN);
    else if (a->restoreOrderExt.lo_lo < b->restoreOrderExt.lo_lo)
        return(LESSTHAN);
    else
        return(EQUAL);
}
```

啟動 dsmBeginGetData 呼叫

在選取和排序要接收的物件之後，請將它們提交到 IBM Spectrum Protect 以執行還原或擷取作業。 **dsmBeginGetData** 呼叫會開始還原或擷取作業。 物件會以您要求的次序傳回至應用程式用戶端。

請在這些呼叫中 完成這兩個參數的資訊：

mountWait
此參數會告訴伺服器應用程式用戶端是否會等待裝載離線媒體，以便取得物件的資料，或在處理還原或擷取作業期間，是否應略過該物件。

dsmGetObjListP
此參數是一種資料結構，其中包含了 **objId** 欄位，這是一份所有已還原或擷取的物件 ID 清單。 每個 **objId** 都會與 **partialObjData** 結構相關聯，而該結構會說明是要擷取完整 **objId** 或只擷取物件的特定區段。

每個 **objId** 的 長度為 8 位元組，所以單一還原或擷取要求可以包含 數千個物件。 單一呼叫中要求物件的數目限制 在 DSM_MAX_GET_OBJ 或 DSM_MAX_PARTIAL_GET_OBJ。

接收要還原或擷取的每一個物件

傳送 **dsmBeginGetData** 呼叫之後，您就可以執行程序來接收從伺服器傳送的每個物件。

關於這項作業

DSM_RC_MORE_DATA DSM_RC_MORE_DATA 回覆碼表示已傳回了緩衝區，而且您應再次呼叫 **dsmGetData**。請檢查 **DataBlk.numBytes** 以了解傳回位元組的實際數量。

當您取得物件的所有資料時，您必須傳送 **dsmEndGetObj** 呼叫。 如果收到了多個物件，請再次傳送 **dsmGetObj** 呼叫。

如果您需要停止程序，例如您想要捨棄所有物件的還原串流內尚未收到之任何剩餘的資料，請傳送 **dsmEndGetData** 呼叫。此呼叫會清除從伺服器傳送至用戶端的資料。但是，使用此方法可能會需要一些時間來完成。 如果您要結束還原作業，請使用 **dsmTerminate** 來關閉階段作業。

程序

1. 傳送 **dsmGetObj** 呼叫以識別您所要求而來自資料串流的物件，並且取得與物件相關的第一個資料區塊。
2. 請在需要時傳送更多的 **dsmGetData** 呼叫，以便取得其餘的物件資料。

還原和擷取流程圖範例

狀態圖和流程圖可用來說明如何執行還原或擷取作業。

從"取得物件中"指向 **dsmEndGetData** 的箭頭 是表示您可以在呼叫 **dsmGetObj** 或 **dsmGetData** 後傳送 **dsmEndGetData** 呼叫。 如果在從 IBM Spectrum Protect 儲存體取得物件時發生錯誤狀況而且您想要停止作業的話，您可能需要這樣做。 但是在一般情況中，請先呼叫 **dsmEndGetObj** 。

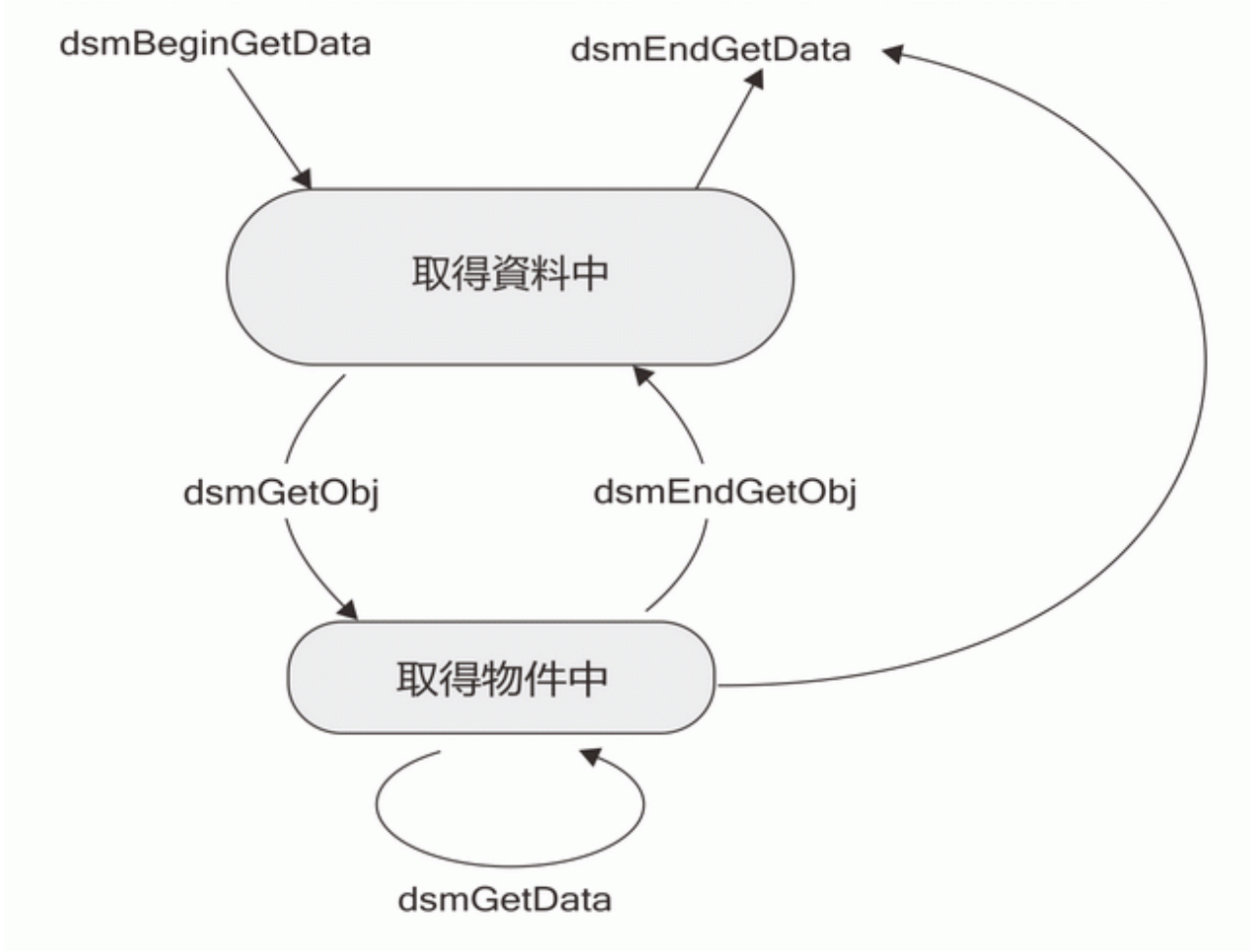


圖 *17.* 還原和擷取作業的狀態圖

第 58 頁的圖 18 顯示了執行還原或擷取作業的流程圖。

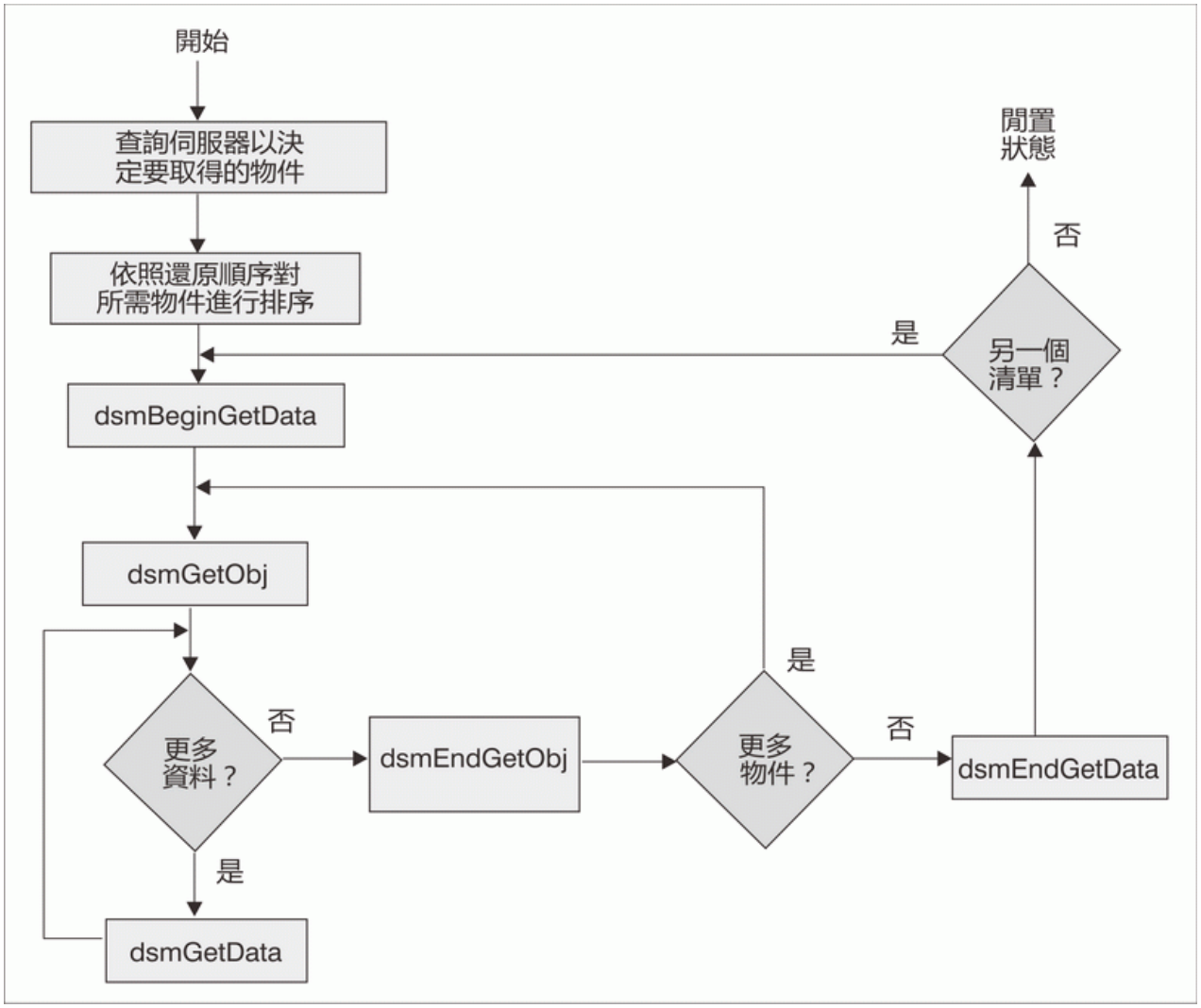


圖 *18.* 還原和擷取作業的流程圖

從伺服器接收資料的程式碼範例

本範例示範如何使用此 API 函數，從 IBM Spectrum Protect 儲存體擷取資料。

dsmBeginGetData 函數呼叫可在 switch 陳述式中出現，因此可根據執行還原或擷取作業，來呼叫不同的參數。 **dsmGetData** 函數呼叫是在迴圈中執行，而且會重複地從伺服器取得資料，直到設定了旗號，讓程式執行跳出迴圈。

圖 *19.* 從伺服器接收資料範例

```
/* 呼叫 dsmBeginQuery 及建立物件鏈結的物件來還原。 */
/* 處理這個清單來建立適用於 GetData 呼叫適當的清單。 */
/*設定 getList 結構來指向清單。              */
/* 設定這個範例來執行部分物件擷取。如果要 */
/* 只擷取完整的物件，請設定：                          */
/*       getList.stVersion = dsmGetListVersion;                    */
/*       getList.partialObjData = 空值；                          */
dsmGetList getList;
getList.stVersion = dsmGetListPORVersion;   /* 結構版本      */
getList.numObjId  = 項目；                /* 清單中項目數量 */
getList.objId     = (ObjID *)rest_ibuff;
                                  /* 來還原的物件 ID 清單 */
getList.partialObjData = (PartialObjData *) part_ibuff;
                                    /* 部分物件資料清單 */
switch(get_type)
{
   case (Restore_Get) :
```

```
     rc = dsmBeginGetData
(dsmHandle,bFalse,gtBackup,&getList);
     break;
   case (Retrieve_Get) :
     rc = dsmBeginGetData
(dsmHandle,bFalse,gtArchive,&getList);
     break;
   default : ;
}
if (rc)
{
   printf("*** dsmBeginGetData
 failed: ");
   rcApiOut(dsmHandle, rc);
   return rc;
}
/* 從清單來取得每一個物件及確認這是否已經在  */
/* 伺服器上。如果是的話，起始設定適用於資料驗證檢查 */
/* 的物件屬性結構。完成後，請呼叫 dsmGetObj
。                */
 rc = dsmGetObj
(dsmHandle,objId,&dataBlk);
done = bFalse;
while(!done)
{
   if (   (rc == DSM_RC_MORE_DATA)
       || (rc == DSM_RC_FINISHED))
   {
      if (rc == DSM_RC_MORE_DATA)
      {
         dataBlk.numBytes = 0;
         rc = dsmGetData
(dsmHandle,&dataBlk);
      }
      else
         done = bTrue;
   }
   else
   {
      printf("*** dsmGetObj
 or dsmGetData
 failed: ");
      rcApiOut(dsmHandle, rc);
      done = bTrue;
   }
} /* while */
rc = dsmEndGetObj
(dsmHandle);
/* 檢查 rc 從 dsmEndGetObj */
/* 檢查 rc 從 dsmEndGetData */
rc = dsmEndGetData
(dsmHandle);
return 0;
```

更新和刪除伺服器上的物件

API 應用程式可使用 **dsmUpdateObj** 或 **dsmUpdateObjEx** 函數呼叫，來更新保存或備份的物件。 您只能在階段作業狀態中使用任一呼叫，以便一次更新一個物件。 使用 **dsmUpdateObjEx** 來更新若干包含相同名稱的任何保存物件。

若要選取保存物件，請將 **dsmSendType** 函數呼叫設定為 **stArchive**。

- 使用 **dsmUpdateObj** 時，只會更新具備指定名稱的最新保存物件。
- 使用 **dsmUpdateObjEx** 時，可以藉由指定適當的物件 ID 來更新任何保存物件。

對於保存物件，應用程式可更新以下欄位：

- 說明
- 物件資訊
- 擁有者

若要選取備份物件，請將 **dsmSendType** 設為 **stBackup**。 對於備份物件，則只能更新作用中副本。

對於備份物件，應用程式可更新以下欄位：

- 管理類別
- 物件資訊
- 擁有者

從伺服器刪除物件

API 應用程式可執行呼叫以便刪除保存的物件，或關閉備份的物件。 刪除保 存的物件需視管理者登錄節點時所給予的節點授權而定。管理者可指定能刪除保存物件的節點。

使用 **dsmDeleteObj** 函數呼叫，來刪除保存物件和關閉備份物件。 使用此 **delType** 可從伺服器上移除備份物件。 這是根據 **objID** 而定，它會從伺服器資料庫刪除物件。 只有物件的擁有者可以刪除物件。 您可以刪除物件的任何版本（作用中或非作用中）。 伺服器會重新調解這些版本。 如果您刪除物件的作用中版本，則第一個非作用中版本會變成作用中。 如果您刪除物件的非作用中版本，則所有舊的版本都會往前推進。 節點必須登錄有 **backDel** 許可權。

當系統執行下一次的物件到期循環時，儲存體中的保存物件會被標示刪除。 在從伺服器刪除保存物件時，您會無法擷取它。

當您停用伺服器中的備份物件時，物件會從作用中狀態變為非作用中狀態。 根據所指定的管理類別，這些狀態有相關聯的不同保留原則。

與 **dsmSendObj** 呼叫相似的是，**dsmDeleteObj** 呼叫會在交易界限內傳送。 第 47 頁的圖 12 中的狀態圖解會顯示 **dsmDeleteObj** 呼叫前需先呼叫 **dsmBeginTxn**，而接著要呼叫 **dsmEndTxn** 。

記載事件

API 應用程式可將事件訊息紀錄至中央位置。 應用程式可以直接記載到 IBM Spectrum Protect 伺服器、本端機器或兩者。 **dsmLogEventEx** 函數呼叫是在階段作業中執行。 如果要檢視記載在伺服器上的訊息，請透過管理用戶端來使用 **query actlog** 指令。

如果應用程式寫入許多用戶端訊息到用戶端日誌 dsmLogType（是 logLocal 或 logBoth），請使用 IBM Spectrum Protect 用戶端選項 errorlogretention 來刪改用戶端錯誤日誌檔。

如需 IBM Spectrum Protect 日誌的相關資訊，請參閱 IBM Spectrum Protect 伺服器說明文件。

IBM Spectrum Protect API 狀態圖摘要

在檢閱有關使用 IBM Spectrum Protect API 建立您自己的應用程式的所有注意事項之後，請檢閱整個應用程式的這個狀態圖摘要。

第 61 頁的圖 20 包含了 API 的狀態圖。 它包含了所有先前顯示的狀態圖，以及數個先前未顯示的其他呼叫。

此圖的重點包括：

- 隨時呼叫 **dsmQueryApiVersionEx**。 它沒有相關的狀態。 請參閱第 13 頁的圖 1 以取得範例。
- 只能先呼叫 **dsmQueryCliOptions**，才能呼叫 **dsmInitEx**。
- 使用 **dsmRegisterFS** 、**dsmUpdateFS** 和 **dsmDeleteFS** 來管理檔案空間。 您可從閒置的階段作業狀態中進行這些呼叫。 使用 **dsmBeginQuery** 呼叫查詢檔案空間。 如需檔案空間呼叫的相關資訊，請參閱第 20 頁的『管理檔案空間』。
- 請從閒置的階段作業狀態或傳送物件交易狀態中傳送 **dsmBindMC** 呼叫。 請參閱第 24 頁的圖 8 中的範例。
- 從閒置的階段作業狀態傳送 **dsmChangePW** 呼叫。

  註: 如果 **dsmInitEx** 呼叫傳回密碼到期回覆碼，則必須在啟動有效的階段作業前，進行 **dsmChangePW** 呼叫。 請參閱 第 17 頁的圖 4 以取得使用 **dsmChangePW** 的範例。
- 如果呼叫傳回錯誤，狀態會維持相同。 例如，如果 **dsmGetObj** 傳回錯誤，狀態會保持「取得資料中」，而呼叫 **dsmEndGetObj** 是呼叫順序錯誤。

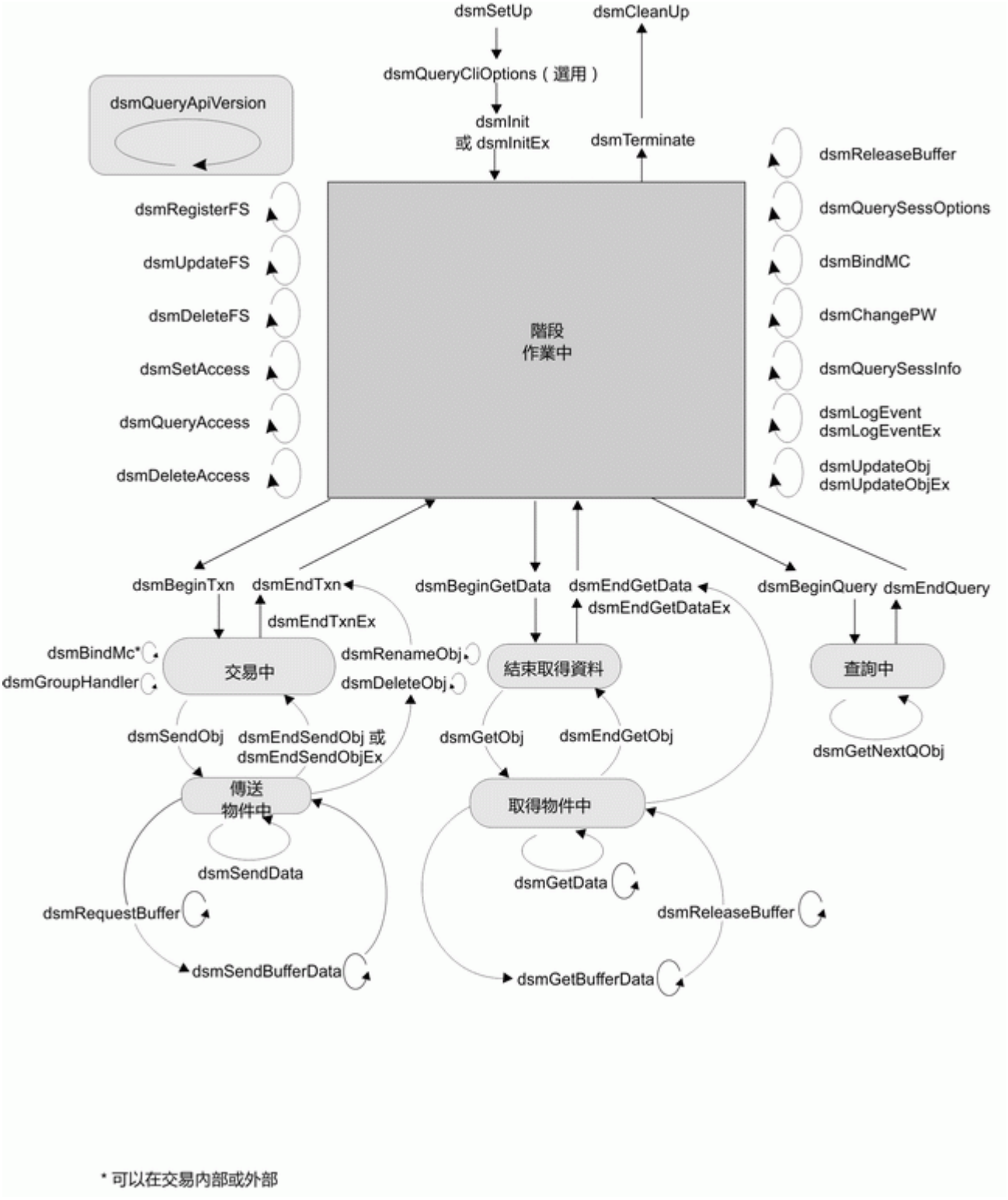


圖 *20. API* 的摘要狀態圖

第 4 章 瞭解交互作業能力

API 有二種類型的交互作業能力：備份保存用戶端與 API 應用程式之間，以及不同的作業系統之間。

備份保存用戶端交互作業能力

備份保存指令行可以存取 API 物件以提供有限的互運性。 您只能從備份保存指令行用戶端檢視及存取 API 物件，而無法從任何圖形介面來檢視或存取它們。 備份保存指令行用戶端只能還原檔案的內容，因此您應只在援救類型的作業中使用它。

提供下列指令行動作：

- Delete archive
- Delete filespace
- 查詢
- 還原
- 擷取
- Set access

路徑資訊是備份保存用戶端物件的實際目錄。 相對地，API 物件路徑資訊和現有的目錄可能沒有任何關係：路徑可能完全是自行設定的。 互運性不會改變這些物件類型的這個方面。 為了順利使用此 特性，請遵循限制和慣例。

注意事項:

1. 備份保存用戶端和儲存於保留保護伺服器上的 API 物件之間，沒有交互作業能力。
2. 您無法使用備份保存用戶端 GUI，來存取使用 API 用戶端所儲存的檔案。 您只能使用指令行來存取這些檔案。

命名 API 物件

建立 API 物件名稱的一致命名慣例。 命名慣例必須提供檔案空間名稱、高階限定元和低階限定元。 檔案空間名稱和高階限定元可以參照實際目錄名稱。 每一個物件名稱都可以包含一個以上適用於低階限定元的目錄名稱。

為了方便起見，請使用不加上目錄資訊作為字首的物件名稱，來作為低階限定元。 如需相關資訊，請參閱第 18 頁的『物件名稱和 ID』。

從 API 或備份保存指令行參照檔案空間名稱時，該名稱必須是完整名稱。 例如，在 UNIX 或 Linux 作業系統上，請登錄下列檔案空間：

- /a
- /a/b

當您參照 /a 時，會顯示只與檔案空間 /a 相關的物件。 如果要檢視與 /a/b 相關的物件，您必須指定 /a/b 作為檔案空間名稱。

在登錄這兩個檔案空間之後，如果您將物件 b 備份到檔案空間 /a，那麼 /a/b 的查詢會繼續顯示只與檔案空間 /a/b 相關的物件。

當您嘗試以 API 查詢或刪除檔案空間時，檔案空間參照不受這項限制的約束。 在這兩種情況下，如果您使用萬用字元，那麼檔案空間名稱不必是完整名稱。 例如，/a* 會參照 /a 和 /a/b。

提示: 如果交互作業能力對您而言很重要，請避免採用重疊的檔案空間名稱。

在 Windows 系統上，當您從備份保存指令行介面存取 API 物件時，請以大括弧 { } 括住物件的檔案空間名稱。當您登錄或參照檔案空間名稱時，Windows 作業系統會自動將名稱轉為大寫字母。 不過，這項自動功能並不會發生在其餘的物件名稱規格。如果您要完整的互運性，當您備份 API 物件時，請在 應用程式中使用大寫字母的高階限定元和低階限定元。 如果您的應用程式不在將物件傳送到伺服器之前將高階限定元（目錄

名稱）和低階限定元（檔名）處理成大寫字母，您將無法透過備份保存用戶端直接使用名稱來存取這些物件。

例如，如果某個物件以 {"FileSpacename"}\TEST\MYDIRNAME\file.txt 儲存在伺服器上，您無法直接還原或查詢 file.txt 物件，因為應用程式未在將該檔案複製到伺服器之前將檔名處理成大寫字母。操作這些物件的唯一方法是使用萬用字元。 例如，若要查詢 \TEST\MYDIRNAME\file.txt，備份保存用戶端使用者必須將萬用字元用於在傳送到伺服器之前未處理成使用大寫字母的物件名稱的所有部分。必須使用下列指令來查詢此 file.txt 檔案：

```
dsmc query backup {"FileSpaceName"}\TEST\MYDIRNAME\*
```

如果任何其他限定元也以小寫文字儲存，則也必須使用萬用字元來查詢這些限定元。例如，若要查詢儲存為 {"FileSpaceName"}\TEST\mydirname\file.txt 的物件，請使用下列指令：

```
dsmc query backup {"FileSpaceName"}\TEST\*\*
```

以下的範例說明這些概念。 在 Windows 和 UNIX 或 Linux 環境中，您不需要指定完整的高階或低階限定元。 然而，如果您沒有指定完整的限定元，就必須使用萬用字元。

平台	範例
Windows	如果要查詢 MYFS 檔案空間中的所有備份檔，請輸入下列字串： `dsmc q ba "{MYFS}\*\*"` 每一個高階和低階限定元至少必須使用一個星號 (*)。
UNIX 或 Linux	如果要查詢 /A 檔案空間中的所有備份檔，請輸入下列字串： `dsmc q ba "/A/*/*"` 每一個高階和低階限定元至少必須使用一個星號 (*)。

您可以和 API 一起使用的備份保存用戶端指令

您可以在應用程式內使用備份保存用戶端指令的子集。 例如，您可以檢視及管理其他使用者在同一個節點或不同的節點上所擁有的物件。

如果要檢視及管理其他使用者在同一個節點或不同的節點上所擁有的物件，請執行下列步驟：

1. 以 **set access** 指令提供存取權。
2. 指定擁有者和節點。 在備份保存指令行使用 *fromowner* 和 *fromnode* 選項指定擁有者和節點。 例如：

```
dsmc q ba "/A/*/*"  -fromowner=other_owner  -fromnode=other_node
```

第 64 頁的表 18 說明您可以和 API 物件一起使用的指令。

表 *18.* 您可以和 *API* 物件一起使用的備份保存用戶端指令

指令	說明
Delete Archive	現行使用者所擁有的保存檔可以刪除。 set access 指令設定對這個指令沒有影響。
Delete Filespace	**delete filespace** 指令會影響 API 物件。

表 18. 您可以和 *API* 物件一起使用的備份保存用戶端指令 (繼續)

指令	說明
查詢	從備份保存指令行中，您可以查詢已備份和保存的 API 物件，以及其他使用者所擁有的物件，或是存在其他節點的物件。請參閱第 63 頁的『命名 API 物件』以取得有關查詢 API 物件的資訊。 您可以使用現有的 *–fromowner* 選項，查詢不同使用者所擁有且已指定 set access 許可權的物件。 您可以使用現有的 *–fromnode* 選項，查詢不同節點上且已指定 set access 許可權的物件。如需相關資訊，請參閱第 96 頁的『 dsmInitEx 』。
還原 **擷取**	**註:** 只有在異常情況下，才使用這些指令。如果加密金鑰是已知的或儲存在密碼檔中，則可還原或擷取使用應用程式管理的金鑰所加密的 API 物件。以透通加密方式所加密的 API 物件則無法透過備份保存用戶端來還原或擷取。 這些指令會以位元檔傳回資料，而這些檔案是使用預設檔案屬性所建立。您可以還原或擷取其他使用者所擁有的 API 物件，或者來自不同節點的 API 物件。 set access 指令會決定哪些物件合格。
設定存取權	**set access** 指令可允許使用者管理另一個使用者所擁有的 API 物件，或是來自另一個節點的 API 物件。

範例

```
dsmc query backup f:\*\*
```

```
          大小      備份日期                管理類別         A/I 檔案
          ----      ----------              ----------       --- ----
            1  B   11/14/2018 08:22:24       DEFAULT           A  \\viola\f$\dir1\test
API         1  B   11/14/2018 08:21:41       DEFAULT           A  \\viola\f$\dir1\test
```

```
dsmc query backup "/home/*/*"
```

```
          大小      備份日期                管理類別         A/I 檔案
          ----      ----------              ----------       --- ----
            1  B   11/15/2018 08:22:24       DEFAULT           A  /home/user1/test
API         1  B   11/15/2018 08:21:41       DEFAULT           A  /home/user1/test
```

作業系統交互作業能力

IBM Spectrum Protect API 支援跨平台交互作業能力。UNIX 或 Linux 系統上的應用程式可以操作從 Windows 系統備份的檔案空間和物件。類似地，Windows 系統可以操作從 UNIX 或 Linux 系統備份的檔案空間和物件。

關於這項作業

依預設，一個 UNIX 系統中物件的名稱與另一個 UNIX 系統中物件的名稱相容。依預設，Windows 系統中物件的物件與 UNIX 系統中物件的名稱不相容。數個參數控制 IBM Spectrum Protect 檔案空間中物件的命名。如果您適當地設定應用程式，則在 Windows 系統和 UNIX 系統上執行的應用程式都可以使用物件的名稱。使用相同的參數來備份和還原物件。

限制: 使用 Unicode 的 Windows 應用程式會建立與在 UNIX 系統上執行的應用程式不相容的檔案空間。

程序

為了達到交互作業能力，請完成下列設定作業：

1. 建立一致的命名慣例。 選取 `dir` 定界字元的字元，例如正斜線 (/) 或反斜線 (\)。將目錄定界字元放置在檔案空間名稱、高階限定元和低階限定元前面。
2. 呼叫 **dsmInitEx** 時，請將 **dirDelimiter** 欄位的值設定為您所選取的目錄定界字元，並將 **bCrossPlatform** 設為 **bTrue**。

3. 當您使用 IBM Spectrum Protect 介面時，請將 **useUnicode** 旗標設為 **bFalse**。 Unicode 檔名與非 Unicode 檔名不相容。

透過用戶端節點 Proxy 支援來備份多個節點

您可以將共用儲存體的多個節點之備份作業，合併到 IBM Spectrum Protect 伺服器的共同目標節點名稱。 當執行備份的系統可能隨時間改變（如具有叢集）時，這個方法非常有用。 您也可以使用 asnodename 選項，從執行備份之系統以外的不同系統來還原資料。

關於這項作業

您可以在 **dsmInitEx** 選項字串上使用 asnodename 選項，於 IBM Spectrum Protect 伺服器的目標節點名稱下執行資料的備份、保存、還原以及擷取、查詢或刪除。 您也可以在 dsm.opt 或 dsm.sys 檔中指定 asnodename 選項。

限制: 請勿將目標節點當作傳統節點使用，尤其是在您將檔案備份到伺服器之前，進行檔案加密時，更是如此。

程序

如果要啟用這個選項，請完成下列步驟：

1. 在共用資料環境的所有節點上安裝 API 用戶端。
2. 向 IBM Spectrum Protect 伺服器登錄尚未登錄的每一個節點。 請登錄您共用資料環境中所使用的每一個代理站節點共用的共同目標節點名稱。
3. 向伺服器登錄共用資料環境中的每一個代理站節點。代理站節點名稱是用於鑑別。 在使用 asnodename 選項的情況下，不會使用該代理站節點名稱來儲存資料。
4. 要求管理者使用 **grant proxynode** 指令，將 proxy 權限授與共用環境中的所有節點，才能存取 IBM Spectrum Protect 伺服器上的目標節點名稱。指令。
5. 使用 **query proxynode** 管理用戶端指令來顯示獲授權代表另一個節點來執行用戶端作業的用戶端節點。 這個權限是由 **grant proxynode** 指令所授予。 或使用查詢類型為 **qtProxyNodeAuth** 的 **dsmQuery** 指令，來查看這個節點能夠代理哪些節點。
6. 如果應用程式是利用使用者資料加密方式（不是 TSMENCRKEY），請確定所有節點皆使用相同的加密金鑰。 您必須對共用節點環境中備份的所有檔案使用相同的加密金鑰。

相關工作

透過用戶端節點 Proxy 支援備份資料（UNIX 和 Linux 系統）

透過用戶端節點 Proxy 支援備份資料（Windows 系統）

第 5 章 搭配 Unicode 來使用 API

IBM Spectrum Protect API 支援 Unicode UCS2，它是一種固定長度的雙位元組字碼頁，具有所 有已知字碼頁的字碼點，例如日文、中文或德文。 它支援 65,535 個唯一字碼點。

限制: 此特性僅適用於 Windows。

使用 Unicode，您的應用程式可以從相同機器上以任何字集來 備份和還原檔名。 例如，在英文機器上，您可以用任何的其他語言字碼頁來備份和還原檔名。

Unicode 的使用時機

您可以簡化支援多種語言的應用程式，方法為撰寫 Unicode 應用程式以及利用 IBM Spectrum Protect Unicode 介面。

如果有符合下列的條件之一，請使用 IBM Spectrum Protect Unicode 介面：

- 如果已針對 Unicode 來編譯您的應用程式，而且在呼叫 IBM Spectrum Protect API 之前，已將該應用程式轉換為多位元組字集 (MBCS)。
- 如果您是撰寫新的應用程式，並且想要讓您的應用程式支援 Unicode。
- 如果您的應用程式使用從作業系統或使用 Unicode 的其他應用程式中所傳入的字串。

如果您不需要 Unicode，則不需要重新編譯您的應用程式。

API 繼續支援 dsm 介面。 API SDK 包含 callmtu1.c 和 callmtu2.c 範例程式，示範如何使用 Unicode API。 請使用 **makemtu** 來編譯這些程式。

設定 Unicode

如果要設定及使用 Unicode，您必須執行特定的程序，使 API 在伺服器上登錄 Unicode 檔案空間，而且該檔案空間的所有檔名都變成 Unicode 字串。

限制: 您無法將 Unicode 和非 Unicode 檔名儲存在相同的檔案空間。

1. 以 -DUNICODE 旗號來編譯程式碼。
2. 您應用程式中的所有字串必須都是 wchar 字串。
3. 遵循 tsmapitd.h 檔中的結構以及 tsmapifp.h 檔中的函數定義來呼叫 API。
4. 在 **tsmInitEx** 函數呼叫時設定 *useUnicode* 旗號為 *bTrue*。 任何新的檔案空間都會被登錄為 Unicode 檔案空間。

當您傳送資料至先前登錄的非 Unicode 檔案空間時，API 會繼續以非 Unicode 來傳送檔名。 請將伺服器上的舊檔案空間更名為 fsname_old，並起始新的 Unicode 檔案空間供新資料使用。 API 會從舊檔案空間還原非 Unicode 資料。 請使用查詢檔案空間所傳回 **tsmQryRespFSData** 結 構中的 **bIsUnicode** 欄位，來判定檔案空間是否為 Unicode。

每一個 **dsmXXX** 函數呼叫都有一個對應的 **tsmXXX** 函數呼叫。 兩者之間的差異是在於使用的結構不同。 所有的 **tsmXXX** 函數呼叫結構在使用 Unicode 旗號來編譯時都具有 dsChar_t 類型的字串。dsChar_r 對映至 wchar。 這些介面之間沒有其他的差異。

限制: 使用其中一種介面。 不要混合 **dsmXXX** 函數呼叫與 **tsmXXX** 函數呼叫介面。確保使用 IBM Spectrum Protect 結構和 IBM Spectrum Protect 版本定義。

某些常數繼續在 dsmapitd.h 檔中定義，所以在編譯時，您需要 dsmapitd.h 和 tsmapitd.h 檔案。

您可以在其他作業系統（例如 UNIX 或 Linux）使用 IBM Spectrum Protect 介面，但在這些作業系統上，dsChar_t 類型會對映至 char，因為只有 Windows 作業系統才支援 Unicode。您可以只撰寫一種變量的應用程式，使用 IBM Spectrum Protect 介面，在一種以上的作業系統中編譯。如果您撰寫新的應用程式，請使用 IBM Spectrum Protect 介面。

如果您是升級現有的應用程式：

1. 將 **dsmXXX** 函數呼叫結構和呼叫轉換為 IBM Spectrum Protect 介面。
2. 移轉現有的檔案空間。
3. 將 *useUnicode* 旗標設為 *true* 來備份新檔案空間。

註: 當您使用具備 Unicode 能力的用戶端來存取節點之後，即無法以舊版的 API 或其他作業系統的 API 連接到相同的節點名稱。 如果您的應用程式使用跨平台功能，請不要使用 Unicode 旗號。 Unicode 和 非 Unicode 作業系統之間沒有跨平台支援。

當您啟用 *useUnicode* 旗號時，所有的字串結構都會視為 Unicode 字串。 在伺服器上，只有下列欄位是真正的 Unicode：

· 檔案空間名稱

· 高階

· 低階

· 保存說明

其餘的欄位在傳送到伺服器之前都會轉換為本端字碼頁的 mbcs。 欄位（如 nodename）為 wchar 字串。 它們必須在目前的語言環境中有效。 例如，在日文機器上，您可以備份中文名稱的檔案，但是節點名稱必須是日文的有效字串。 選項檔維持目前的字碼頁。 如果您需要建立 Unicode 併入排除 清單，請配合檔名使用 *inclexcl* 選項，並建立內含 Unicode 型樣的 Unicode 檔案。

相關參考

inclexcl 選項

第 6 章 API 函數呼叫

第 69 頁的表 19 提供按字母順序排列的 API 函數呼叫清單、簡短說明，以及關於函數呼叫的更詳細資訊位置，其中包括：

Element	說明
目的	說明函數呼叫。
語法	包含函數呼叫的實際 C 程式碼。 此程式碼是從 UNIX 或 Linux 版本的 dsmapifp.h 標頭檔複製而來。. 請參閱第 171 頁的『附錄 C API 函數定義原始檔』。 此檔案在其他作業系統上略有不同。 使用 其他作業系統的應用程式設計師應檢查標頭檔 dsmapifp.h 的版本，以取得 API 定義的正確語法。
參數	說明函數呼叫中的每個參數，指出函數呼叫在使用上為輸入 (I) 或輸出 (O)。 某些參數可同時指定為輸入以及輸出 (I/O)。 本節所參考的資料類型是在 dsmapitd.h 標頭檔中定義。 請參閱第 133 頁的『附錄 B API 類型定義原始檔』。
回覆碼	包含函數呼叫的特定回覆碼清單。 任何呼叫可能出現的一般系統錯誤（例如通訊錯誤、伺服器問題以及使用者錯誤） 並不在這份清單中。 回覆碼是定義於 dsmrc.h 標頭檔。 請參閱第 123 頁的『附錄 A API 回覆碼原始檔：dsmrc.h』。

表 *19. API* 函數呼叫

函數呼叫與位置	說明
第 71 頁的『dsmBeginGetData』	對儲存體中的物件清單啟動還原或擷取作業。
第 72 頁的『dsmBeginQuery』	對 IBM Spectrum Protect 啟動查詢要求以取得資訊。
第 76 頁的『dsmBeginTxn』	啟動開始一個完整動作的一或多個交易。 可能是所有動作都成功，或都不成功。
第 77 頁的『dsmBindMC』	將管理類別結合或連結到傳送的物件。
第 78 頁的『dsmChangePW』	變更 IBM Spectrum Protect 密碼。
第 79 頁的『dsmCleanUp』	如果呼叫 **dsmSetUp**，則使用此呼叫。
第 79 頁的『dsmDeleteAccess』	刪除物件之備份版本或保存副本的現行授權規則。
第 80 頁的『dsmDeleteFS』	從儲存體刪除檔案空間。
第 80 頁的『dsmDeleteObj』	關閉備份物件，或刪除儲存體中的保存物件。
第 82 頁的『dsmEndGetData』	結束自儲存體取得物件的 **dsmBeginGetData** 階段作業。
第 82 頁的『dsmEndGetDataEx』	提供已傳送的免用 LAN 總位元組數。
第 82 頁的『dsmEndGetObj』	結束為特定物件取得資料的 **dsmGetObj** 階段作業。
第 83 頁的『dsmEndQuery』	表示 **dsmBeginQuery** 動作的結束。
第 83 頁的『dsmEndSendObj』	指出傳送到儲存體的資料結尾。
第 84 頁的『dsmEndSendObjEx』	提供壓縮資訊以及已傳送的位元組數。
第 84 頁的『dsmEndTxn』	結束 IBM Spectrum Protect 交易。

表 19. API 函數呼叫 (繼續)

函數呼叫與位置	說明
第 86 頁的『dsmEndTxnEx』	提供群組前導物件 ID 資訊，以使用於 **dsmGroupHandlerfunction** 呼叫。
第 87 頁的『dsmGetData 』	從 IBM Spectrum Protect 取得資料的位元組串流，並將其置於呼叫者的緩衝區中。
第 88 頁的『dsmGetBufferData』	從 IBM Spectrum Protect 伺服器取得資料的 an IBM Spectrum Protect 配置的緩衝區。
第 88 頁的『dsmGetNextQObj』	從先前的 **dsmBeginQuery** 呼叫取得下一個查詢回應，並將其置於呼叫者的緩衝區中。
第 91 頁的『dsmGetObj 』	從資料串流取得所要求的物件資料，並將其置於呼叫者 的緩衝區中。
第 92 頁的『dsmGroupHandler』	根據給定的輸入來決定對邏輯檔案群組執行的動作。
第 93 頁的『dsmInit』	啟動 API 階段作業並將用戶端連接到儲存體。
第 96 頁的『 dsmInitEx 』	使用可允許延伸驗證的附加參數來啟動 API 階段作業。
第 99 頁的『dsmLogEvent』	將使用者訊息記載至伺服器日誌檔或本端錯誤日誌，或同時記載到這兩個檔案。
第 100 頁的『dsmLogEventEx』	將使用者訊息記載至伺服器日誌檔或本端錯誤日誌，或同時記載到這兩個檔案。
第 101 頁的『dsmQueryAccess』	查詢伺服器，以取得物件之備份版本或保存副本的所有存取授權規則。
第 101 頁的『dsmQueryApiVersion 』	執行查詢要求，以取得應用程式用戶端所存取的 API 媒體庫版本。
第 102 頁的『dsmQueryApiVersionEx』	執行查詢要求，以取得應用程式用戶端所存取的 API 媒體庫版本。
第 102 頁的『dsmQueryCliOptions』	查詢使用者選項檔中的重要選項值。
第 103 頁的『dsmQuerySessInfo 』	對 IBM Spectrum Protect 啟動查詢要求，以取得 與 **dsmHandle** 中指定的階段作業相關的作業資訊。
第 104 頁的『dsmQuerySessOptions』	查詢在 **dsmHandle** 指定的階段作業中有效的重要選項值。
第 105 頁的『dsmRCMsg』	取得與 API 回覆碼相關的訊息文字。
第 106 頁的『dsmRegisterFS 』	對伺服器登錄新的檔案空間。
第 106 頁的『 dsmReleaseBuffer 』	傳回 IBM Spectrum Protect 配置的緩衝區。
第 107 頁的『dsmRenameObj』	將高階或低階物件名稱加以更名。
第 108 頁的『dsmRequestBuffer』	取得 IBM Spectrum Protect 為緩衝區副本排除所配置的緩衝區。
第 109 頁的『dsmRetentionEvent』	傳送物件 ID 清單給伺服器，並將對這些物件執行保留事件作業。
第 110 頁的『dsmSendBufferData』	傳送 IBM Spectrum Protect 配置的緩衝區的資料。

表 19. API 函數呼叫 (繼續)

函數呼叫與位置	說明
第 111 頁的『dsmSendData 』	透過緩衝區將資料的位元組串流傳送至 IBM Spectrum Protect。
第 112 頁的『dsmSendObj 』	啟動要求，將單一物件傳送儲存體。
第 114 頁的『dsmSetAccess』	為其他使用者或節點提供對物件備份版本或保存副本的存取權、對所有物件的存取權，或者對選擇集的存取權。
第 115 頁的『dsmSetUp』	改寫環境變數值。
第 117 頁的『dsmTerminate』	結束與伺服器的階段作業並清除 IBM Spectrum Protect 環境。
第 117 頁的『dsmUpdateFS 』	更新儲存體中的檔案空間。
第 118 頁的『dsmUpdateObj』	更新與伺服器作用中備份物件相關的 objInfo 資訊，或更新保存物件。
第 119 頁的『dsmUpdateObjEx』	更新與特定保存物件相關聯的 objInfo 資訊（即使有多個物件具有相同的名稱），或更新作用中的備份物件。

相關參考
API 回覆碼

dsmBeginGetData

dsmBeginGetData 函數呼叫會針對儲存體中的物件清單啟動一個還原或擷取作業。 這份物件清單 是包含在 **dsmGetList** 結構中。 應用程式會利用呼叫 **dsmBeginGetData** 之前 所查詢的值來建立這份清單。

呼叫者必須先使用從物件查詢取得的還原順序欄位，來排序此呼叫中所含的清單。 這樣即可以最 有效率的方法從儲存體進行還原物件，而不必回捲或重新裝載資料磁帶。

在取得所有物件時，最大 *dsmGetList.numObjID* 是 DSM_MAX_GET_OBJ。 在取得部份物件時，最大值是 DSM_MAX_PARTIAL_GET_OBJ。

在呼叫 **dsmBeginGetData** 之後，請對 **dsmGetObj** 執行一或多次呼叫，以取得清單中的每一個物件。 在取得 每一個物件後（或不需要物件的其他資料），就會傳送 **dsmEndGetObj** 呼叫。

取得所有物件或取消 **dsmEndGetObj** 之後，就會傳送 **dsmEndGetData** 呼叫。 然後 您可以重新開始這個循環。

語法

```
dsInt16_t dsmBeginGetData
  (dsUint32_t       dsmHandle,
   dsBool_t             mountWait,
   dsmGetType           getType,
   dsmGetList           *dsmGetObjListP
);
```

參數

dsUint32_t dsmHandle (I)
將此呼叫與先前的 **dsmInitEx** 呼叫結合的 handle。

dsBool_t mountWait (I)
布林的 true 或 false 值，表示當需要的資料目前為離線時，應用程式用戶端是否要等待裝載離線媒體。 若 mountWait 為 true， 應用程式會等待伺服器裝載所需要的媒體。 應用程式會等待媒體裝載完成或取消要求為止。

dsmGetType getType (I)
一種列舉類型，由 **gtBackup** 和 **gtArchive** 組成，可指出要取得的物件類型。

dsmGetList *dsmGetObjListP (I)

含有要還原或擷取之物件或部分物件的相關資訊的結構。 結構會指向物件 ID 清單， 若是部分物件的還原或擷取，則指向相關偏移和長度的清單。 如果應用程式使用部分物件 還原或擷取函數，請將 **dsmGetList.stVersion** 欄位設定為 **dsmGetListPORVersion**。 在部分物件還原或擷取時，您不能在傳送資料時壓縮資料。 若要強制執行此動作，請將 **ObjAttr.objCompressed** 設定為 *bTrue*。

請參閱第 58 頁的圖 19 和第 133 頁的『附錄 B API 類型定義原始檔』，以取得關於 此結構的其他資訊。

請參閱第 53 頁的『部分物件還原或擷取』，以取得關於部分物件的還原或擷取的其他資訊。

回覆碼

回覆碼的號碼會用括弧 () 括住。

表 *20. dsmBeginGetData* 的回覆碼

回覆碼	說明
DSM_RC_ABORT_INVALID_OFFSET (33)	在部分物件擷取期間指定的偏移大於物件的長度。
DSM_RC_ABORT_INVALID_LENGTH (34)	在部分物件擷取期間指定的長度大於物件 的長度，或者偏移加上長度的值超過物件結尾。
DSM_RC_NO_MEMORY (102)	沒有剩餘的 RAM 可以完成要求。
DSM_RC_NUMOBJ_EXCEED (2029)	**dsmGetList.numObjId** 大於 DSM_MAX_GET_OBJ。
DSM_RC_OBJID_NOTFOUND (2063)	找不到物件 ID。 物件未還原。
DSM_RC_WRONG_VERSION_PARM (2065)	應用程式用戶端的 API 版本與 IBM Spectrum Protect 媒體庫版本不同。

dsmBeginQuery

dsmBeginQuery 函數呼叫會對伺服器啟動查詢要求，以取得資料、檔案空間和管理類別的相關資訊。

明確地說，**dsmBeginQuery** 可以查詢下列項目：

- 保存的資料
- 備份的資料
- 作用中的備份資料
- 檔案空間
- 管理類別 (Management class)

從呼叫傳回的查詢資料是由一或多個對 **dsmGetNextQObj** 的呼叫所取得。 查詢完成時，會傳送 **dsmEndQuery** 呼叫。

語法

```
dsInt16_t dsmBeginQuery
  (dsUint32_t       dsmHandle,
   dsmQueryType          queryType,
   dsmQueryBuff          *queryBuffer
);
```

參數

dsUint32_t dsmHandle (I)

將此呼叫與先前的 **dsmInitEx** 呼叫結合的 handle。

dsmQueryType queryType (I)

識別要執行的查詢類型。 指派下列其中一個選項：

qtArchive

查詢保存的物件。

qtBackup

查詢備份的物件。

qtBackupActive

只針對您傳送的整個檔 案空間名稱查詢作用中的備份物件。 這個查詢稱為「捷徑」，這是從儲存體查詢作用中物件的有效方法。

必備項目: 您必須以 root 使用者身分登入 UNIX 或 Linux 作業系統。

qtFilespace

查詢已登錄的檔案空間。

qtMC

顯示已定義的管理類別。

qtBackupGroups

查詢關閉的群組。

qtOpenGroups

查詢開啟的群組。

qtProxyNodeAuth

查詢這個節點可代理的節點。

qtProxyNodePeer

查詢具有相同目標的對等節點。

dsmQueryBuff *queryBuffer (I)

指出映射至特定資料結構之緩衝區的指標。 此結構與您傳遞的查詢類型有關。 這些結構包含每一種查詢類型的選取準則。 請完成每一個結構中的欄位，來指定您要執行的查詢範圍。 每一個結構中的 stVersion 欄位都包含結構版本號碼。

資料結構及其相關的欄位包含下列項目：

qryArchiveData

objName

完整的物件名稱。 您可以在名稱的高階或低階部分使用萬用字元，例如星號 (*) 或問號 (?)。 星號可符合零個以上的字元，而問號會符合一個字元。 objName 的 objType 欄位可以包含下列其中一個值：

- DSM_OBJ_FILE
- DSM_OBJ_DIRECTORY
- DSM_OBJ_ANY_TYPE

如需高階和低階名稱的相關資訊，請參閱下列主題：第 19 頁的『高階與低階名稱』。

owner

物件的擁有者名稱。

insDateLowerBound

保存物件的插入日期低界限值。 為無界限的低界限值將「year」元件設定為 DATE_MINUS_INFINITE。

insDateUpperBound

保存物件的插入日期高界限值。 為無界限的高界限值將「year」元件設定為 DATE_PLUS_INFINITE。

expDateLowerBound

到期日的低界限值。 為無界限的低界限值將「year」元件設定為 DATE_MINUS_INFINITE。

expDateUpperBound

到期日的高界限值。 為了符合設為 NOLIMIT 的管理類別 **RETVer** 設定，請將「year」元件設定成 DATE_PLUS_INFINITE。

descr

保存說明。 輸入星號 (*) 可搜尋所有說明。

qryBackupData

objName

完整的物件名稱。 您可以在名稱的高階或低階部分使用萬用字元，例如星號 (*) 或問號 (?)。 星號可符合零個以上的字元，而問號會符合一個字元。 objName 的 objType 欄位可以包含下列其中一個值：

- DSM_OBJ_FILE
- DSM_OBJ_DIRECTORY
- DSM_OBJ_ANY_TYPE

如需高階和低階名稱的相關資訊，請參閱下列主題：第 19 頁的『高階與低階名稱』。

owner

物件的擁有者名稱。

objState

您可以查詢下列其中一種物件狀態：

- DSM_ACTIVE
- DSM_INACTIVE
- DSM_ANY_MATCH

pitDate

時間點值。 有此欄位的查詢會傳回在這個日期和時間之前備份的最新物件。 objState 可以是作用中或非作用中。 在 pitDate 之前刪除的物件則不會傳回。 例如：

```
Mon - backup ABC(1), DEF, GHI
Tue - backup ABC(2), delete DEF
Thr - backup ABC(3)
```

在星期五時，呼叫時間點值為星期三凌晨 12:00:00 的查詢時，呼叫會傳回下列資訊：

```
 ABC(2) - an Inactive copy
 GHI    - an Active copy
```

呼叫不會傳回 DEF，因為該物件在時間點值之前已刪除。

qryABackupData

objName

完整的物件名稱。 您可以在名稱的高階或低階部分使用萬用字元，例如星號 (*) 或問號 (?)。 星號可符合零個以上的字元，而問號會符合一個字元。 objName 的 objType 欄位可以包含下列其中一個值：

- DSM_OBJ_FILE
- DSM_OBJ_DIRECTORY
- DSM_OBJ_ANY_TYPE

如需高階和低階名稱的相關資訊，請參閱下列主題：第 19 頁的『高階與低階名稱』。

qryFSData

fsName

在此欄位中輸入特定檔案空間的名稱，或輸入一個星號 (*) 來擷取所有已登錄檔案空間的相關資訊。

qryMCData

mcName

輸入特定管理類別的名稱，或輸入一個空白字串（" "）來擷取所有管理類別的相關資訊。

註: 您不能使用星號（*）。

mcDetail

決定是否傳回管理類別之備份和保存副本群組的資訊。 下列值有效：

- bTrue
- bFalse

qryBackupGroup:

groupType

群組類型是 DSM_GROUPTYPE_PEER。

fsName

檔案空間名稱。

owner

擁有者 ID。

groupLeaderObjId

群組前導物件 ID。

objType

物件類型。

qryProxyNodeAuth：

targetNodeName

目標節點名稱。

peerNodeName

對等節點名稱。

hlAddress

高階名稱的對等位址。

llAddress

低階名稱的對等位址。

qryProxyNodePeer：

targetNodeName

目標節點名稱。

peerNodeName

對等節點名稱。

hlAddress

高階名稱的對等位址。

llAddress

低階名稱的對等位址。

回覆碼

下表說明 **dsmBeginQuery** 函數呼叫的回覆碼。

表 *21. dsmBeginQuery* 的回覆碼

回覆碼	回覆碼號碼	說明
DSM_RC_NO_MEMORY	102	沒有足夠的記憶體以完成要求。
DSM_RC_FILE_SPACE_NOT_FOUND	124	找不到指定的檔案空間。
DSM_RC_NO_POLICY_BLK	2007	無法取得伺服器原則資訊。
DSM_RC_INVALID_OBJTYPE	2010	無效的物件類型。
DSM_RC_INVALID_OBJOWNER	2019	無效的物件擁有者名稱。
DSM_RC_INVALID_OBJSTATE	2024	無效的物件條件。
DSM_RC_WRONG_VERSION_PARM	2065	應用程式用戶端的 API 版本與 IBM Spectrum Protect 媒體庫版本不同。

dsmBeginTxn

dsmBeginTxn 函數呼叫會開始以完整動作開始的一或多個 IBM Spectrum Protect 交易； 可以是所有的動作皆順利完成或者沒有任何動作完成。 動作可以是單一呼叫或一系列的 呼叫。 例如，一個 **dsmSendObj** 呼叫後面接著許多 **dsmSendData** 呼叫，即可視為單一動作。 同樣地，具有 **dataBlkPtr**（指出內含要備份之物件的資料區）的 **dsmSendObj** 呼叫 亦視為單一動作。

嘗試在單一交易中將一個以上的物件分組在一起來進行資料轉送作業。 將物件加以分組時， 可有效增進 IBM Spectrum Protect 系統的效能。 從伺服器和用戶端的觀點來看，啟動和結束每一個交易時， 都會有特定數量的額外作業。

您在單一交易中可執行的動作有許多限制。 這些限制包括：

- 您可以在單一交易中傳送或刪除的最大物件數目。 這個限制是位於 **dsmQuerySessInfo** 在 *ApiSessInfo.maxObjPerTxn* 欄位所傳回的資料中。 它會對應到 *TxnGroupMax* 伺服器選項。
- 在單一交易中傳送至伺服器的所有物件（備份或保存）， 其具備的副本目的地必須和物件的管理類別連結中所定義的相同。 這個值是位於 **dsmBindMC** 在 **mcBindKey.backup_copy_dest** 或 **mcBindKey.archive_copy_dest** 欄位所傳回的資料中。

透過 API，應用程式用戶端可以監視和控制這些限制，或者 API 可以監視這些限制。 如果 API 監視限制，當發生到達一或多個限制時，適當的 API 呼叫回覆碼會通知應用程式 用戶端。

請務必使 **dsmBeginTxn** 呼叫與 **dsmEndTxn** 呼叫相符，使一對 **dsmBeginTxn** 和 **dsmEndTxn** 呼叫內的動作集最佳化。

語法

```
dsInt16_t dsmBeginTxn  (dsUint32_t dsmHandle);
```

參數

dsUint32_t dsmHandle (I)
將此呼叫與先前的 **dsmInitEx** 呼叫結合的 handle。

回覆碼

回覆碼的號碼會用括弧 () 括住。

表 *22. dsmBeginTxn* 的回覆碼

回覆碼	說明
DSM_RC_ABORT_NODE_NOT_AUTHORIZED (36)	在 TXN 作業中不容許使用 FROMNODE 或 FROMOWNER。

dsmBindMC

dsmBindMC 函數呼叫會將管理類別結合或連結到傳遞的物件。 這個物件是透過選項檔中所指的「併入排除」清單來傳送。 如果在「併入」清單中找不到特定管理類別的相符項目，則指定預設的管理類別。 「排除」清單可以防止物件被備份， 但不能防止物件被保存。

應用程式用戶端可以使用 mcBindKey 結構中傳回的參數，來決定此物件 是否應該備份或保存，或者是否因為不同的副本目的地而必須啟動新的交易。 如需相關資訊，請參閱 **dsmBeginTxn**。

您必須先呼叫 **dsmBindMC** 然後再呼叫 **dsmSendObj** ，因為每個物件都必須有相關的管理類別。 此呼叫可以在交易內部或交易外部執行。 例如，在多重物件交易中，如果 **dsmBindMC** 指出物件 和前一個物件有不同的副本目的地，則必須結束交易，然後啟動新的交易。 在此情形下， 另一個 **dsmBindMC** 是不需要的，因為已經為此物件執行過一次。

語法

```
dsInt16_t dsmBindMC
  (dsUint32_t       dsmHandle,
   dsmObjName             *objNameP,
   dsmSendType             sendType,
   mcBindKey              *mcBindKeyP
);
```

參數

dsUint32_t dsmHandle (I)
將此呼叫與先前的 **dsmInitEx** 呼叫結合的 handle。

dsmObjName *objNameP (I)
結構的指標，其中包含檔案空間名稱、高階物件名稱、低階物件名稱和物件類型。

dsmSendType sendType (I)
指出此管理類別連結是針對保存或備份傳送執行。 此呼叫可能的值包括：

名稱	說明
stBackup	備份物件
stArchive	保存物件
stBackupMountWait	備份物件
stArchiveMountWait	保存物件

對於 **dsmBindMC** 呼叫而言，stBackup 和 stBackupMountWait 是同等的， 而 stArchive 和 stArchiveMountWait 是同等的。

mcBindKey *mcBindKeyP (O)
這是傳回管理類別資訊所在的 mcBindKey 結構的位址。 應用程式用戶端 可使用此處傳回的資訊來決定此物件是否適用於多重物件交易，或對連結到物件的管理類別 執行管理類別查詢。

回覆碼

回覆碼的號碼會用括弧 () 括住。

表 *23. dsmBindMC* 的回覆碼

回覆碼	說明
DSM_RC_NO_MEMORY (102)	沒有剩餘的 RAM 可以完成要求。
DSM_RC_INVALID_PARM (109)	其中一個傳送的參數具有無效的值。
DSM_RC_TL_EXCLUDED (185)	備份物件已被排除，無法傳送。
DSM_RC_INVALID_OBJTYPE (2010)	無效的物件類型。
DSM_RC_INVALID_SENDTYPE (2022)	無效的傳送類型。
DSM_RC_WRONG_VERSION_PARM (2065)	應用程式用戶端 API 版本與 IBM Spectrum Protect 媒體庫 版本不同。

dsmChangePW

dsmChangePW 函數呼叫會變更 IBM Spectrum Protect 密碼。在多使用者作業系統上（例如 UNIX 或 Linux），只有 root 使用者或授權使用者可使用這個呼叫。

在 Windows 作業系統上， 您可以在 dsm.opt 檔中指定密碼。 在此情形下，**dsmChangePW** 不會更新 dsm.opt 檔。 對 **dsmChangePW** 進行呼叫之後，您必須個別更新 dsm.opt 檔。

如果 **dsmInitEx** 傳回 DSM_RC_VERIFIER_EXPIRED，此呼叫必須順利處理完成。 在此情形下，如果 **dsmChangePW** 呼叫失敗，階段作業就會結束。

如果因為其他原因呼叫 **dsmChangePW** ，則不論回覆碼為何，階段作業仍會維持開啟。

語法

```
dsInt16_t dsmChangePW
  (dsUint32_t       dsmHandle,
   char                  *oldPW,
   char    *newPW);
```

參數

dsUint32_t dsmHandle (I)
將此呼叫與先前的 **dsmInitEx** 呼叫結合的 handle。

char *oldPW (I)
呼叫者的舊密碼。 最大長度是 DSM_MAX_VERIFIER_LENGTH。

char *newPW (I)
呼叫者的新密碼。 最大長度是 DSM_MAX_VERIFIER_LENGTH。

回覆碼

回覆碼的號碼會用括弧 () 括住。

表 *24. dsmChangePW* 的回覆碼

回覆碼	說明
DSM_RC_ABORT_BAD_VERIFIER (6)	輸入的密碼不正確。
DSM_RC_AUTH_FAILURE (137)	身份驗證失效。 舊密碼不正確。
DSM_RC_NEWPW_REQD (2030)	您必須為新密碼輸入一個值。
DSM_RC_OLDPW_REQD (2031)	您必須為舊密碼輸入一個值。
DSM_RC_PASSWD_TOOLONG (2103)	指定的密碼太長。
DSM_RC_NEED_ROOT (2300)	API 呼叫者必須是 root 使用者或是授權使用者。

dsmCleanUp

若呼叫 **dsmSetUp**，則使用 **dsmCleanUp** 函數呼叫。 **dsmCleanUp** 函數呼叫應該在 **dsmTerminate** 之後呼叫。 在呼叫 **dsmCleanUp** 之後，就不能進行任何其他的呼叫。

這個呼叫沒有特定的回覆碼。

語法

```
dsInt16_t DSMLINKAGE dsmCleanUp
    (dsBool_t        mtFlag);
```

參數

dsBool_t mtFlag (I)
此參數指定 API 是在單一執行緒或多執行緒模式中使用。 可能的值包括：

- DSM_SINGLETHREAD
- DSM_MULTITHREAD

dsmDeleteAccess

dsmDeleteAccess 函數呼叫會刪除物件之備份版本或保存副本現行的授權規則。 當您刪除授權規則時，就會取消使用者對該規則所指定的任何檔案的存取權。

當您使用 **dsmDeleteAccess** 時，一次只能刪除一個規則。 請透過 **dsmQueryAccess** 指令來取得規則 ID。

這個呼叫沒有特定的回覆碼。

語法

```
dsInt16_t DSMLINKAGE dsmDeleteAccess
             (dsUint32_t        dsmHandle,
              dsUint32_t          ruleNum) ;
```

參數

dsUint32_t dsmHandle (I)
將此呼叫與先前的 **dsmInitEx** 呼叫結合的 handle。

dsUint32_t ruleNum (I)
要刪除的存取規則的規則 ID。 這個值是從 **dsmQueryAccess** 函數呼叫取得。

dsmDeleteFS

dsmDeleteFS 函數呼叫會從儲存體刪除檔案空間。 若要刪除檔案空間，您必須擁有 IBM Spectrum Protect 管理者給予您的適當許可權。 如果要判斷您是否具備必要的許可權， 請呼叫 **dsmQuerySessInfo** 。 此函數呼叫會傳回類型為 *ApiSessInfo* 的資料結構， 其中包括兩個欄位， *archDel* 和 *backDel*。

註:

- 在 UNIX 或 Linux 作業系統上，只有 root 使用者或授權使用者可以刪除檔案空間。
- 如果您需要刪除的檔案空間包含備份版本，您必須擁有備份刪除權限 (***backDel*** = BACKDEL_YES)。 如果檔案空間包含保存副本，您必須擁有保存刪除權限 (*archDel* = ARCHDEL_YES)。 如果檔案空間同時包含 備份版本和保存副本，則您需要以上兩種類型的刪除權限。
- 使用 Archive Manager 伺服器時，並無法實際移除檔案空間。 即使檔案空間實際上並未被刪除，這個函數呼叫仍會傳回 *rc=0*。 唯一能驗證檔案空間是否已刪除的方法，是向伺服器發出檔案空間查詢。
- IBM Spectrum Protect 伺服器刪除檔案空間功能是一項背景處理程序。如果發生在傳遞回覆碼之前偵測到的那些錯誤以外的錯誤，它們會記錄在 IBM Spectrum Protect 伺服器日誌中。

語法

```
dsInt16_t dsmDeleteFS
  (dsUint32_t       dsmHandle,
   char                  *fsName,
   unsigned char   repository);
```

參數

dsUint32_t dsmHandle (I)
將此呼叫與先前的 **dsmInitEx** 呼叫結合的 handle。

char *fsName (I)
要刪除的檔案空間名稱之指標。 此處不可使用萬用字元。

unsigned char repository (I)
指出要刪除的檔案空間是備份儲存庫、保存儲存庫，或兩者皆是。 此欄位可能的值包括：

```
DSM_ARCHIVE_REP      /* 保存儲存庫    */
DSM_BACKUP_REP       /* 備份儲存庫     */
DSM_REPOS_ALL        /* 所有的儲存庫類型 */
```

回覆碼

回覆碼的號碼會用括弧 () 括住。

表 *25. dsmDeleteFS* 的回覆碼

回覆碼	說明
DSM_RC_ABORT_NOT_AUTHORIZED (27)	您沒有要刪除檔案空間的必要權限。
DSM_RC_INVALID_REPOS (2015)	儲存庫的值無效。
DSM_RC_FSNAME_NOTFOUND (2060)	找不到檔案空間名稱。
DSM_RC_NEED_ROOT (2300)	API 呼叫者必須是 root 使用者。

dsmDeleteObj

dsmDeleteObj 函數呼叫可停用備份物件、 刪除備份物件或刪除儲存體中的保存物件。 **dtBackup** 類型只能停用目前為作用中的備份副本。 **dtBackupID** 類型從伺服器移除 指定的物件 ID。 請從交易內呼叫此函數。

如需相關資訊，請參閱 **dsmBeginTxn**。

限制: 不能刪除包含在保留集中的備份物件。為了滿足長期資料保留需求，這些檔案會保留在伺服器儲存體中而且會依據該保留集專屬的到期日到期，到期之後即可刪除。

在傳送 **dsmDeleteObj** 之前，請先傳送在第 26 頁的『查詢 IBM Spectrum Protect 系統』 所說明 的查詢順序，以取得 **delInfo** 的資訊。 對 **dsmGetNextQObj** 的呼叫會傳回名稱為 **qryRespBackupData** 的資料結構（備份查詢）或 **qryRespArchiveData** 的資料結構（保存查詢）。 這些資料 結構包含您在 **delInfo** 所需的資訊。

maxObjPerTxn 的值會決定在單一交易中可刪除的最大物件數目。 如果要取得這個值，請呼叫 **dsmQuerySessInfo** 。

提示: 您的節點必須具有管理者所設定的適當許可權。如果要刪除保存物件，就必須具備保存刪除權限。 您不需要備份刪除權限，就可以停用備份物件。

語法

```
dsInt16_t dsmDeleteObj
  (dsUint32_t       dsmHandle,
   dsmDelType            delType,
   dsmDelInfo  delInfo)
```

參數

dsUint32_t dsmHandle (I)
將此呼叫與先前的 **dsmInitEx** 呼叫結合的 handle。

dsmDelType delType (I)
指出要刪除的物件（備份或保存）類型。 可能的值包括：

名稱	說明
dtArchive	要刪除的物件之前已經保存。
dtBackup	要停用的物件之前已經備份。
dtBackupID	要刪除的物件之前已經備份。 **限制:** 使用此 **delType** 和 *objID* 可從伺服器移除備份物件。 只有物件的擁有者可以刪除物件。 您可以刪除物件的任何版本（作用中或非作用中）。 伺服器會重新調解這些版本。 如果您刪除物件的作用中版本，則第一個非作用中版本會變成作用中。 如果您刪除物件的非作用中版本，則所有舊的版本都會往前推進。 節點必須登錄有 **backDel** 許可權。

dsmDelInfo delInfo (I)
其欄位是用來識別物件的結構。 這些欄位是不同的，端視物件為備份物件或保存物件而定。 停用 備份物件的結構 delBack 包含物件名稱和物件副本群組。 保存物件的結構 delArch 包含物件 ID。

移除備份物件的結構 delBackID 包含物件 ID。

回覆碼

回覆碼的號碼會用括弧 () 括住。

表 *26. dsmDeleteObj* 的回覆碼

回覆碼	說明
DSM_RC_FS_NOT_REGISTERED (2061)	檔案空間名稱未登錄。

表 26. *dsmDeleteObj* 的回覆碼 (繼續)

回覆碼	說明
DSM_RC_WRONG_VERSION_PARM (2065)	應用程式用戶端 API 版本與 IBM Spectrum Protect 媒體庫 版本不同。

dsmEndGetData

dsmEndGetData 函數呼叫會結束 **dsmBeginGetData** 階段作業，此階段作業會從儲存體取得物件。

在處理完您要還原的所有物件後，**dsmEndGetData** 函數呼叫會啟動，或提前結束 get 處理程序。 在繼續其他處理之前，請先呼叫 **dsmEndGetData** 來結束 **dsmBeginGetData** 階段作業。

端視呼叫 **dsmEndGetData** 的時機而定，API 可能需要先完成對部分資料串流的處理，然後才能停止處理程序。 因此，呼叫者不應該預期此呼叫可以立即傳回。 如果應用程式需要立即關閉階段作業並結束還原，請使用 **dsmTerminate**。

這個呼叫沒有特定的回覆碼。

語法

```
dsInt16_t dsmEndGetData
  (dsUint32_t dsmHandle);
```

參數

dsUint32_t dsmHandle (I)
將此呼叫與先前的 **dsmInitEx** 呼叫結合的 handle。

dsmEndGetDataEx

dsmEndGetDataEx 函數呼叫提供已傳送的免用 LAN 總位元組數。 這是 **dsmEndGetData** 函數呼叫 的延伸。

語法

這個呼叫沒有特定的回覆碼。

```
dsInt16_t dsmEndGetDataEx (dsmEndGetDataExIn_t * dsmEndGetDataExInP,
                           dsmEndGetDataExOut_t * dsmEndGetDataExOutP);
```

參數

dsmEndGetDataExIn_t *dsmEndGetDataExInP (I)
傳遞 end get 物件 dsmHandle，其識別階段作業並使它與後續呼叫產生關聯。

dsmEndGetDataExOut_t * dsmEndGetDataExOutP (O)
此結構包含此輸入參數：

totalLFBytesRecv
已接收的免用 LAN 總位元組數。

dsmEndGetObj

dsmEndGetObj 函數呼叫會結束 **dsmGetObj** 階段作業，此階段作業會為指定的物件取得資料。

收到物件的資料結尾之後啟動 **dsmEndGetObj** 呼叫。 這樣可以指出已收到所有的資料，或不會再收到此物件的資料。 您必須呼叫 **dsmEndGetObj** ，才能啟動另一個 **dsmGetObj** 呼叫。

端視呼叫 **dsmEndGetObj** 的時機而定，API 可能需要先完成對部分資料串流的處理， 然後才能停止處理程序。 不要預期此呼叫可以立即傳回。

語法

```
dsInt16_t dsmEndGetObj
  (dsUint32_t dsmHandle);
```

參數

dsUint32_t dsmHandle (I)
將此呼叫與先前的 **dsmInitEx** 呼叫結合的 handle。

回覆碼

回覆碼的號碼會用括弧 () 括住。

表 *27. dsmEndGetObj* 的回覆碼

回覆碼	說明
DSM_RC_NO_MEMORY (102)	沒有剩餘的 RAM 可以完成要求。

dsmEndQuery

dsmEndQuery 函數呼叫表示 **dsmBeginQuery** 動作的結束。 應用程式用戶端會傳送 **dsmEndQuery** 來完成查詢。 此呼叫會在透過 **dsmGetNextQObj** 取得所有查詢回應之後傳送，或者在傳回所有資料之前傳送以結束查詢。

提示: 在此情況下，IBM Spectrum Protect 會繼續將查詢資料從伺服器傳送到用戶端，但是 API 會捨棄任何其餘的資料。

一旦傳送 **dsmBeginQuery** ，就必須在其他活動開始前先傳送 **dsmEndQuery** 。

這個呼叫沒有特定的回覆碼。

語法

```
dsInt16_t dsmEndQuery
  (dsUint32_t dsmHandle);
```

參數

dsUint32_t dsmHandle (I)
將此呼叫與先前的 **dsmInitEx** 呼叫結合的 handle。

dsmEndSendObj

dsmEndSendObj 函數呼叫會指出傳送到儲存體的資料結尾。

輸入 **dsmEndSendObj** 函數呼叫來指出從 **dsmSendObj** 以及 **dsmSendData** 呼叫取得的資料結尾。 如果沒有執行此作業，就會發生通訊協定違規。

語法

```
  dsInt16_t dsmEndSendObj
  (dsUint32_t dsmHandle);
```

參數

dsUint32_t dsmHandle (I)
將此呼叫與先前的 **dsmInitEx** 呼叫結合的 handle。

回覆碼

回覆碼的號碼會用括弧 () 括住。

表 *28. dsmEndSendObj* 的回覆碼

回覆碼	說明
DSM_RC_NO_MEMORY (102)	沒有剩餘的 RAM 可以完成此要求。

dsmEndSendObjEx

dsmEndSendObjEx 函數呼叫提供其他有關已處理的位元組數資訊。 該資訊包括：已傳送的位元組總數、壓縮資訊、不需 LAN 的位元組數，以及刪除重複資料資訊。

dsmEndSendObjEx 函數呼叫是 **dsmEndSendObj** 函數呼叫的延伸。

語法

```
dsInt16_t dsmEndSendObjEx (dsmEndSendObjExIn_t  *dsmEndSendObjExInP,
                           dsmEndSendObjExOut_t *dsmEndSendObjExOutP);
```

參數

dsmEndSendObjExIn_t *dsmEndSendObjExInP (I)
這個參數會傳遞結束傳送物件 dsmHandle，其識別階段作業並使它與後續呼叫產生關聯。

dsmEndSendObjExOut_t *dsmEndSendObjExOutP (O)
此參數傳遞結束傳送物件資訊：

名稱	說明
totalBytesSent	從應用程式讀取的總位元組數。
objCompressed	顯示物件是否壓縮的旗號。
totalCompressedSize	壓縮之後的總位元組大小。
totalLFBytesSent	已傳送的免用 LAN 總位元組數。
objDeduplicated	顯示此 API 是否會對物件刪除重複資料的旗標。
totalDedupSize	在刪除重複資料之後傳送的位元組總數。

回覆碼

回覆碼的號碼會用括弧 () 括住。

表 *29. dsmEndSendObjEx* 的回覆碼

回覆碼	說明
DSM_RC_NO_MEMORY (102)	沒有剩餘的 RAM 可以完成此要求。

dsmEndTxn

dsmEndTxn 函數呼叫會結束 IBM Spectrum Protect 交易。

將 **dsmEndTxn** 函數呼叫與 **dsmBeginTxn** 合併，用來識別呼叫 或視為交易的一組呼叫。

應用程式用戶端可以在 **dsmEndTxn** 呼叫中指定必須確定或結束交易。

在異動範圍內執行下列所有呼叫：

- **dsmSendObj**
- **dsmSendData**
- **dsmEndSendObj**
- **dsmDeleteObj**

語法

```
dsInt16_t dsmEndTxn
  (dsUint32_t       dsmHandle,
   dsUint8_t             vote,
   dsUint16_t            *reason
);
```

參數

dsUint32_t dsmHandle (I)
將此呼叫與先前的 **dsmInitEx** 呼叫結合的 handle。

dsUint8_t vote (I)
指出應用程式用戶端是否要確定在先前的 **dsmBeginTxn** 呼叫以及這次呼叫之間完成的所有動作。 可能的值如下：

```
   DSM_VOTE_COMMIT     /* commit current transaction    */
   DSM_VOTE_ABORT      /* roll back current transaction */
```

只有在應用程式發現停止交易的原因時，才能使用 DSM_VOTE_ABORT。

dsUint16_t *reason (O)
如果對 **dsmEndTxn** 的呼叫結束但有錯誤，或者 vote 的值不符，此參數會有一個原因碼，指出 vote 失敗的原因。 呼叫的回覆碼可能是零，而原因碼可能為零以外的數字。 因此，應用程式用戶端必須檢查回覆碼以及原因碼 (if (rc || reason)) 兩者的錯誤，才能假設順利完成。

如果應用程式指定的 vote 為 DSM_VOTE_ABORT，則原因碼為 DSM_RS_ABORT_BY_CLIENT (3)。 請參閱第 123 頁的『附錄 A API 回覆碼原始檔：dsmrc.h』，以取得可能的原因碼清單。 回覆碼清單中的數字 1 到 50 是 保留給原因碼使用。 如果伺服器結束交易，則回覆碼為 DSM_RC_CHECK_REASON_CODE。 在此情形下，原因值會包含更多有關中斷原因的資訊。

回覆碼

回覆碼的號碼會用括弧 () 括住。

表 *30. dsmEndTxn* 的回覆碼

回覆碼	說明
DSM_RC_ABORT_CRC_FAILED (236)	從伺服器收到的 CRC 與用戶端所計算的 CRC 不符。
DSM_RC_INVALID_VOTE (2011)	指定給 vote 的值無效。
DSM_RC_CHECK_REASON_CODE (2302)	交易被中斷。 請檢查原因欄位。
DSM_RC_ABORT_STGPOOL_COPY_CONT_NO (241)	寫入其中一個副本儲存區失敗，IBM Spectrum Protect 儲存區選項 COPYCONTINUE 已設為 NO。交易終止。

表 30. *dsmEndTxn* 的回覆碼 (繼續)

回覆碼	說明
DSM_RC_ABORT_RETRY_SINGLE_TXN (242)	這個中斷代碼指出現行交易已中斷，因為在儲存作業期間發生問題。 在個別交易中傳送每一個檔案即可解決問題。 這個錯誤一般發生在下列情況中： · 下一個儲存區有不同的副本儲存區清單。 · 作業在交易途中被切換到這個儲存區。 .

dsmEndTxnEx

dsmEndTxnEx 函數呼叫提供群組前導物件 ID 資訊，可使用於 **dsmGroupHandler** 函數呼叫。 這是 **dsmEndTxn** 函數呼叫的延伸。

語法

```
dsInt16_t dsmEndTxnEx (dsmEndTxnExIn_t *dsmEndTxnExInP
                       dsmEndTxnExOut_t *dsmEndTxnExOutP);
```

參數

dsmEndTxnExIn_t *dsmEndTxnExInP (I)
此結構包含下列參數：

dsmHandle
識別階段作業並使它與後續 IBM Spectrum Protect 呼叫產生關聯的 handle。

dsUint8_t vote (I)
指出應用程式用戶端是否要確定在先前的 **dsmBeginTxn** 呼叫以及這次呼叫之間完成的所有動作。 可能的值為：

```
DSM_VOTE_COMMIT     /* commit current transaction    */
DSM_VOTE_ABORT      /* roll back current transaction */
```

只有在應用程式找到停止交易的原因時，才能使用 DSM_VOTE_ABORT。

dsmEndTxnExOut_t *dsmEndTxnExOutP (O)
此結構包含下列參數：

dsUint16_t *reason (O)
如果對 **dsmEndTxnEx** 的呼叫發生錯誤，或者 *vote* 的值不符，此參數會有一個原因碼，指出 vote 失敗的原因。

提示: 呼叫的回覆碼可能是零，而原因碼可能為零以外的數字。 因此，應用程式用戶端必須檢查回覆碼以及原因碼 (`if (rc || reason)`) 兩者的錯誤，才能假設順利完成。

如果應用程式指定的 vote 為 DSM_VOTE_ABORT，則原因碼為 DSM_RS_ABORT_BY_CLIENT (3)。 請參閱第 123 頁的『附錄 A API 回覆碼原始檔：dsmrc.h』，以取得可能的原因碼清單。 回覆碼清單中的數字 1 到 50 是 保留給原因碼使用。 如果伺服器結束交易，則回覆碼為 DSM_RC_CHECK_REASON_CODE。 在此情形下，原因值會包含更多有關中斷原因的資訊。

groupLeaderObjId
當 **dsmGroupHandler** 呼叫使用 DSM_ACTION_OPEN 旗號時，所傳回的群組前導物件 ID。

回覆碼

回覆碼的號碼會用括弧 () 括住。

表 *31. dsmEndTxnEx* 的回覆碼

回覆碼	說明
DSM_RC_INVALID_VOTE (2011)	指定給 vote 的值無效。
DSM_RC_CHECK_REASON_CODE (2302)	交易被中斷。 請檢查原因欄位。
DSM_RC_ABORT_STGPOOL_COPY_CONT_NO (241)	寫入其中一個副本儲存區失敗，IBM Spectrum Protect 儲存區選項 COPYCONTINUE 已設為 NO。 交易終止。
DSM_RC_ABORT_RETRY_SINGLE_TXN (242)	在同步寫入作業期間，交易中物件即將傳送到具有不同副本儲存區的目的地。 請結束目前的異動，並在每個物件本身的異動範圍內重新傳送物件。

dsmGetData

dsmGetData 函數呼叫會從 IBM Spectrum Protect 取得資料的位元組串流，並將其置於呼叫者的緩衝區中。 當前一個 **dsmGetObj** 或 **dsmGetData** 呼叫有更多可接收的資料時，應用程式用戶端會呼叫 **dsmGetData** 。

語法

```
dsInt16_t dsmGetData
  (dsUint32_t       dsmHandle,
   DataBlk *dataBlkPtr);
```

參數

dsUint32_t dsmHandle (I)
將此呼叫與先前的 **dsmInitEx** 呼叫結合的 handle。

DataBlk *dataBlkPtr (I/O)
指向一個結構，其中包含一個指標（指向已接收資料緩衝區）以及緩衝區的大小。 傳回時，此結構包含實際轉送的位元組數目。 請參閱第 133 頁的『附錄 B API 類型定義原始檔』以取得類型定義。

回覆碼

回覆碼的號碼會用括弧 () 括住。

表 *32. dsmGetData* 的回覆碼

回覆碼	說明
DSM_RC_ABORT_INVALID_OFFSET (33)	在部分物件擷取期間指定的偏移大於物件的長度。
DSM_RC_ABORT_INVALID_LENGTH (34)	在部分物件擷取期間指定的長度大於物件 的長度，或者偏移加上長度的值超過物件結尾。
DSM_RC_FINISHED (121)	已完成處理。 已收到最後的緩衝區。 請檢查 numBytes 的資料數量，然後呼叫 IBM Spectrum ProtectdsmEndGetObj 。
DSM_RC_NULL_DATABLKPTR (2001)	資料區塊指標為空值。
DSM_RC_ZERO_BUFLEN (2008)	資料區塊指標的緩衝區長度為零。
DSM_RC_NULL_BUFPTR (2009)	資料區塊指標的緩衝區指標為空值。
DSM_RC_WRONG_VERSION_PARM (2065)	應用程式用戶端的 API 版本與 IBM Spectrum Protect 程式庫版本不同。
DSM_RC_MORE_DATA (2200)	尚有其他資料必須取得。

dsmGetBufferData

dsmGetBufferData 函數呼叫會透過緩衝區接來自 IBM Spectrum Protect 的資料的位元組串流。在每一個呼叫之後，應用程式必須複製資料並透過 **dsmReleaseBuffer** 呼叫來釋放緩衝區。 如果應用程式保留的緩衝區數量等於 **dsmInitEx** 呼叫中指定的 numTsmBuffers，則在呼叫 **dsmReleaseBuffer** 之前，會持續鎖定 **dsmGetBufferData** 函數。

語法

```
dsInt16_t  dsmGetBufferData   (getDatatExIn_t        *dsmGetBufferDataExInP,
                               getDataExOut_t        *dsmGetBufferDataExOutP) ;
```

參數

getDataExIn_t * dsmGetBufferDataExInP (I)
此結構包含下列輸入參數。

dsUint32_t dsmHandle
識別階段作業並使它與先前 **dsmInitEx** 呼叫產生關聯的 handle。

getDataExOut_t * dsmGetBufferDataExOutP (0)
此結構包含下列輸出參數。

dsUint8_t tsmBufferHandle(0)
識別已接收緩衝區的 handle。

char *dataPtr(0)
資料寫入的位址。

dsUint32_t numBytes(0)
IBM Spectrum Protect 寫入的實際位元組數。

回覆碼

回覆碼的號碼會用括弧 () 括住。

表 *33. dsmGetBufferData* 的回覆碼

回覆碼	說明
DSM_RC_BAD_CALL_SEQUENCE (2041)	此呼叫不是在適當狀態下發出。
DSM_RC_OBJ_ENCRYPTED (2049)	此函數無法用於加密的物件。
DSM_RC_OBJ_COMPRESSED (2048)	此函數無法用於壓縮的物件。
DSM_RC_BUFF_ARRAY_ERROR (2045)	發生緩衝區陣列錯誤。

dsmGetNextQObj

dsmGetNextQObj 函數呼叫從先前的 **dsmBeginQuery** 呼叫取得下一個查詢回應，並將回應置於呼叫者緩衝區中。

dsmGetNextQObj 呼叫會被呼叫一次或數次。 每次呼叫函數時，就會擷取一筆查詢記錄，或傳回錯誤或 DSM_RC_FINISHED 原因碼。 如果傳回 DSM_RC_FINISHED，表示沒有其他資料可以處理。 如果已擷取所有查詢資料，或不再需要其他查詢資料，請傳送 **dsmEndQuery** 呼叫來結束查詢程序。

dataBlkPtr 參數必須指向以 **qryResp*Data** 結構類型所定義的緩衝區。 呼叫 **dsmGetNextQObj** 的上下文會決定查詢回應中所輸入的結構類型。

語法

```
dsInt16_t dsmGetNextQObj (dsUint32_t   dsmHandle,
   DataBlk *dataBlkPtr);
```

參數

dsUint32_t dsmHandle (I)
將此呼叫與先前的 **dsmInitEx** 呼叫相結合的 handle。

DataBlk *dataBlkPtr (I/O)
指向一個結構，其中包含一個指標（指向要接收的資料的緩衝區）以及緩衝區的大小。 這個緩衝區是 **qryResp*Data** 回應結構。 在傳回時，這個結構包含轉送的位元組數目。 下表說明與每一個查詢類型相關的結構。 如需 **DataBlk** 的類型定義相關資訊，請參閱下列主題：第 133 頁的『附錄 B API 類型定義原始檔』。

表 *34. DataBlk* 指標結構

查詢	回應結構	感興趣的欄位
qtArchive	**qryRespArchiveData**	**sizeEstimate** 包含在先前的 **dsmSendObj** 呼叫所傳送的值。 **mediaClass** 可以有 MEDIA_FIXED 值（如果物件位於磁碟）或 MEDIA_LIBRARY 值（如果物件位於磁帶）。 **clientDeduplicated** 指出此物件是否已被用戶端刪除重複資料。
qtBackup	**qryRespBackupData**	**restoreOrderExt** 為 dsUint16_t 類型。在 **dsmBeginGetData** 呼叫中還原多個物件時，會依這個欄位排序。這個呼叫的排序程式碼範例是在 API 範例 dapiqry.c 中。如需排序範例，請參閱下列主題：第 54 頁的圖 16。 **sizeEstimate** 包含在先前的 **dsmSendObj** 呼叫所傳送的值。 **mediaClass** 可以有 MEDIA_FIXED 值（如果物件位於磁碟）或 MEDIA_LIBRARY 值（如果物件位於磁帶）。 **clientDeduplicated** 指出此物件是否已被用戶端刪除重複資料。
qtBackupActive	**qryARespBackupData**	
qtBackupGroups	**qryRespBackupData**	**dsBool_t isGroupLeader** 如果為 True，表示這個物件是群組前導。

表 *34. DataBlk* 指標結構 *(*繼續*)*

查詢	回應結構	感興趣的欄位
qtOpenGroups	**qryRespBackupData**	**dsBool_t isOpenGroup;** 如果為 True，表示這個群組為已開啟且尚未完成。
qtFilespace	**qryRespFSData**	**backStartDate** 包含以 **backStartDate** 動作更新檔案空間時的伺服器時間戳記。 **backCompleteDate** 包含以 **backCompleteDate** 動作更新檔案空間時的伺服器時間戳記。 **lastReplStartDate** 包含前次在伺服器開始抄寫的時間戳記。 **lastReplCmpltDate** 包含前次在伺服器開始抄寫的時間戳記（即使失敗了也不例外）。 **lastBackOpDateFromServer** 包含前次在伺服器上儲存的時間戳記。 **lastBackOpDateFromLocal** 包含前次在用戶端上儲存的時間戳記。
qtMC	**qryRespMCData** **qryRespMCDetailData**	
qtProxyNodeAuth	**qryRespProxyNodeData** **targetNodeName** **peerNodeName** **hlAddress** **llAddress**	
qtProxyNodePeer	**qryRespProaxyNodeData** **targetNodeName** **peerNodeName** **hlAddress** **llAddress**	

回覆碼

下表說明 **dsmGetNextQObj** 函數呼叫的回覆碼。

表 *35.* ***dsmGetNextQObj*** 函數呼叫的回覆碼

回覆碼	回覆碼號碼	說明
DSM_RC_ABORT_NO_MATCH	2	找不到與所要求的查詢相符的項目。
DSM_RC_FINISHED	121	已完成處理（啟動 **dsmEndQuery**）。 沒有其他資料可以處理。
DSM_RC_UNKNOWN_FORMAT	122	IBM Spectrum Protect 嘗試還原或擷取的檔案使用不明格式。

表 35. ***dsmGetNextQObj*** 函數呼叫的回覆碼 (繼續)

回覆碼	回覆碼號碼	說明
DSM_RC_COMM_PROTOCOL_ERROR	136	通訊協定錯誤。
DSM_RC_NULL_DATABLKPTR	2001	指標並未指向資料區塊。
DSM_RC_INVALID_MCNAME	2025	無效的管理類別名稱。
DSM_RC_BAD_CALL_SEQUENCE	2041	呼叫序列是無效的。
DSM_RC_WRONG_VERSION_PARM	2065	應用程式用戶端 API 的版本與 IBM Spectrum Protect 程式庫版本不同。
DSM_RC_MORE_DATA	2200	尚有其他資料必須取得。
DSM_RC_BUFF_TOO_SMALL	2210	緩衝區太小。

dsmGetObj

dsmGetObj 函數呼叫會從 IBM Spectrum Protect 資料串流取得所要求的物件資料並將該資料置於 呼叫者的緩衝區。 **dsmGetObj** 呼叫會使用物件 ID 來從資料串流取得下一個物件或部分的物件。

指出的物件的資料放置於 **DataBlk** 所指向的緩衝區中。 如果有其他資料可用，您必須對 **dsmGetData** 進行一或多次呼叫，來接收剩餘的物件資料，直到傳回 DSM_RC_FINISHED 的回覆碼。 請檢查 DataBlk 中 **numBytes** 欄位，查看緩衝區中是否有任何剩餘的資料。

要求物件的順序應該和呼叫 **dsmBeginGetData** 時在 **dsmGetList** 參數中的順序一樣。 例外的情況是當應用程式用戶端需要在資料串流中忽略物件來取得清單後面的物件時。 如果 物件 ID 所指出的物件不是串流中的下一個物件，就會處理資料串流，直到找到物件或者串流完成為止。 請小心使用這項特性，因為它您可能需要處理和捨棄大量的資料，才能找到所要求的物件。

需求: 如果 **dsmGetObj** 傳回失敗碼（NOT FINISHED 或 MORE_DATA），則必須終止階段作業以停止還原作業。在使用加密並收到 RC_ENC_WRONG_KEY 的情況下，尤其需要這麼做。必須啟動具有正確金鑰的新階段作業。

語法

```
dsInt16_t dsmGetObj
  (dsUint32_t       dsmHandle,
   ObjID                *objIdP,
   DataBlk *dataBlkPtr);
```

參數

dsUint32_t dsmHandle (I)
將此呼叫與先前的 **dsmInitEx** 呼叫結合的 handle。

ObjID *objIdP (I)
指向要還原的物件 ID 的指標。

DataBlk *dataBlkPtr (I/O)
指向放置還原資料之緩衝區的指標。

回覆碼

回覆碼的號碼會用括弧 () 括住。

表 *36. dsmGetObj* 的回覆碼

回覆碼	說明
DSM_RC_ABORT_INVALID_OFFSET (33)	在部分物件擷取期間指定的偏移大於物件的長度。
DSM_RC_ABORT_INVALID_LENGTH (34)	在部分物件擷取期間指定的長度大於物件 的長度，或者偏移加上長度的值超過物件結尾。
DSM_RC_FINISHED (121)	已完成處理（啟動 **dsmEndGetObj** ）。
DSM_RC_WRONG_VERSION_PARM (2065)	應用程式用戶端的 API 版本與 IBM Spectrum Protect 程式庫版本不同。
DSM_RC_MORE_DATA (2200)	尚有其他資料必須取得。
RC_ENC_WRONG_KEY (4580)	**dsmInitEx** 呼叫中提供的金鑰或儲存的金鑰，與用來加密這個物件的金鑰不符。 請終止階段作業並提供正確的金鑰。

dsmGroupHandler

dsmGroupHandler 函數呼叫根據給定的輸入來決定對 邏輯檔案群組執行的動作。 用戶端使一些個別物件相互關聯，以便在 IBM Spectrum Protect 伺服器 上當作一個邏輯群組來參照和管理。

如需相關資訊，請參閱第 51 頁的『檔案分組』。

語法

```
dsInt16_t dsmGroupHandler (dsmGroupHandlerIn_t   *dsmGroupHandlerInP,
                           dsmGroupHandlerOut_t  *dsmGroupHandlerOutP);
```

參數

dsmGroupHandlerIn_t *dsmGroupHandlerInP (I)
傳遞群組屬性給 API。

groupType
群組的類型。 這些值包括：

· DSM_GROUPTYPE_PEER - 同層級群組

actionType
要執行的動作。 這些值包括：

· DSM_GROUP_ACTION_OPEN - 建立新群組
· DSM_GROUP_ACTION_CLOSE - 確定和儲存開啟的群組
· DSM_GROUP_ACTION_ADD - 附加至群組
· DSM_GROUP_ACTION_ASSIGNTO - 指定至另一個群組
· DSM_GROUP_ACTION_REMOVE- 移除群組中的成員

memberType.
物件的群組類型。 這些值包括：

· DSM_MEMBERTYPE_LEADER - 群組前導
· DSM_MEMBERTYPE_MEMBER - 群組成員

***uniqueGroupTagP**
關聯群組的唯一字串 ID。

leaderObjId
群組前導的物件 ID。

***objNameP**
指向群組前導的物件名稱的指標。

memberObjList
要移除或指定的物件清單。

dsmGroupHandlerOut_t *dsmGroupHandlerOutP (O)
傳遞 API 所完成的結構的位址。 會傳回結構版本號碼。

回覆碼

回覆碼的號碼會用括弧 () 括住。

表 *37. dsmGroupHandler* 的回覆碼

回覆碼	說明
DSM_RC_ABORT_INVALID_GROUP_ACTION (237)	嘗試對群組前導或成員進行無效的作業。

dsmInit

dsmInit 函數呼叫會啟動 API 階段作業並將用戶端連接 到 IBM Spectrum Protect 儲存體。 應用程式用戶端一次只能開啟一個作用中階段作業。 如果要以不同的參數開啟另一個階段作業，請先使用 **dsmTerminate** 呼叫來結束現行階段作業。

如果要允許跨節點的查詢、還原或擷取作業，請使用 *-fromnode* 和 *-fromowner* 字串選項。 如需相關資訊，請參閱 第 20 頁的『跨節點和跨擁有者存取物件』。

語法

```
dsInt16_t  dsmInit  (dsUint32_t        *dsmHandle,
   dsmApiVersion         *dsmApiVersionP,
   char                  *clientNodeNameP,
   char                  *clientOwnerNameP,
   char                  *clientPasswordP,
   char                  *applicationType,
   char                  *configfile,
   char          *options);
```

參數

dsUint32_t *dsmHandle (O)
識別此起始設定階段作業並使它與後續 IBM Spectrum Protect 呼叫產生關聯的 handle。

dsmApiVersion *dsmApiVersionP (I)
指向資料結構的指標，用來識別應用程式用戶端在此階段作業中所用的 API 版本。 此結構包含 dsmapitd.h 檔中設定的三個常數（DSM_API_VERSION、DSM_API_RELEASE 和 DSM_API_LEVEL）的值。 之前必須執行 **dsmQueryApiVersion** 的呼叫，以確保應用程式用戶端 API 版本與安裝於使用者工作站 上的 API 程式庫版本相容。

char *clientNodeNameP (I)
此參數是 IBM Spectrum Protect 階段作業的節點之指標。 所有的 階段作業必須有一個相關的節點名稱。 dsmapitd.h 檔案中的 DSM_MAX_NODE_LENGTH 常數 會設定節點名稱允許的大小上限。

節點名稱不區分大小寫。

如果此參數將兩者都設定為 NULL，而且 *passwordaccess* 設定為 *prompt*，則 API 會嘗試先從傳遞的選項字串中取得節點名稱。 如果不在那裡，則 API 會嘗試從配置檔或選項檔取得節點名稱。 如果尋找節點名稱的嘗試失敗，UNIX 或 Linux API 會使用系統主電腦名稱，而其他作業系統的 API 會傳回 DSM_RC_REJECT_ID_UNKNOWN 代碼。

如果此參數為空值，並且 dsm.opt 檔案（對於 Windows API）或 dsm.sys 檔案（對於 UNIX or Linux API）中的 passwordaccess 選項設定為 generate，則 API 會使用 nodename 選項值或系統主機名稱。

char *clientOwnerNameP (I)
此參數是 IBM Spectrum Protect 階段作業擁有者的指標。 如果啟動階段作業的作業系統是多使用者 作業系統，則不論物件的擁有者是誰，擁有者名稱 NULL（root 使用者）均有權備份、保存、 還原或擷取屬於應用程式的任何物件。

擁有者名稱是區分大小寫的。

如果此參數為空值並且 dsm.sys 檔案中的 passwordaccess 選項設定為 generate，則 API 會使用登入使用者 ID。

註: 在多使用者作業系統上，擁有者名稱不必與執行應用程式的階段作業的作用中使用者 ID 相符。

char *clientPasswordP (I)
此參數是執行 IBM Spectrum Protect 階段作業的節點之密碼的指標。 dsmapitd.h 檔案中 的 DSM_MAX_VERIFIER_LENGTH 常數會設定密碼所允許的大小上限。

密碼不區分大小寫。

除了在先啟動密碼檔的情況下，如果 *passwordaccess* 是設定為 *generate*，則此參數的值會被忽略。

char *applicationType (I)
此參數可識別正在執行階段作業的應用程式。 應用程式用戶端會定義這個值。

每次 API 應用程式用戶端對伺服器啟動一個階段作業時，伺服器就會更新用戶端的應用程式類型（或平台）。 因為此應用程式類型值會輸入至伺服器的 **platform** 欄位，建議您讓應用程式 類型值包含作業系統縮寫。 最大字串長度為 DSM_MAX_PLATFORM_LENGTH。

如果要察看應用程式類型的現行值，請呼叫 **dsmQuerySessInfo** 。

char *configfile (I)
此參數指向內含 API 配置檔完整名稱的字串。 API 配置檔中指定的選項會置換在用戶端選項檔中的規格。 在安裝 IBM Spectrum Protect（用戶端或 API）時定義選項檔。

char *options (I)
指向可包含使用者選項的字串，例如：

- *Compressalways*
- *Servername*（僅適用於 UNIX 或 Linux）
- *TCPServeraddr*
- *Fromnode*
- *Fromowner*
- *EnableClientEncryptKey*
- *Passwordaccess*

應用程式用戶端可使用選項清單來置換配置檔所設定的這些選項值。

選項格式為：

1. 選項清單中指定的每一個選項是以破折線 (-) 開頭，後面接著選項關鍵字。
2. 而關鍵字則是接著等號（=），然後是選項參數。
3. 如果選項參數包含空格，請用單引號或雙引號將參數括住。
4. 如果指定一個以上的選項，請用空格隔開選項。

如果選項為 NULL，則所有選項值都會取自使用者選項檔或 API 配置檔。

回覆碼

回覆碼的號碼會用括弧 () 括住。

表 *38. dsmInit* 的回覆碼

回覆碼	說明
DSM_RC_ABORT_SYSTEM_ERROR (1)	伺服器偵測到系統錯誤並已通知用戶端。
DSM_RC_REJECT_VERIFIER_EXPIRED (52)	密碼過期，必須更新。
DSM_RC_REJECT_ID_UNKNOWN (53)	找不到節點名稱。
DSM_RC_AUTH_FAILURE (137)	鑑別失敗。
DSM_RC_NO_STARTING_DELIMITER (148)	型樣中沒有起始定界符號。
DSM_RC_NEEDED_DIR_DELIMITER (149)	在 "match directories" meta-string ("...") 前後必須緊接著一個目錄定界符號，但未找到。
DSM_RC_NO_PASS_FILE (168)	密碼檔無法使用。
DSM_RC_UNMATCHED_QUOTE (177)	選項字串中有不相符的引號。
DSM_RC_NLS_CANT_OPEN_TXT (0610)	無法開啟訊息本文檔。
DSM_RC_INVALID_OPT (400)	選項字串中的登錄無效。
DSM_RC_INVALID_DS_HANDLE (2014)	無效的 DSM handle。
DSM_RC_NO_OWNER_REQD (2032)	當 *passwordaccess* 設定為 *generate* 時，擁有者參數必須為 NULL。
DSM_RC_NO_NODE_REQD (2033)	當 *passwordaccess* 設定為 *generate* 時，節點參數必須為 NULL。
DSM_RC_WRONG_VERSION (2064)	應用程式用戶端的 API 版本值比 IBM Spectrum Protect 版本高。
DSM_RC_PASSWD_TOOLONG (2103)	指定的密碼太長。
DSM_RC_NO_OPT_FILE (2220)	找不到配置檔。
DSM_RC_INVALID_KEYWORD (2221)	選項字串中指定的關鍵字無效。
DSM_RC_PATTERN_TOO_COMPLEX (2222)	併入/排除型樣太複雜了，IBM Spectrum Protect 無法解譯它。
DSM_RC_NO_CLOSING_BRACKET (2223)	在型樣內沒有結束括弧。
DSM_RC_INVALID_SERVER (2225)	在多使用者環境中，找不到系統配置檔中的伺服器。
DSM_RC_NO_HOST_ADDR (2226)	沒有足夠的資訊可連接主電腦。
DSM_RC_MACHINE_SAME (2227)	選項檔中定義的節點名稱不能與系統 主電腦名稱相同。
DSM_RC_NO_API_CONFIGFILE (2228)	無法開啟配置檔。
DSM_RC_NO_INCLEXCL_FILE (2229)	找不到納入-排除檔案。
DSM_RC_NO_SYS_OR_INCLEXCL (2230)	找不到 `dsm.sys` 檔案或納入-排除檔案。

相關概念

用戶端選項檔案概觀

處理選項

dsmInitEx

dsmInitEx 函數呼叫會啟動 API 階段作業並使用其他參數進行延伸驗證。

語法

```
dsInt16_t  dsmInitEx   (dsUint32_t        *dsmHandleP,
                        dsmInitExIn_t         *dsmInitExInP,
                        dsmInitExOut_t         *dsmInitExOutP) ;
```

參數

dsUint32_t *dsmHandleP (O)
識別此起始設定階段作業並使它與後續 IBM Spectrum Protect 呼叫產生關聯的 handle。

dsmInitExIn_t *dsmInitExInP
此結構包含下列輸入參數：

dsmApiVersion *dsmApiVersionP (I)
此參數是指向資料結構的指標，這個結構是用來識別應用程式用戶端在此階段作業中所用的 API 版本。 此結構包含在 dsmapitd.h 檔案中設定的四個常數的值：DSM_API_VERSION、DSM_API_RELEASE、DSM_API_LEVEL 和 DSM_API_SUBLEVEL. 呼叫 **dsmQueryApiVersionEx** 並驗證應用程式用戶端的 API 版本與在使用者工作站上安裝的 API 程式庫的版本相容。

char *clientNodeNameP (I)
此參數是 IBM Spectrum Protect 階段作業的節點之指標。 所有階段作業都必須與節點名稱相關聯。 dsmapitd.h 檔案中的 DSM_MAX_NODE_LENGTH 常數會設定節點名稱的大小上限。

節點名稱不區分大小寫。

如果此參數 設定為 NULL 而且 **passwordaccess** 設定為 prompt ，則 API 會嘗試先從傳遞的選項字串中取得節點名稱。 如果不在那裡，則 API 會嘗試從配置檔或選項檔取得節點名稱。 如果尋找節點名稱的嘗試失敗，UNIX 或 Linux API 會使用系統主機名稱，而其他作業系統的 API 會傳回 DSM_RC_REJECT_ID_UNKNOWN。

如果此參數為空值，並且 dsm.opt 檔案（對於 Windows API）或 dsm.sys 檔案（對於 UNIX or Linux API）中的 **passwordaccess** 選項設定為 generate，則 API 會使用 nodename 選項值或系統主機名稱。

char *clientOwnerNameP (I)
此參數是 IBM Spectrum Protect 階段作業擁有者的指標。 如果作業系統是多使用者平台，則不論物件的擁有者是誰，擁有者名稱 NULL（root 使用者）均有權備份、保存、還原或擷取屬於應用程式的任何物件。

擁有者名稱是區分大小寫的。

如果此參數為空值並且 dsm.sys 檔案中的 **passwordaccess** 選項設定為 generate，則 API 會使用登入使用者 ID。

註: 在多使用者平台上，擁有者名稱不必與執行應用程式的階段作業的作用中使用者 ID 相符。

char *clientPasswordP (I)
執行 IBM Spectrum Protect 階段作業的節點之密碼的指標。 dsmapitd.h 檔案中 的 DSM_MAX_VERIFIER_LENGTH 常數會設定密碼所允許的大小上限。

密碼不區分大小寫。

除了在先啟動密碼檔的情況下，如果 **passwordaccess** 是設定為 generate，則此參數的值會被忽略。

char *userNameP;
指向具有此節點用戶端權限的管理使用者名稱的指標。

char *userPasswordP;
指向 **userName** 參數的密碼的指標（若已提供一個值）。

char *applicationType (I)
指出正在執行 IBM Spectrum Protect 階段作業的應用程式。 應用程式用戶端會定義這個值。

每次 API 應用 程式用戶端對伺服器啟動一個階段作業時，伺服器上就會更新用戶端的應用程式類型（或作業系統）。 該值會輸入至伺服器的 **platform** 欄位。考量在該值中使用作業系統 ID。字串長度上限在 DSM_MAX_PLATFORM_LENGTH 常數中定義。

如果要檢視應用程式類型的現行值，請呼叫 **dsmQuerySessInfo** 。

char *configfile (I)
指向內含 API 配置檔完整名稱的字串。API 配置檔中指定的選項會置換在用戶端選項檔中的規格。在安裝 IBM Spectrum Protect（用戶端或 API）時定義選項檔。

char *options (I)
指向可包含使用者選項的字串，例如：

- `Compressalways`
- `Servername`（僅適用於 UNIX 和 Linux 系統）
- `TCPServeraddr`（不適用於 UNIX 系統）
- `Fromnode`
- `Fromowner`
- `Passwordaccess`

應用程式用戶端可使用選項清單來置換配置檔所設定的這些選項值。

選項具有下列格式：

1. 選項清單中指定的每一個選項是以破折線 (-) 開頭，後面接著選項關鍵字。
2. 關鍵字是接著等號 (=)，然後是選項參數。
3. 如果選項參數包含空格，請用單引號或雙引號將參數括住。
4. 如果指定一個以上的選項，請用空格隔開選項。

如果選項為 NULL，則所有選項值都會取自使用者選項檔或 API 配置檔。

dirDelimiter
位於檔案空間、高階或低階名稱開頭的目錄區隔字元。 只有在應用程式置換系統預設值時，才必須指定 **dirDelimiter** 參數。在 UNIX 或 Linux 環境中，預設值是正斜線 (/)。在 Windows 環境中，預設值是反斜線 (\)。

useUnicode
指示是否啟用 Unicode 的布林旗標。**useUnicode** 旗標必須是 `false` 以實現 UNIX 與 Windows 系統之間的跨平台交互作業能力。

bCrossPlatform
必須設定的布林旗標 (`bTrue`)， 以實現 UNIX 與 Windows 系統之間的跨平台交互作業能力。當設定 `bCrossPlatform` 旗標時，API 可確保檔案空間不是 Unicode 並且應用程式不使用 Unicode。使用 Unicode 的 Windows 應用程式與使用非 Unicode 編碼的應用程式不相容。對於使用 Unicode 的 Windows 應用程式，不能設定 `bCrossPlatform` 旗標。

UseTsmBuffers
指出是否要使用緩衝區副本排除。

numTsmBuffers
`useTsmBuffers=bTrue` 時的緩衝區數量。

bEncryptKeyEnabled
指出是否要使用應用程式管理的金鑰進行加密。

encryptionPasswordP
加密密碼。

限制: 當 `encryptkey=save` 時，如果加密金鑰存在，則會忽略在 **encryptionPasswordP** 中指定的值。

dsmAppVersion *appVersionP (I)
此參數是指向資料結構的指標，這個結構是用來識別啟動 API 階段作業的應用程式的版本資訊。此結構包含在 `tsmapitd.h` 檔案中設定的四個常數的值：applicationVersion、applicationRelease、applicationLevel 和 applicationSubLevel。

dsmInitExOut_t *dsmInitExOut P
此結構包含輸出參數。

dsUint32_t *dsmHandle (0)
定義此起始設定階段作業並且將它與後續 API 呼叫相結合的 handle。

infoRC
有關回覆碼的其餘資訊。 請檢查函數回覆碼和 **infoRC** 的值。如果 **infoRC** 值為 DSM_RC_REJECT_LASTSESS_CANCELED (69)，則 IBM Spectrum Protect 表示管理者取消了前次階段作業。

回覆碼

回覆碼的號碼會用括弧 () 括住。

表 *39.* ***dsmInitEx*** 的回覆碼

回覆碼	說明
DSM_RC_ABORT_SYSTEM_ERROR (1)	IBM Spectrum Protect 伺服器偵測到系統錯誤並已通知用戶端。
DSM_RC_REJECT_VERIFIER_EXPIRED (52)	密碼已過期，必須更新。下一個呼叫必須是 **dsmChangePW**，並使用這個呼叫傳回的 handle。
DSM_RC_REJECT_ID_UNKNOWN (53)	找不到節點名稱。
DSM_RC_TA_COMM_DOWN (103)	通訊鏈結中斷。
DSM_RC_AUTH_FAILURE (137)	鑑別失敗。
DSM_RC_NO_STARTING_DELIMITER (148)	型樣中沒有起始定界符號。
DSM_RC_NEEDED_DIR_DELIMITER (149)	在 "match directories" meta-string ("...") 前後必須緊接著一個目錄定界符號，但未找到。
DSM_RC_NO_PASS_FILE (168)	密碼檔無法使用。
DSM_RC_UNMATCHED_QUOTE (177)	選項字串中有不相符的引號。
DSM_RC_NLS_CANT_OPEN_TXT (0610)	無法開啟訊息本文檔。
DSM_RC_INVALID_OPT (2013)	選項字串中的登錄無效。
DSM_RC_INVALID_DS_HANDLE (2014)	無效的 DSM handle。
DSM_RC_NO_OWNER_REQD (2032)	當 **passwordaccess** 設定為 `generate` 時，擁有者參數必須為 NULL。
DSM_RC_NO_NODE_REQD (2033)	當 **passwordaccess** 設定為 generate 時，節點參數必須為 NULL。
DSM_RC_WRONG_VERSION (2064)	應用程式用戶端的 API 版本值比 IBM Spectrum Protect 版本高。
DSM_RC_PASSWD_TOOLONG (2103)	指定的密碼太長。
DSM_RC_NO_OPT_FILE (2220)	找不到配置檔。
DSM_RC_INVALID_KEYWORD (2221)	選項字串中指定的關鍵字無效。
DSM_RC_PATTERN_TOO_COMPLEX (2222)	併入/排除型樣太複雜，使得 IBM Spectrum Protect 無法解譯。

表 39. ***dsmInitEx*** 的回覆碼 (繼續)

回覆碼	說明
DSM_RC_NO_CLOSING_BRACKET (2223)	在型樣內沒有結束括弧。
DSM_RC_INVALID_SERVER (2225)	在多使用者環境中，找不到系統配置檔中的伺服器。
DSM_RC_NO_HOST_ADDR (2226)	沒有足夠的資訊可連接主電腦。
DSM_RC_MACHINE_SAME (2227)	選項檔中定義的節點名稱不能與系統主機名稱相同。
DSM_RC_NO_API_CONFIGFILE (2228)	無法開啟配置檔。
DSM_RC_NO_INCLEXCL_FILE (2229)	找不到納入-排除檔案。
DSM_RC_NO_SYS_OR_INCLEXCL (2230)	找不到 dsm.sys 或納入-排除檔案。

相關概念
用戶端選項檔案概觀
處理選項

dsmLogEvent

dsmLogEvent 函數呼叫會將使用者訊息 (ANE4991 I) 記載至伺服器日誌檔或本端錯誤日誌，或同時記載到這兩個檔案。 呼叫中傳入 了 **logInfo** 類型的結構。 此呼叫必須在 **InSession** 狀態下，從階段作業內部執行。 請不要 在 send、get 或 query 內執行它。 如果要擷取伺服器上所記載的訊息，請透過管理用戶端來使用 **query actlog** 指令。

請參閱摘要狀態圖 第 61 頁的圖 20。

語法

```
dsInt16_t dsmLogEvent
     (dsUint32_t        dsmHandle,
     logInfo        *logInfoP);
```

參數

dsUint32_t dsmHandle(I)
將此呼叫與先前的 **dsmInitEx** 呼叫相結合的 handle。

logInfo *logInfoP (I)
傳送訊息及目的地。 應用程式用戶端會負責為結構配置儲存體。

logInfo 結構中的欄位包括：

訊息
要記載的訊息文字。 它必須是以空值結束的字串。 最大長度是 DSM_MAX_RC_MSG_LENGTH。

dsmLogtype
指定要記載訊息的地方。 可能的值包括：**logServer**、**logLocal**、**logBoth**。

回覆碼

回覆碼的號碼會用括弧 () 括住。

表 *40. dsmLogEvent* 的回覆碼

回覆碼	說明
DSM_RC_STRING_TOO_LONG (2120)	訊息字串太長。

dsmLogEventEx

dsmLogEventEx 函數呼叫會將使用者訊息記載至伺服器日誌檔、本端錯誤日誌或兩者。 必須在 **InSession** 狀態下，從階段作業內進行此呼叫。 無法在 send、get 或 query 呼叫內進行此呼叫。

摘要狀態圖: 如需階段作業互動的概觀，請參閱下列主題中的摘要狀態圖：

第 61 頁的圖 20

嚴重性決定 IBM Spectrum Protect 訊息編號。 如果要檢視記載在伺服器上的訊息，請透過管理用戶端來使用 **query actlog** 指令。 如果應用程式產生許多寫入用戶端日誌 dsmLogType（是 logLocal 或 logBoth）的用戶端訊息，請使用 IBM Spectrum Protect 用戶端選項 errorlogretention 來刪改用戶端錯誤日誌檔。 如需相關資訊，請參閱 IBM Spectrum Protect 伺服器說明文件。

語法

```
extern dsInt16_t DSMLINKAGE  dsmLogEventEx(
       dsUint32_t             dsmHandle,
       dsmLogExIn_t          *dsmLogExInP,
       dsmLogExOut_t         *dsmLogExOutP
);
```

參數

dsUint32_t dsmHandle(I)
將此呼叫與先前的 **dsmInitEx** 呼叫相結合的 handle。

dsmLogExIn_t *dsmLogExInP
此結構包含輸入參數。

dsmLogSeverity severity;
此參數是指事件的嚴重性。 可能的值為：

```
   logSevInfo,       /* information ANE4990 */
   logSevWarning,    /* warning     ANE4991 */
   logSevError,      /* Error       ANE4992 */
   logSevSevere      /* severe      ANE4993 */
```

char appMsgID[8];
此參數為一字串，用來識別特定的應用程式訊息。 適合的格式是三個字元後面接著四個數字，例如：DSM0250。

dsmLogType logType;
此參數指定事件要導入至何處。 參數有下列的可能值：

- logServer
- logLocal
- logBoth

char *message;
此參數是記載到日誌的事件訊息內文。 文字必須是以空值結束的字串。 最大長度是 DSM_MAX_RC_MSG_LENGTH。

限制: 傳送至伺服器的訊息必須是英文。 非英文的訊息不會正確顯示。

dsmLogExOut_t *dsmLogExOutP
此結構包含輸出參數。 目前沒有輸出參數。

回覆碼

回覆碼的號碼會用括弧 () 括住。

表 *41. dsmLogEventEx* 的回覆碼

回覆碼	說明
DSM_RC_STRING_TOO_LONG (2120)	訊息字串太長。

dsmQueryAccess

dsmQueryAccess 函數呼叫會查詢伺服器，以取得物件 之備份版本或保存副本的所有存取授權規則。 有一個指向存取規則陣列的指標會傳入 呼叫中，並且傳回完成的陣列。 指向規則數目的指標會傳入，以指出陣列中有多少規則。

這個呼叫沒有特定的回覆碼。

語法

```
dsInt16_t DSMLINKAGE dsmQueryAccess
                   (dsUint32_t            dsmHandle),
                    qryRespAccessData     **accessListP,
                    dsUint16_t            *numberOfRules) ;
```

參數

dsUint32_t dsmHandle (I)
將此呼叫與先前的 **dsmInitEx** 呼叫結合的 handle。

qryRespAccessData **accessListP (O)
指向 API 程式庫所配置的 qryRespAccessData 元素陣列的指標。 每一個元素都會對應到一個存取規則。 陣列中的元素數目是在 **numberOfRules** 參數中傳回。 這項 資訊是在每一個 qryRespAccessData 元素中傳回，包括下列項目：

名稱	說明
ruleNumber	存取規則的 ID。 它定義刪除的規則。
AccessType	備份或保存類型。
Node	您提供存取權的節點。
Owner	您提供存取權的使用者。
objName	高階或低階檔案空間描述子。

dsUint32_t *numberOfRules (O)
傳回 accessList 陣列中的規則數目。

dsmQueryApiVersion

dsmQueryApiVersion 函數呼叫會執行查詢要求，以取得應用程式用戶端所存取的 API 程式庫版本。

所有對 API 的更新都是以向上相容的格式來進行。 當應用程式用戶端的 API 版本或版次小於或等於一般使用者工作站上的 API 程式庫版本時，就可以在無須變更的情況下操作。 請注意，在繼續之前，萬一 **dsmQueryApiVersion** 呼叫傳回一個比應用程式用戶端更舊的版本或版本版次，則某些 API 呼叫可能會被強化，而使用者的舊版 API 可能不支援這種強化。

應用程式 API 版本號碼 是儲存在 dsmapitd.h 標頭檔中 ，並以 DSM_API_VERSION、DSM_API_RELEASE 和 DSM_API_LEVEL 常數的形式來儲存。

這個呼叫沒有特定的回覆碼。

語法

```
void dsmQueryApiVersion
  (dsmApiVersion *apiVersionP);
```

參數

dsmApiVersion *apiVersionP (O)
此參數是指向結構的指標，其中包含 API 程式庫版本、版次及層次元件。 比方說，如果程式庫是 1.1.0 版，則在從呼叫傳回之後，結構的欄位會包含下列值：

```
dsmApiVersionP->version  = 1
dsmApiVersionP->release  = 1
dsmApiVersionP->level    = 0
```

dsmQueryApiVersionEx

dsmQueryApiVersionEx 函數呼叫會執行查詢要求，以取得應用程式用戶端所存取的 API 程式庫版本。

所有對 API 的更新都是以向上相容的格式來進行。 當應用程式用戶端的 API 版本或版次小於或等於使用者工作站上的 API 程式庫版本時，就可以在無須變更的情況下操作。 請參閱 README_api_enu 檔中的程式碼變更摘要，以取得新版相容性的例外。 如果 **dsmQueryApiVersionEx** 呼叫傳回的版本或版本版次與應用程式用戶端的不同，請注意，在繼續之前，某些 API 呼叫可能會被強化，而使用者的舊版 API 可能不支援這種強化。

應用程式 API 版本號碼是以 DSM_API_VERSION、DSM_API_RELEASE、DSM_API_LEVEL 和 DSM_API_SUBLEVEL 常數的形式，儲存在 dsmapitd.h 標頭檔中。

這個呼叫沒有特定的回覆碼。

語法

```
void dsmQueryApiVersionEx (dsmApiVersionEx *apiVersionP);
```

參數

dsmApiVersionEx *apiVersionP (O)
此參數是指向結構的指標，其中包含 API 程式庫的版本、版次及子層次元件。 比方說，如果程式庫是 5.5.0.0 版，則在從呼叫傳回之後，結構的欄位會包含下列值：

- ApiVersionP->version = 5
- ApiVersionP->release = 5
- ApiVersionP->level = 0
- ApiVersionP->subLevel = 0

dsmQueryCliOptions

dsmQueryCliOptions 函數呼叫會查詢使用者選項檔中的重要選項值。 **optStruct** 這類型的結構會在呼叫中傳送，並且包含資訊。 此呼叫是在呼叫 **dsmInitEx** 之前執行，它會在階段作業之前決定設定。

這個呼叫沒有特定的回覆碼。

語法

```
dsInt16_t dsmQueryCliOptions
    (optStruct     *optstructP);
```

參數

optStruct *optstructP (I/O)
此參數會傳送 API 所完成的結構的位址。 應用程式用戶端會負責為結構配置儲存體。 順利傳回後，就會在結構的欄位中輸入適當的資訊。

在 **optStruct** 結構中會傳回下列資訊：

名稱	說明
dsmiDir	環境 DSMI_DIR 變數的值。
dsmiConfig	DSMI_CONFIG 環境變數所指定的用戶端選項檔。
serverName	IBM Spectrum Protect 伺服器名稱。
commMethod	選取的通訊方法。 請參閱 dsmapitd.h 檔中 DSM_COMM_* 的 #defines。
serverAddress	以通訊方法為主的伺服器位址。
nodeName	用戶端節點（機器）名稱。
壓縮	這個欄位提供有關壓縮選項的資訊。
passwordAccess	這些值包括：*bTrue* 代表產生， *bFalse* 代表提示。

相關概念
處理選項

dsmQuerySessInfo

dsmQuerySessInfo 函數呼叫會對 IBM Spectrum Protect 發出查詢要求，以取得關於 **dsmHandle** 中的指定階段作業之作業的資訊。 類型為 **ApiSessInfo** 的結構會在呼叫中傳送，並且輸入所有可用的階段作業相關資訊。 此呼叫是在順利完成 **dsmInitEx** 呼叫後啟動。

在 **ApiSessInfo** 結構中傳回的資訊包括下列項目：

- 伺服器資訊：埠號、日期和時間，以及類型
- 用戶端預設值：應用程式類型、刪除許可權、定界字元和交易限制
- 階段作業資訊：登入 ID 及擁有者
- 原則資料：網域、作用中的原則集和保留寬限期

請參閱第 133 頁的『附錄 B API 類型定義原始檔』，以取得傳送的結構內容以及其中每個欄位的相關資訊。

語法

```
dsInt16_t dsmQuerySessInfo
  (dsUint32_t       dsmHandle,
   ApiSessInfo   *SessInfoP);
```

參數

dsUint32_t dsmHandle (I)
將此呼叫與先前的 **dsmInitEx** 呼叫結合的 handle。

ApiSessInfo *SessInfoP (I/O)
此參數會傳送 API 所輸入的結構的位址。 應用程式用戶端會負責為結構配置儲存體，並且完成用來指出所用結構之版本的欄位登錄。 順利傳回後，就會在結構的欄位中輸入適當的資訊。 **adsmServerName** 是在 IBM Spectrum Protect 伺服器的 **define server** 指令所提供的名稱。 如果 archiveRetentionProtection 欄位的值為 true，則會對伺服器啟用保留保護支援。

回覆碼

回覆碼的號碼會用括弧 () 括住。

表 *42. dsmQuerySessInfo* 的回覆碼

回覆碼	說明
DSM_RC_NO_SESS_BLK (2006)	沒有伺服器階段作業區塊資訊。
DSM_RC_NO_POLICY_BLK (2007)	沒有伺服器原則資訊可用。
DSM_RC_WRONG_VERSION_PARM (2065)	應用程式用戶端的 API 版本與 IBM Spectrum Protect 程式庫版本不同。

dsmQuerySessOptions

dsmQuerySessOptions 函數呼叫會查詢在 **dsmHandle** 指定的階段作業中有效的重要選項值。 optStruct 這類型的結構會在呼叫中傳送，並且包含資訊。

此呼叫是在順利完成 **dsmInitEx** 呼叫後啟動。 傳回的值可能與 **dsmQueryCliOptions** 呼叫所傳回的值不同，根據傳遞 給 **dsmInitEx** 呼叫的值而定，主要是 optString 和 optFile。 有關選項優先順序的資訊，請參閱第 1 頁的『瞭解配置檔和選項檔』。

這個呼叫沒有特定的回覆碼。

語法

```
dsInt16_t dsmQuerySessOptions
    (dsUint32_t        dsmHandle,
    optStruct             *optstructP
);
```

參數

dsUint32_t dsmhandle(I)
將此呼叫與先前的 **dsmInitEx** 呼叫相結合的 handle。

optStruct *optstructP (I/O)
此參數會傳送 API 所完成的結構的位址。 應用程式用戶端會負責為結構配置儲存體。 順利傳回後，就會在結構的欄位中輸入適當的資訊。

在 optStruct 結構中傳回的資訊是：

名稱	說明
dsmiDir	DSMI_DIR 環境變數的值。
dsmiConfig	DSMI_CONFIG 環境變數指定的 dsm.opt 檔。
serverName	選項檔中的 IBM Spectrum Protect 伺服器段落名稱。
commMethod	選取的通訊方法。 請參閱 dsmapitd.h 檔中 DSM_COMM_* 的 #defines。
serverAddress	以通訊方法為主的伺服器位址。
nodeName	用戶端節點（機器）的名稱。
壓縮 (compression)	壓縮選項的值（bTrue=on 和 bFalse=off）。
compressAlways	compressalways 選項的值（bTrue=on 和 bFalse=off）。
passwordAccess	bTrue 值代表產生，bFalse 代表提示。

相關概念
處理選項

dsmRCMsg

dsmRCMsg 函數呼叫會取得與 API 回覆碼相關的訊息文字。

重要: 如果在 **dsmInitEx** 失敗後呼叫 **dsmRCMsg**，則不會忽略 Servername 選項，並且會在預設為伺服器定義的日誌檔中列印錯誤。

msg 參數會顯示用括弧 ( ) 括住訊息字首回覆碼，後面緊接著 訊息文字。 例如，**dsmRCMsg** 呼叫可能會傳回下列文字：

```
ANS0264E (RC2300) Only root user can execute dsmChangePW
 or dsmDeleteFS
.
```

就字元在 ANSII 和 OEM 字碼頁中會不同的某些語言而言，可能有必要在將字串印出之前，將它們從 ANSII 轉換成 OEM（例如，東歐單位元組字集）。 以下是個範例：

```
   dsmRCMsg(dsmHangle, rc, msgBuf);
   #ifdef WIN32
   #ifndef WIN64
   CharToOemBuff(msgBuf, msgBuf, strlen(msgBuf));
   #endif
   #endif
   printf("
```

語法

```
dsInt16_t dsmRCMsg  (dsUint32_t       dsmHandle,
   dsInt16_t              dsmRC,
   char        *msg);
```

參數

dsUint32_t dsmHandle (I)
將此呼叫與先前的 **dsmInitEx** 呼叫結合的 handle。

dsInt16_t dsmRC (I)
相關訊息文字的 API 回覆碼。 API 回覆碼是列示於 dsmrc.h 檔案中。 如需相關資訊，請參閱 第 123 頁的『附錄 A API 回覆碼原始檔：dsmrc.h』。

char *msg (O)
此參數是與回覆碼 **dsmRC** 相關的訊息文字。 呼叫者必須負責為訊息文字配置足夠的空間。

msg 的最大長度是定義成 DSM_MAX_RC_MSG_LENGTH。

在具有「國家語言支援」以及可選擇語言訊息檔的平台上，API 會以國家語言傳回訊息字串。

回覆碼

回覆碼的號碼會用括弧 () 括住。

表 *43. dsmRCMsg* 的回覆碼

回覆碼	說明
DSM_RC_NULL_MSG (2002)	dsmRCMsg 呼叫的 **msg** 參數 是 NULL 指標。
DSM_RC_INVALID_RETCODE (2021)	傳遞給 **dsmRCMsg** 呼叫的回覆碼是無效的代碼。
DSM_RC_NLS_CANT_OPEN_TXT (0610)	無法開啟訊息本文檔。

dsmRegisterFS

dsmRegisterFS 函數呼叫向 IBM Spectrum Protect 伺服器登錄新的檔案空間。 登錄檔案空間之後才能將資料備份到該伺服器。

應用程式用戶端不應該使用備份保存用戶端會使用的相同檔案空間名稱。

- 在 UNIX 或 Linux 上，執行 **df** 指令可取得這些名稱。
- 在 Windows 上，這些名稱通常是與系統各個磁碟機相關聯的磁區標籤。

語法

```
dsInt16_t dsmRegisterFS
  (dsUint32_t       dsmHandle,
   regFSData     *regFilespaceP);
```

參數

dsUint32_t dsmHandle (I)
將此呼叫與先前的 **dsmInitEx** 呼叫結合的 handle。

regFSData *regFilespaceP (I)
此參數會傳送檔案空間名稱以及您向 IBM Spectrum Protect 伺服器登錄所需的相關資訊。

提示: *fstype* 欄位包括字首 **"API:"**。 所有檔案空間查詢都會顯示此字串。 例如，如果使用者在 **dsmRegisterFS** 中傳遞 *myfstype* 作為 *fstype*，則查詢時，伺服器上的實際值字串會傳回 API:myfstype。 此字首可區分 API 物件以及備份保存物件。

fsInfo 的可用區域現在是 DSM_MAX_USER_FSINFO_LENGTH。

回覆碼

回覆碼的號碼會用括弧 () 括住。

表 *44. dsmRegisterFS* 的回覆碼

回覆碼	說明
DSM_RC_INVALID_FSNAME (2016)	無效的檔案空間名稱。
DSM_RC_INVALID_DRIVE_CHAR (2026)	磁碟機字母不是英文字母。
DSM_RC_NULL_FSNAME (2027)	空的檔案空間名稱。
DSM_RC_FS_ALREADY_REGED (2062)	檔案空間已登錄。
DSM_RC_WRONG_VERSION_PARM (2065)	應用程式用戶端的 API 版本與 IBM Spectrum Protect 程式庫版本不同。
DSM_RC_FSINFO_TOOLONG (2106)	檔案空間資訊太長。

dsmReleaseBuffer

dsmReleaseBuffer 函數會將緩衝區傳回 IBM Spectrum Protect。應用程式會先呼叫 **dsmGetDataEx**，並將所有資料移出緩衝區來準備釋放緩衝區，然後再呼叫 **dsmReleaseBuffer**。 **dsmReleaseBuffer** 需要將 ***UseTsmBuffers*** 設為 *btrue* 並為 *numTsmBuffers* 提供非零值，來呼叫 **dsmInitEx**。 如果應用程式即將呼叫 **dsmTerminate** 但仍保有資料緩衝區，則也應該呼叫 **dsmReleaseBuffer**。

dsmReleaseBufferSyntax

```
dsInt16_t  dsmReleaseBuffer   (releaseBufferIn_t        *dsmReleaseBufferInP,
                               releaseBufferOut_t       *dsmReleaseBufferOutP) ;
```

參數

releaseBufferIn_t * dsmReleaseBufferInP (I)
此結構包含下列輸入參數。

dsUint32_t dsmHandle (I)
將此呼叫與先前的 **dsmInitEx** 呼叫相結合的 handle。

dsUint8_t tsmBufferHandle(I)
識別此緩衝區的 handle。

char *dataPtr(I)
應用程式寫入的位址。

回覆碼

回覆碼的號碼會用括弧 () 括住。

表 *45. dsmReleaseBuffer* 的回覆碼

回覆碼	說明
DSM_RC_BAD_CALL_SEQUENCE	此呼叫不是在適當狀態下發出。
DSM_RC_INVALID_TSMBUFFER	**dataPtr** 的值或 handle 無效。
DSM_RC_BUFF_ARRAY_ERROR	發生緩衝區陣列錯誤。

dsmRenameObj

dsmRenameObj 函數呼叫會將高階或低階物件名稱加以更名。 若是備份物件，則傳入現行物件名稱，並且變更高階或低階物件名稱。 對於保存物件來說，會傳入現行物件檔案空間名稱和物件 ID，並變更高階或低階物件名稱。 請在 **dsmBeginTxn** 和 **dsmEndTxn** 呼叫內使用此函數呼叫。

合併旗號決定是否將重複的備份物件名稱與現有的備份合併。 如果新名稱對應到現有的物件，而且合併為真，則現行物件會轉換成新名稱，並且變成新名稱的作用中版本，而具有該名稱的現有作用中物件則成為最上方的物件非作用副本。 如果新名稱對應到現有的物件，而且合併為假，則函數會傳回回覆碼 DSM_RC_ABORT_DUPLICATE_OBJECT。

限制:

- 只有物件的擁有者可以將物件更名。
- 如果在 IBM Spectrum Protect 伺服器上啟用了資料保留保護，或者如果您已連接至 IBM Spectrum Protect for Data Retention 伺服器，則 **dsmRenameObj** 函數不受支援。

dsmRenameObj 函數呼叫會測試下列合併條件：

- 目前的 **dsmObjName** 物件以及新的高階或低階物件在擁有者、 副本群組和管理類別上必須相符。
- 目前的 **dsmObjName** 的備份時間必須 比目前作用中、具新名稱的物件更近。
- 只能有一個目前 **dsmObjName** 的作用中副本，而不能有非作用的副本。

語法

```
dsInt16_t dsmRenameObj  (dsmRenameIn_t       *dsmRenameInP,
                         dsmRenameOut_t      *dsmRenameOutP);
```

參數

dsUint32_t dsmHandle (I)
將此呼叫與先前的 **dsmInitEx** 呼叫結合的 handle。

dsmRenameIn_t *dsmRenameInP
此結構包含輸入參數。

dsUint8_t repository (I);
此參數指出要刪除的檔案空間是在備份儲存庫或保存儲存庫。

dsmObjName *objNameP (I);
此參數是結構的指標，其中包含現行檔案空間名稱、高階物件名稱、低階物件名稱以及物件類型。

char newHl [DSM_MAX_HL_LENGTH + 1];
此參數指定新的高階名稱。

char newLl [DSM_MAX_LL_LENGTH + 1];
此參數指定新的低階名稱。

dsBool_t merge;
此參數決定備份物件是否與重複的指名物件合併。 其值為 true 或 false。

ObjID;
保存物件的物件 ID。

dsmRenameOut_t *dsmRnameOutP
此結構包含輸出參數。

註: 沒有輸出參數。

回覆碼

回覆碼的號碼會用括弧 () 括住。

表 *46. dsmRenameObj* 的回覆碼

回覆碼	說明
DSM_RC_ABORT_MERGE_ERROR (45)	伺服器偵測到合併錯誤。
DSM_RC_ABORT_DUPLICATE_OBJECT (32)	物件已經存在，合併為假（false）。
DSM_RC_ABORT_NO_MATCH (2)	找不到物件。
DSM_RC_REJECT_SERVER_DOWNLEVEL (58)	IBM Spectrum Protect 伺服器必須是 3.7.4.0 版或更高版本才能讓此函數生效。

dsmRequestBuffer

dsmRequestBuffer 函數會將緩衝區傳回 IBM Spectrum Protect。應用程式會先 呼叫 **dsmGetDataEx**，並將所有資料移出緩衝區來準備釋放緩衝區，然後再呼叫 **dsmRequestBuffer**。

dsmReleaseBuffer 需要以設為 *btrue* 的 *UseTsmBuffers* 和提供的非零值的 *numTsmBuffers* 來呼叫 **dsmInitEx**。 如果應用程式即將呼叫 **dsmTerminate** 但仍保有 IBM Spectrum Protect 緩衝區，則也應該呼叫 **dsmReleaseBuffer**。

語法

```
dsInt16_t  dsmRequestBuffer   (getBufferIn_t        *dsmRequestBufferInP,
                               getBufferOut_t       *dsmRequestBufferOutP) ;
```

參數

getBufferIn_t * dsmRequestBufferInP (I)
此結構包含下列輸入參數：

dsUint32_t dsmHandle
識別階段作業並使它與先前 **dsmInitEx** 呼叫產生關聯的 handle。

getBufferOut_t *dsmRequestBufferOut P (0)
此結構包含輸出參數。

dsUint8_t tsmBufferHandle(0)
識別此緩衝區的 handle。

char *dataPtr(0)
應用程式寫入的位址。

dsUint32_t *bufferLen(0)
可寫入此緩衝區的最大位元組數。

回覆碼

回覆碼的號碼會用括弧 () 括住。

表 *47. dsmRequestBuffer* 的回覆碼

回覆碼	說明
DSM_RC_BAD_CALL_SEQUENCE (33)	此呼叫不是在適當狀態下發出。
DSM_RC_SENDDATA_WITH_ZERO_SIZE (34)	如果傳送的物件長度為 0，則不允許呼叫 **dsmReleaseBuffer**。
DSM_RC_BUFF_ARRAY_ERROR (121)	無法取得有效的緩衝區。

dsmRetentionEvent

dsmRetentionEvent 函數呼叫會將一個物件 ID 清單傳送至 IBM Spectrum Protect 伺服器，並將對這些物件執行保留事件作業。請在 **dsmBeginTxn** 和 **dsmEndTxn** 呼叫內使用此函數呼叫。

註: 伺服器必須是 5.2.2.0 版或更高版本才能讓此函數生效。

一個呼叫的最大物件數限制為 *maxObjPerTxn* 的值， 該值是從 **dsmQuerySessInfo** 呼叫的 *ApisessInfo* 結構中傳回的值。

只有物件的擁有者可以對該物件傳送事件。

以下是可能的事件：

eventRetentionActivate
只能對連結到事件型管理類別的物件發出。 傳送這個事件會對此物件啟動該事件，且此物件的保留狀態會從 DSM_ARCH_RETINIT_PENDING 變更為 DSM_ARCH_RETINIT_STARTED。

eventHoldObj
這個事件會對物件發出保留或刪除保留，所以在發出釋放之前，此物件不會過期且無法刪除。

eventReleaseObj
這個事件只能對 ***objectHeld*** 欄位中有 DSM_ARCH_HELD_TRUE 值的物件發出，它會移除對物件的保留限制，回復原始的保留原則。

在傳送 ***dsmRetentionEvent*** 之前，請先傳送第 26 頁的『查詢 IBM Spectrum Protect 系統』中說明的查詢順序，以取得物件的資訊。 **dsmGetNextQObj** 呼叫會針對保存查詢傳回名稱為 **qryRespArchiveData** 的資料結構。 此資料結構包含 **dsmRetentionEvent** 需要的資訊。

語法

```
extern dsInt16_t DSMLINKAGE dsmRetentionEvent(
  dsmRetentionEventIn_t          *ddsmRetentionEventInP,
  dsmRetentionEventOut_t         *dsmRetentionEventOutP
   );
```

參數

dsmRetentionEventIn_t *dsmRetentionEventP
此結構包含下列輸入參數：

dsUint16_t stVersion;
此參數指示結構版本。

dsUint32_t dsmHandle (I)
將此呼叫與先前的 **dsmInitEx** 呼叫相結合的 handle。

dsmEventType_t evenType (I);
此參數指示事件類型。 請參閱本節開頭，以瞭解這些可能值的意義：**eventRetentionActivate**、**eventHoldObj**、**eventReleaseObj**

dsmObjList_t objList;
此參數指示要表示的物件 ID 清單。

回覆碼

回覆碼的號碼會用括弧 () 括住。

表 *48. dsmRetentionEvent* 的回覆碼

回覆碼	說明
DSM_RC_ABORT_NODE_NOT_AUTHORIZED (36)	節點或使用者沒有適當的權限。
DSM_RC_ABORT_TXN_LIMIT_EXCEEDED (249)	異動中有太多物件。
DSM_RC_ABORT_OBJECT_ALREADY_HELD (250)	物件已保留，無法發出另一個保留。
DSM_RC_REJECT_SERVER_DOWNLEVEL (58)	伺服器必須是 5.2.2.0 版或更高版本才能讓此函數生效。

dsmSendBufferData

dsmSendBufferData 函數呼叫會透過前一個 **dsmReleaseBuffer** 呼叫中所提供的緩衝區，將資料的位元組串流傳送至 IBM Spectrum Protect。應用程式用戶端可以傳送任何類型的資料，以便儲存在伺服器上。此資料通常是檔案資料，但不限於一定是檔案資料。 如果您所要傳送資料的位元組串流很大，您可以重複呼叫 **dsmSendBufferData** 數次。 無論呼叫成功與否，都會釋放緩衝區。

限制: 當使用 *useTsmBuffers* 選項時，即使物件已併入壓縮作業中，該物件還是不會被壓縮。

語法

```
dsInt16_t  dsmSendBufferData   (sendBufferDataIn_t        *dsmSendBufferDataExInP,
                                sendBufferDataOut_t       *dsmSendBufferDataOutP) ;
```

參數

sendBufferDataIn_t * dsmSendBufferDataInP (I)
此結構包含下列輸入參數。

dsUint32_t dsmHandle (I)
將此呼叫與先前的 **dsmInitEx** 呼叫相結合的 handle。

dsUint8_t tsmBufferHandle(I)
識別要傳送之緩衝區的 handle。

char *dataPtr(I)
應用程式資料寫入的位址。

dsUint32_t numBytes(I)
應用程式寫入的實際位元組數（一定要小於 **dsmReleaseBuffer** 提供的值）。

回覆碼

回覆碼的號碼會用括弧 () 括住。

表 *49. dsmSendBufferData* 的回覆碼

回覆碼	說明
DSM_RC_BAD_CALL_SEQUENCE (2041)	此呼叫不是在適當狀態下發出。
DSM_RC_INVALID_TSMBUFFER (2042)	**dataPtr** 的值或 handle 無效。
DSM_RC_BUFF_ARRAY_ERROR (2045)	發生緩衝區陣列錯誤。
DSM_RC_TOO_MANY_BYTES (2043)	***numBytes*** 的值大於 **dsmReleaseBuffer** 呼叫提供的緩衝區大小。

dsmSendData

dsmSendData 函數呼叫會透過緩衝區將資料的位元組串流傳送至 IBM Spectrum Protect。 應用程式用戶端可以傳送任何類型的資料，以便儲存在伺服器上。 這些資料通常是檔案資料，但並不限於此。 如果您要傳送的資料的位元組串流很大，您可以重複多次呼叫 **dsmSendData** 。

限制: 在 **dsmSendData** 呼叫傳回之前，應用程式用戶端無法重複使用 **dsmSendData** 指定的緩衝區。

提示: 如果 IBM Spectrum Protect 傳回代碼 157 (DSM_RC_WILL_ABORT)，請呼叫 **dsmEndSendObj** ，然後使用 DSM_VOTE_COMMIT 的 vote 呼叫 **dsmEndTxn** 。 然後應用程式會收到回覆碼 2302 （DSM_RC_CHECK_REASON_CODE），並且將原因碼傳回給應用程式使用者。這會通知使用者伺服器為何結束交易。

語法

```
dsInt16_t dsmSendData
  (dsUint32_t       dsmHandle,
   DataBlk *dataBlkPtr);
```

參數

dsUint32_t dsmHandle (I)
將此呼叫與先前的 **dsmInitEx** 呼叫結合的 handle。

DataBlk *dataBlkPtr (I/O)
此參數指向一個結構，其中包含一個指標（指向要傳送資料的緩衝區）以及緩衝區的大小。 傳回時，此結構包含實際轉送的位元組數目。 請參閱第 133 頁的『附錄 B API 類型定義原始檔』以取得類型定義。

回覆碼

回覆碼的號碼會用括弧 () 括住。

表 *50. dsmSendData* 的回覆碼

回覆碼	說明
DSM_RC_NO_COMPRESS_MEMORY (154)	沒有足夠的記憶體可執行資料壓縮或展開。
DSM_RC_COMPRESS_GREW (155)	在壓縮期間，相較於原始資料，被壓縮的資料會改變其大小。
DSM_RC_WILL_ABORT (157)	發生不明或異常的錯誤，導致交易中止。
DSM_RC_WRONG_VERSION_PARM (2065)	應用程式用戶端的 API 版本與 IBM Spectrum Protect 程式庫版本不同。
DSM_RC_NEEDTO_ENDTXN (2070)	需要結束交易。
DSM_RC_OBJ_EXCLUDED (2080)	併入/排除清單排除物件。

表 *50. dsmSendData* 的回覆碼 *(繼續)*

回覆碼	說明
DSM_RC_OBJ_NOBCG (2081)	物件沒有備份副本群組，因此不會傳送至伺服器。
DSM_RC_OBJ_NOACG (2082)	物件沒有保存副本群組，因此不會傳送至伺服器。
DSM_RC_SENDDATA_WITH_ZERO_SIZE (2107)	物件無法傳送 *sizeEstimate* 為零位元組的資料。

dsmSendObj

dsmSendObj 函數呼叫會啟動將單一物件傳送儲存體的要求。 您可以在許多連結的交易中執行多個 **dsmSendObj** 呼叫以及相關的 **dsmSendData** 呼叫，以增進效能。

dsmSendObj 呼叫會將物件的資料當作傳入記憶體緩衝區的位元組串流來處理。 **dsmSendObj** 呼叫中的 **dataBlkPtr** 參數允許應用程式用戶端執行下列作業：

- 在單一呼叫中傳送物件的資料及屬性（屬性是透過 **objAttrPtr** 來傳送）。
- 透過 **dsmSendObj** 呼叫來指定部分的物件資料，以及透過一或多次的 **dsmSendData** 呼叫來指定剩餘的資料。

另外，應用程式用戶端只能透過 **dsmSendObj** 呼叫來指定屬性，以及透過對 **dsmSendData** 的一次或多次呼叫來指定物件資料。 若使用這種方法，請在 **dsmSendObj** 呼叫中將 **dataBlkPtr** 設定為 NULL。

提示: 在特定的物件類型中，位元組資料串流可能不會與資料結合；例如，沒有延伸屬性的目錄。

呼叫 **dsmSendObj** 之前，必須先呼叫 **dsmBindMC** ，才能將管理類別適當地連結到您要備份或保存的物件。 API 會維持這個連結，使得連結可以將適當的管理類別以與傳送至伺服器的物件加以結合。 如果允許 **dsmSendObj** 呼叫所連結的管理類別作為類型為目錄 (DSM_OBJ_DIRECTORY) 的物件的預設值，則預設值可能不是預設的管理類別。 反之，會使用具有最高保留時間的管理類別。 如果有一個以上的管理類別具有此保留時間，則使用第一個發現的管理類別。

請在傳送到儲存體的所有物件資料後面執行 **dsmEndSendObj** 呼叫。 如果沒有物件資料可傳送到伺服器，或者所有的資料都是包含在 **dsmSendObj** 呼叫中，請先啟動一個 **dsmEndSendObj** 呼叫，然後再啟動另一個 **dsmSendObj** 呼叫。 如果必須經由 **dsmSendData** 呼叫進行多個資料傳送，**dsmEndSendObj** 會接在最後一個傳送之後，以指出狀態變更。

提示: 如果 IBM Spectrum Protect 傳回代碼 157 (DSM_RC_WILL_ABORT)，請使用 DSM_VOTE_COMMIT 的 vote 呼叫 **dsmEndTxn** 。 應用程式會收到回覆碼 2302（DSM_RC_CHECK_REASON_CODE），並且將原因碼傳回給應用程式使用者。這會通知使用者伺服器為何結束交易。

如果原因碼是 11（DSM_RS_ABORT_NO_REPOSIT_SPACE），則 *sizeEstimate* 對於實際的資料數量可能太小。 應用程式需要決定更精確的 *sizeEstimate*，然後重新傳送資料。

語法

```
dsInt16_t dsmSendObj
  (dsUint32_t       dsmHandle,
   dsmSendType            sendType,
   void                  *sendBuff,
   dsmObjName             *objNameP,
   ObjAttr               *objAttrPtr,
   DataBlk *dataBlkPtr);
```

參數

dsUint32_t dsmHandle (I)
將此呼叫與先前的 **dsmInitEx** 呼叫結合的 handle。

dsmSendType sendType (I)
此參數會指定執行的傳送類型。 可能的值包括：

名稱	說明
stBackup	傳送到伺服器的備份物件。
stArchive	傳送到伺服器的保存物件。
stBackupMountWait	您要伺服器等待，直到裝載必要的裝置（例如磁帶）之後的備份物件。
stArchiveMountWait	您要伺服器等待，直到裝載必要的裝置（例如磁帶）之後的保存物件。

註: 如果您的應用程式使用者可能傳送資料到磁帶，請使用 **MountWait** 類型。

void *sendBuff (I)
此參數是結構的指標，其中包含呼叫的 **sendType** 的其他特定資訊。 目前只有 **stArchive** 的 **sendType** 具有相關的結構。 此結構稱為 **sndArchiveData**，而且包含保存說明。

dsmObjName *objNameP (I)
此參數是結構的指標，其中包含檔案空間名稱、高階物件名稱、低階物件名稱以及物件類型。 如需相關資訊，請參閱 第 18 頁的『物件名稱和 ID』。

ObjAttr *objAttrPtr (I)
此參數會將想要的物件屬性傳遞到應用程式。 請參閱第 133 頁的『附錄 B API 類型定義原始檔』以取得類型定義。

屬性如下：

- **owner** 是指物件的擁有者。 從 IBM Spectrum Protect 儲 存體取回物件時，決定要宣告擁有者是特定名稱或空字串很重要。 如需相關資訊，請參閱 第 19 頁的『以階段作業擁有者的身分存取物件』。
- **sizeEstimate** 是要傳送到伺服器的資料物件總計的最佳預估值。 盡可能使用最精確的大小，因為伺服器會使用這個屬性，在其儲存體資源中進行有效率的空間配置以及物件放置。

  如果您指定的大小預估比實際傳送的位元組數目低很多，伺服器可能無法配置足夠的空間，而且會以原因碼 11（DSM_RS_ABORT_NO_REPOSIT_SPACE）結束交易。

  註: 大小預估 是就資料物件的總計大小而言（以位元組為單位）。

  小於 DSM_MIN_COMPRESS_SIZE 的物件不會壓縮。

  如果您的物件沒有位元資料（只有來自此呼叫的屬性資訊），則 **sizeEstimate** 應該為零。

  註: 從 5.1.0 版開始，交易內的複製目的地不會再被檢查有關零長度物件的一致性。
- **objCompressed** 是一個布林值，用來指出物件資料是否已經壓縮。

  如果物件已被壓縮 (object *compressed=bTrue*)，IBM Spectrum Protect 就不會再壓縮它。 如果 物件未被壓縮，IBM Spectrum Protect 會根據管理者設定的 壓縮選項的值以及 API 配置來源中的設定來決定是否壓縮物件。

  如果您的應用程式打算使用部分物件還原或擷取，您就不能在傳送資料時壓縮資料。 如果要強制執行此動作，請將 *ObjAttr.objCompressed* 設定為 *bTrue*。
- **objInfo** 可儲存特定物件的相關資訊。

  限制: 資訊並不會自動儲存到此處。 使用這個屬性時，必須設定 *objInfoLength* 屬性為顯示 *objInfo* 的長度。
- **mcNameP** 包含一個管理類別名稱，這個管理類別會置換取自 **dsmBindMC** 的管理類別。
- **disableDeduplication** 是一個布林值。 當它設為 true 時，用戶端不會刪除此物件的重複資料。

DataBlk *dataBlkPtr (I/O)
此參數指向一個結構，其中包含一個指標（指向要備份或保存的資料緩衝區）以及該緩衝區的大小。 此參數僅適用於 **dsmSendObj** 。 如果您要在後續的 **dsmSendData** 呼叫開始傳送資料，而不要在 **dsmSendObj** 呼叫傳送，請將 DataBlk 結構中的緩衝區指標設定為 NULL。 傳回時，此結構包含實際轉送的位元組數目。 請參閱第 133 頁的『附錄 B API 類型定義原始檔』以取得類型定義。

回覆碼

回覆碼的號碼會用括弧 () 括住。

表 *51. dsmSendObj* 的回覆碼

回覆碼	說明
DSM_RC_NO_COMPRESS_MEMORY (154)	沒有足夠的記憶體可執行資料壓縮或展開。
DSM_RC_COMPRESS_GREW (155)	在壓縮期間，相較於原始資料，被壓縮的資料會改變其大小。
DSM_RC_WILL_ABORT (157)	發生不明或異常的錯誤，導致交易中止。
DSM_RC_TL_NOACG (186)	此檔案的管理類別不具備傳送類型的有效副本群組。
DSM_RC_NULL_OBJNAME (2000)	物件名稱為空值。
DSM_RC_NULL_OBJATTRPTR (2004)	物件屬性指標為空值。
DSM_RC_INVALID_OBJTYPE (2010)	無效的物件類型。
DSM_RC_INVALID_OBJOWNER (2019)	無效的物件擁有者。
DSM_RC_INVALID_SENDTYPE (2022)	無效的傳送類型。
DSM_RC_WILDCHAR_NOTALLOWED (2050)	不容許使用萬用字元。
DSM_RC_FS_NOT_REGISTERED (2061)	檔案空間未登錄。
DSM_RC_WRONG_VERSION_PARM (2065)	應用程式用戶端的 API 版本與 IBM Spectrum Protect 程式庫版本不同。
DSM_RC_NEEDTO_ENDTXN (2070)	需要結束交易。
DSM_RC_OBJ_EXCLUDED (2080)	併入/排除清單已排除物件。
DSM_RC_OBJ_NOBCG (2081)	物件沒有備份副本群組，因此不會傳送至伺服器。
DSM_RC_OBJ_NOACG (2082)	物件沒有保存副本群組，因此不會傳送至伺服器。
DSM_RC_DESC_TOOLONG (2100)	說明太長。
DSM_RC_OBJINFO_TOOLONG (2101)	物件資訊太長。
DSM_RC_HL_TOOLONG (2102)	高階限定元太長。
DSM_RC_FILESPACE_TOOLONG (2104)	檔案空間名稱太長。
DSM_RC_LL_TOOLONG (2105)	低階限定元太長。
DSM_RC_NEEDTO_CALL_BINDMC (2301)	必須先呼叫 **dsmBindMC**。

dsmSetAccess

dsmSetAccess 函數呼叫會為其他使用者或節點提供對物件備份版本或保存副本的存取權、 對所有物件的存取權，或者對選擇集的存取權。 當您提供存取權給另一個使用者時，該使用者就可以查詢、還原或擷取您的檔案。 這個指令可支援在下列欄位使用萬用字元：*fs*、 *hl*、*ll*、 *node* 和 *owner*。

註: 您不能使用單一指令來提供存取權給備份版本和保存副本兩者。 您必須指定 備份或保存。

語法

```
dsInt16_t DSMLINKAGE dsmSetAccess
        (dsUint32_t        dsmHandle,
         dsmSetAccessType    accessType,
         dsmObjName            *objNameP,
         char                  *node,
         char                  *owner
);
```

參數

dsUint32_t dsmHandle (I)
將此呼叫與先前的 **dsmInitEx** 呼叫結合的 handle。

dsmAccessType accessType (I)
此參數會指定您要提供存取權的物件類型。 可能的值包括：

名稱	說明
atBackup	指定存取權是設定成備份物件。
atArchive	指定存取權為保存物件所設定。

dsmObjName *objNameP (I)
此參數是結構的指標，其中包含檔案空間名稱、高階物件名稱以及低階物件名稱。

註: 如果要指定所有檔案空間，請使用星號 (*) 代表檔案空間名稱。

char *node (I)
此參數是指向給定存取權的節點名稱的指標。 若是任何節點，請指定星號（*）。

char *owner (I)
此參數是指向給定存取權的節點上的使用者名稱的指標。 若是任何使用者，請指定星號（*）。

回覆碼

回覆碼的號碼會用括弧 () 括住。

表 *52. dsmSetAccess* 的回覆碼

回覆碼	說明
DSM_RC_INVALID_ACCESS_TYPE (2110)	指定的存取類型無效。
DSM_RC_FILE_SPACE_NOT_FOUND (124)	在伺服器找不到指定的檔案空間。
DSM_RC_QUERY_COMM_FAILURE (2111)	在伺服器查詢期間發生通訊錯誤。
DSM_RC_NO_FILES_BACKUP (2112)	此檔案空間沒有備份的檔案。
DSM_RC_NO_FILES_ARCHIVE (2113)	此檔案空間沒有保存的檔案。
DSM_RC_INVALID_SETACCESS (2114)	設定存取權的公式無效。

dsmSetUp

dsmSetUp 函數呼叫會改寫環境變數值。 請先呼叫 **dsmSetUp**，然後再呼叫 **dsmInitEx**。 傳入 **envSetUp** 結構中的值會改寫任何現有的環境變數或預設值。 如果為欄位指定 NULL，就會從環境取得這些值。 如果您沒有設定任何值，就會從預設值取得這些值。

需求:

1. 如果使用 **dsmSetUp**，請務必先呼叫 **dsmTerminate**，然後再呼叫 **dsmCleanUp**。
2. 只有已在配置檔中設定測試旗標 INSTRUMENT: API 並在應用程式中使用 **dsmSetUp** 或 **dsmCleanUp** 呼叫時，才能啟動 API 設備測試。

語法

```
dsInt16_t DSMLINKAGE dsmSetUp
        (dsBool_t    mtFlag,
         envSetUp    *envSetUpP);
```

參數

dsBool_t mtFlag (I)
此參數指定 API 是在單一執行緒或多執行緒模式中使用。 這些值包括：

```
DSM_SINGLETHREAD
DSM_MULTITHREAD
```

需求: 必須開啟多執行緒旗標才能進行不需 LAN 資料傳送。

envSetUp *envSetUpP(I)
此參數是指向結構的指標，其中包含改寫值。 如果您不要置換現有的環境變數，請指定 NULL。
envSetUp 結構中的欄位包括：

名稱	說明
dsmiDir	在 UNIX 或 Linux 中包含訊息檔的完整目錄路徑。 它也會指定 dsm.sys 目錄。
dsmiConfig	用戶端選項檔的完整名稱。
dsmiLog	錯誤日誌目錄的完整路徑。
argv	如果應用程式必須以授權使用者權限來執行，則傳遞呼叫程式的 argv[0] 名稱。如需相關資訊，請參閱 第 17 頁的『控制對密碼檔的存取權』。
logName	若應用程式未使用 dsierror.log，則為錯誤日誌的檔名。
inclExclCaseSensitive	指出併入/排除規則要區分大小寫或不區分大小寫。 此參數只能在 Windows 上使用，不適用於其他作業系統。

回覆碼

回覆碼的號碼會用括弧 () 括住。

表 *53. dsmSetUp* 的回覆碼

回覆碼	說明
DSM_RC_ACCESS_DENIED (106)	存取指定的檔案或目錄遭拒。
DSM_RC_INVALID_OPT (0400)	發現某個無效的選項。
DSM_RC_NO_HOST_ADDR (0405)	此伺服器的 TCPSERVERADDRESS 未定義在系統選項檔的伺服器名稱一節中。
DSM_RC_NO_OPT_FILE (0406)	找不到 filename 所指定的選項檔。
DSM_RC_MACHINE_SAME (0408)	選項檔中定義的 NODENAME 不能與系統 *HostName* 相同。
DSM_RC_INVALID_SERVER (0409)	系統選項檔未包含 SERVERNAME 選項。
DSM_RC_INVALID_KEYWORD (0410)	在 **dsmInitEx** 配置檔、選項字串、dsm.sys 或 dsm.opt 中找到無效的選項關鍵字。
DSM_RC_PATTERN_TOO_COMPLEX (0411)	發出的併入或排除型樣太複雜，IBM Spectrum Protect 無法正確地解譯。
DSM_RC_NO_CLOSING_BRACKET (0412)	併入或排除型樣的建構不正確。 缺少結束的方括弧（]）。

表 53. *dsmSetUp* 的回覆碼 (繼續)

回覆碼	說明
DSM_RC_NLS_CANT_OPEN_TXT (0610)	系統無法開啟訊息本文檔。
DSM_RC_NLS_INVALID_CNTL_REC (0612)	系統無法使用訊息本文檔。
DSM_RC_NOT_ADSM_AUTHORIZED (0927)	您必須是授權使用者，才能有多執行緒和 *passwordaccess* generate。
DSM_RC_NO_INCLEXCL_FILE (2229)	找不到納入-排除檔案。
DSM_RC_NO_SYS_OR_INCLEXCL (2230)	找不到 dsm.sys 或納入-排除檔案。

dsmTerminate

dsmTerminate 函數呼叫結束與 IBM Spectrum Protect 伺服器的階段作業並清除 IBM Spectrum Protect 環境。

語法

這個呼叫沒有特定的回覆碼。

```
dsInt16_t dsmTerminate  (dsUint32_t dsmHandle);
```

參數

dsUint32_t dsmHandle (I)
將此呼叫與先前的 **dsmInitEx** 呼叫結合的 handle。

dsmUpdateFS

dsmUpdateFS 函數呼叫會更新 IBM Spectrum Protect 儲存體中的檔案空間。 這項更新可確保管理者有您的檔案空間的現行記錄。

語法

```
dsInt16_t dsmUpdateFS
  (dsUint32_t       dsmHandle,
   char                  *fs,
   dsmFSUpd              *fsUpdP,
   dsUint32_t            fsUpdAct
);
```

參數

dsUint32_t dsmHandle (I)
將此呼叫與先前的 **dsmInitEx** 呼叫結合的 handle。

char *fs (I)
此參數是檔案空間名稱的指標。

dsmFSUpd *fsUpdP (I)
此參數是指向結構的指標，其中包含您要更新的正確欄位。 您只需要完成需要更新的那些欄位。

dsUint32_t fsUpdAct (I)
一個具有兩個位元組的點陣圖，用來指出有哪些欄位要更新。 位元遮罩有下列值：

- DSM_FSUPD_FSTYPE
- DSM_FSUPD_FSINFO

提示: 在 Windows 作業系統上，選取 **FSINFO** 時，也會更新取自 **dsmDOSAttrib** 的磁碟機代號值。

- DSM_FSUPD_OCCUPANCY
- DSM_FSUPD_CAPACITY
- DSM_FSUPD_BACKSTARTDATE
- DSM_FSUPD_BACKCOMPLETEDATE

如需位元遮罩的說明，請參閱下列主題中的 DSM_FSUPD 定義：第 133 頁的『附錄 B API 類型定義原始檔』。

回覆碼

下表列出 **dsmUpdateFS** 函數呼叫的回覆碼。

表 *54. dsmUpdateFS* 的回覆碼

回覆碼	回覆碼號碼	說明
DSM_RC_FS_NOT_REGISTERED	2061	檔案空間名稱未登錄。
DSM_RC_WRONG_VERSION_PARM	2065	應用程式用戶端的 API 版本與 IBM Spectrum Protect 媒體庫版本不同。
DSM_RC_FSINFO_TOOLONG	2106	檔案空間資訊太長。

dsmUpdateObj

dsmUpdateObj 函數呼叫會更新與伺服器作用中備份或保存物件相關的 meta 資訊。 應用程式位元資料將不會受到影響。 如果要更新物件，您必須提供特定的非萬用字元名稱。 如果要更新保存物件，請將 **dsmSendType** 設定為 **stArchive**。 只有最新的指名保存物件會被更新。

在階段作業狀態中，您只能啟動 **dsmUpdateObj** 呼叫；您無法從交易內部呼叫它，因為它會執行本身的交易。 而且，您一次只能更新一個物件。

限制: 在 UNIX 或 Linux 作業系統上，如果變更擁有者欄位，除非您是 root 使用者，否則就無法查詢或還原物件。

語法

```
dsInt16_t  dsmUpdateObj
   (dsUint32_t        dsmHandle,
    dsmSendType           sendType,
    void                  *sendBuff,
    dsmObjName            *objNameP,
    ObjAttr        *objAttrPtr,    /* objInfo */
    dsUint16_t      objUpdAct);  /* action bit vector */
```

參數

欄位說明和 **dsmSendObj** 中的欄位說明相同，但下列項目除外：

dsmObjName *objNameP (I)
您不能使用萬用字元。

ObjAttr *objAttrPtr (I)
在此呼叫中，**objCompressed** 欄位會被忽略。

其他的差異是：

- **owner**。 如果您指定新的 **owner** 欄位，擁有者就會改變。

- **sizeEstimate**。 如果您指定非零的值，它應該就是實際傳送的資料數量（以位元組為單位）。 此值會儲存在 IBM Spectrum Protect meta 資料以供將來使用。
- **objInfo**。 這個屬性包含要放置在 **objInfo** 欄位中的新資訊。 請將 **objInfoLength** 設定成新 **obiInfo** 的長度。

dsUint16_t objUpdAct
objUpdAct 的位元遮罩和可能的動作包括：

DSM_BACKUPD_MC
更新物件的管理類別。

DSM_BACKUPD_OBJINFO
更新 **objInfo**、**objInfoLength** 和 **sizeEstimate**。

DSM_BACKUPD_OWNER
更新物件的擁有者。

DSM_ARCHUPD_DESCR
更新**說明**欄位。 透過 **SendBuff** 參數來輸入新說明的值。 請參閱範例程式以取得適當的用法。

DSM_ARCHUPD_OBJINFO
更新 **objInfo**、**objInfoLength** 和 **sizeEstimate**。

DSM_ARCHUPD_OWNER
更新物件的擁有者。

回覆碼

回覆碼的號碼會用括弧 () 括住。

表 *55. dsmUpdateObj* 的回覆碼

回覆碼	說明
DSM_RC_INVALID_ ACTION (2232)	無效的動作。
DSM_RC_FS_NOT_REGISTERED (2061)	檔案空間未登錄。
DSM_RC_BAD_CALL_SEQUENCE (2041)	呼叫順序無效。
DSM_RC_WILDCHAR_NOTALLOWED (2050)	不容許使用萬用字元。
DSM_RC_ABORT_NO_MATCH (2)	前一個查詢不符。

dsmUpdateObjEx

dsmUpdateObjEx 函數呼叫會更新與伺服器上的作用中備份或保存物件相關的 meta 資訊。 應用程式位元資料將不會受到影響。 如果要更新物件，您必須指定非萬用字元名稱，或者，也可以指定物件 ID 來更新特定的保存物件。 在指定名稱時不能使用萬用字元。 如果要更新備份物件，請將 **dsmSendType** 參數設為 **stBackup**。 如果要更新保存物件，請將 **dsmSendType** 參數設為 **stArchive**。

在階段作業狀態中，您只能啟動 **dsmUpdateObjEx** 呼叫；您無法從交易內部呼叫它，因為它會執行本身的交易。 您一次只能更新一個物件。

限制: 在 UNIX 或 Linux 作業系統上，如果變更擁有者欄位，除非您是 root 使用者，否則就無法查詢或還原物件。 只能更新現行作用中版本的備份物件。

語法

```
dsInt16_t  dsmUpdateObjEx
   (dsmUpdateObjExIn_t   *dsmUpdateObjExInP,
    dsmUpdateObjExOut_t  *dsmUpdateObjExOutP);
```

參數

dsmUpdateObjExIn_t *dsmUpdateObjExInP
此結構包含下列輸入參數：

dsUint16_t stVersion (I)
目前使用的結構版本。

dsUint32_t dsmHandle (I)
將此呼叫與先前的 **dsmInitEx** 呼叫結合的 handle。

dsmSendType sendType (I)
正在執行的傳送類型。 其值可能是：

stBackup
傳送到伺服器的備份物件。

stArchive
傳送到伺服器的保存物件。

dsmObjName *objNameP (I)
結構的指標，其中包含檔案空間名稱、高階物件名稱、 低階物件名稱和物件類型。 您不能使用萬用字元。

ObjAttr *objAttrPtr (I)
將物件屬性傳遞至應用程式。 更新的值視 **objUpdAct** 欄位中的旗標而定。 在此呼叫中，**objCompressed** 屬性會被忽略。

屬性如下：

- **owner** 會變更擁有者（如果輸入了新名稱）。
- **sizeEstimate** 是實際傳送的資料量（以位元組為單位）。 此值會儲存在 IBM Spectrum Protect meta 資料以供將來使用。
- **objCompressed** 是一個布林值，用來指出物件資料是否已經壓縮。
- **objInfo** 是一個屬性，包含要放置在 **objInfo** 欄位中的新資訊。 請將 **objInfoLength** 設定成新 **objInfo** 的長度。
- **mcNameP** 包含一個管理類別名稱，這個管理類別會置換取自 **dsmBindMC** 的管理類別。

dsUint32_t objUpdAct
指定 **objUpdAct** 的位元遮罩和動作包括：

DSM_BACKUPD_MC
更新物件的管理類別。

DSM_BACKUPD_OBJINFO
更新資訊物件 (**objInfo**)、 資訊物件的長度 (**objInfoLength**)，以及針對備份物件所傳送的資料量 (**sizeEstimate**)。

DSM_BACKUPD_OWNER
更新備份物件的擁有者。

DSM_ARCHUPD_DESCR
更新保存物件的**說明**欄位。 透過 **sendBuff** 參數來輸入新說明的值。

DSM_ARCHUPD_OBJINFO
更新資訊物件 (**objInfo**)、 資訊物件的長度 (**objInfoLength**)，以及針對保存物件所傳送的資料量 (**sizeEstimate**)。

DSM_ARCHUPD_OWNER
更新保存物件的擁有者。

ObjID archObjId
指定特定保存物件的唯一物件 ID。 由於多個保存物件可能會有相同的名稱，因此這個參數可識別特定的保存物件。 您可以使用查詢保存呼叫來取得物件 ID。

dsmUpdateObjExOut_t *dsmUpdateObjExOutP
此結構包含下列輸出參數：

dsUint16_t stVersion (I)
目前使用的結構版本。

回覆碼

在下表中，會用括弧 () 括住回覆碼的號碼。

表 *56. dsmUpdateObjEx* 的回覆碼

回覆碼	說明
DSM_RC_INVALID_ACTION (2012)	無效的動作。
DSM_RC_FS_NOT_REGISTERED (2061)	檔案空間未登錄。
DSM_RC_BAD_CALL_SEQUENCE (2041)	呼叫順序無效。
DSM_RC_WILDCHAR_NOTALLOWED (2050)	不容許使用萬用字元。
DSM_RC_ABORT_NO_MATCH (2)	前一個查詢不符。

附錄 A API 回覆碼原始檔：dsmrc.h

dsmrc.h 標頭檔包含 API 可傳回給應用程式的所有回覆碼。

在這裡提供的資訊包含使用 API 配送的 dsmrc.h 檔的時間點副本。檢視 API 配送套件中的檔案以取得最新版本。

```
/*****************************************************************************
*       IBM Spectrum Protect             *
* API Client Component                                                       *
*                                                                            *
* IBM Confidential                                                           *
* (IBM Confidential-Restricted when combined with the Aggregated OCO   *
* source modules for this program)                                           *
*                                                                            *
* OCO Source Materials                                                       *
*                                                                            *
* 5648-020 (C) Copyright IBM Corporation 1993, 2016                  *
*****************************************************************************/

/*****************************************************************************/
/* 標頭檔名稱：dsmrc.h                                                 */
/*                                                                           */
/* Descriptive-name:  Return codes from IBM Spectrum Protect APIs     */
/*****************************************************************************/
#ifndef _H_DSMRC
#define _H_DSMRC


#ifndef DSMAPILIB

#ifndef _H_ANSMACH
typedef int RetCode ;
#endif

#endif


#define DSM_RC_SUCCESSFUL                0 /* 順利完成    */
#define DSM_RC_OK                        0 /* 順利完成    */

#define DSM_RC_UNSUCCESSFUL             -1 /* 未順利完成 */

/* dsmEndTxn 原因碼 */
#define DSM_RS_ABORT_SYSTEM_ERROR           1
#define DSM_RS_ABORT_NO_MATCH               2
#define DSM_RS_ABORT_BY_CLIENT              3
#define DSM_RS_ABORT_ACTIVE_NOT_FOUND       4
#define DSM_RS_ABORT_NO_DATA                5
#define DSM_RS_ABORT_BAD_VERIFIER           6
#define DSM_RS_ABORT_NODE_IN_USE            7
#define DSM_RS_ABORT_EXPDATE_TOO_LOW        8
#define DSM_RS_ABORT_DATA_OFFLINE           9
#define DSM_RS_ABORT_EXCLUDED_BY_SIZE       10
#define DSM_RS_ABORT_NO_STO_SPACE_SKIP      11
#define DSM_RS_ABORT_NO_REPOSIT_SPACE     DSM_RS_ABORT_NO_STO_SPACE_SKIP
#define DSM_RS_ABORT_MOUNT_NOT_POSSIBLE     12
#define DSM_RS_ABORT_SIZESTIMATE_EXCEED     13
#define DSM_RS_ABORT_DATA_UNAVAILABLE       14
#define DSM_RS_ABORT_RETRY                  15
#define DSM_RS_ABORT_NO_LOG_SPACE           16
#define DSM_RS_ABORT_NO_DB_SPACE            17
#define DSM_RS_ABORT_NO_MEMORY              18

#define DSM_RS_ABORT_FS_NOT_DEFINED         20
#define DSM_RS_ABORT_NODE_ALREADY_DEFED     21
#define DSM_RS_ABORT_NO_DEFAULT_DOMAIN      22
#define DSM_RS_ABORT_INVALID_NODENAME       23
#define DSM_RS_ABORT_INVALID_POL_BIND       24
#define DSM_RS_ABORT_DEST_NOT_DEFINED       25
#define DSM_RS_ABORT_WAIT_FOR_SPACE         26
#define DSM_RS_ABORT_NOT_AUTHORIZED         27
#define DSM_RS_ABORT_RULE_ALREADY_DEFED     28
#define DSM_RS_ABORT_NO_STOR_SPACE_STOP     29
```

```
#define DSM_RS_ABORT_LICENSE_VIOLATION       30
#define DSM_RS_ABORT_EXTOBJID_ALREADY_EXISTS 31
#define DSM_RS_ABORT_DUPLICATE_OBJECT        32

#define DSM_RS_ABORT_INVALID_OFFSET          33    /* 部分物件擷取  */
#define DSM_RS_ABORT_INVALID_LENGTH          34    /* 部分物件擷取 */
#define DSM_RS_ABORT_STRING_ERROR            35
#define DSM_RS_ABORT_NODE_NOT_AUTHORIZED     36
#define DSM_RS_ABORT_RESTART_NOT_POSSIBLE    37
#define DSM_RS_ABORT_RESTORE_IN_PROGRESS     38
#define DSM_RS_ABORT_SYNTAX_ERROR            39

#define DSM_RS_ABORT_DATA_SKIPPED            40
#define DSM_RS_ABORT_EXCEED_MAX_MP           41
#define DSM_RS_ABORT_NO_OBJSET_MATCH         42
#define DSM_RS_ABORT_PVR_ERROR               43
#define DSM_RS_ABORT_BAD_RECOGTOKEN          44
#define DSM_RS_ABORT_MERGE_ERROR             45
#define DSM_RS_ABORT_FSRENAME_ERROR          46
#define DSM_RS_ABORT_INVALID_OPERATION       47
#define DSM_RS_ABORT_STGPOOL_UNDEFINED       48
#define DSM_RS_ABORT_INVALID_DATA_FORMAT     49
#define DSM_RS_ABORT_DATAMOVER_UNDEFINED     50

#define DSM_RS_ABORT_INVALID_MOVER_TYPE      231
#define DSM_RS_ABORT_ITEM_IN_USE             232
#define DSM_RS_ABORT_LOCK_CONFLICT           233
#define DSM_RS_ABORT_SRV_PLUGIN_COMM_ERROR   234
#define DSM_RS_ABORT_SRV_PLUGIN_OS_ERROR     235
#define DSM_RS_ABORT_CRC_FAILED              236
#define DSM_RS_ABORT_INVALID_GROUP_ACTION    237
#define DSM_RS_ABORT_DISK_UNDEFINED          238
#define DSM_RS_ABORT_BAD_DESTINATION         239
#define DSM_RS_ABORT_DATAMOVER_NOT_AVAILABLE 240
#define DSM_RS_ABORT_STGPOOL_COPY_CONT_NO    241
#define DSM_RS_ABORT_RETRY_SINGLE_TXN        242
#define DSM_RS_ABORT_TOC_CREATION_FAIL       243
#define DSM_RS_ABORT_TOC_LOAD_FAIL           244
#define DSM_RS_ABORT_PATH_RESTRICTED         245
#define DSM_RS_ABORT_NO_LANFREE_SCRATCH      246
#define DSM_RS_ABORT_INSERT_NOT_ALLOWED      247
#define DSM_RS_ABORT_DELETE_NOT_ALLOWED      248
#define DSM_RS_ABORT_TXN_LIMIT_EXCEEDED      249
#define DSM_RS_ABORT_OBJECT_ALREADY_HELD     250
#define DSM_RS_ABORT_INVALID_CHUNK_REFERENCE 254
#define DSM_RS_ABORT_DESTINATION_NOT_DEDUP   255
#define DSM_RS_ABORT_DESTINATION_POOL_CHANGED 257
#define DSM_RS_ABORT_NOT_ROOT                258


/* 回覆碼 */

#define DSM_RC_ABORT_SYSTEM_ERROR            DSM_RS_ABORT_SYSTEM_ERROR
#define DSM_RC_ABORT_NO_MATCH                DSM_RS_ABORT_NO_MATCH
#define DSM_RC_ABORT_BY_CLIENT               DSM_RS_ABORT_BY_CLIENT
#define DSM_RC_ABORT_ACTIVE_NOT_FOUND        DSM_RS_ABORT_ACTIVE_NOT_FOUND
#define DSM_RC_ABORT_NO_DATA                 DSM_RS_ABORT_NO_DATA
#define DSM_RC_ABORT_BAD_VERIFIER            DSM_RS_ABORT_BAD_VERIFIER
#define DSM_RC_ABORT_NODE_IN_USE             DSM_RS_ABORT_NODE_IN_USE
#define DSM_RC_ABORT_EXPDATE_TOO_LOW         DSM_RS_ABORT_EXPDATE_TOO_LOW
#define DSM_RC_ABORT_DATA_OFFLINE            DSM_RS_ABORT_DATA_OFFLINE
#define DSM_RC_ABORT_EXCLUDED_BY_SIZE        DSM_RS_ABORT_EXCLUDED_BY_SIZE

#define DSM_RC_ABORT_NO_REPOSIT_SPACE        DSM_RS_ABORT_NO_STO_SPACE_SKIP
#define DSM_RC_ABORT_NO_STO_SPACE_SKIP       DSM_RS_ABORT_NO_STO_SPACE_SKIP

#define DSM_RC_ABORT_MOUNT_NOT_POSSIBLE      DSM_RS_ABORT_MOUNT_NOT_POSSIBLE
#define DSM_RC_ABORT_SIZESTIMATE_EXCEED      DSM_RS_ABORT_SIZESTIMATE_EXCEED
#define DSM_RC_ABORT_DATA_UNAVAILABLE        DSM_RS_ABORT_DATA_UNAVAILABLE
#define DSM_RC_ABORT_RETRY                   DSM_RS_ABORT_RETRY
#define DSM_RC_ABORT_NO_LOG_SPACE            DSM_RS_ABORT_NO_LOG_SPACE
#define DSM_RC_ABORT_NO_DB_SPACE             DSM_RS_ABORT_NO_DB_SPACE
#define DSM_RC_ABORT_NO_MEMORY               DSM_RS_ABORT_NO_MEMORY

#define DSM_RC_ABORT_FS_NOT_DEFINED          DSM_RS_ABORT_FS_NOT_DEFINED
#define DSM_RC_ABORT_NODE_ALREADY_DEFED      DSM_RS_ABORT_NODE_ALREADY_DEFED
#define DSM_RC_ABORT_NO_DEFAULT_DOMAIN       DSM_RS_ABORT_NO_DEFAULT_DOMAIN
#define DSM_RC_ABORT_INVALID_NODENAME        DSM_RS_ABORT_INVALID_NODENAME
#define DSM_RC_ABORT_INVALID_POL_BIND        DSM_RS_ABORT_INVALID_POL_BIND
#define DSM_RC_ABORT_DEST_NOT_DEFINED        DSM_RS_ABORT_DEST_NOT_DEFINED
#define DSM_RC_ABORT_WAIT_FOR_SPACE          DSM_RS_ABORT_WAIT_FOR_SPACE
```

```
#define DSM_RC_ABORT_NOT_AUTHORIZED           DSM_RS_ABORT_NOT_AUTHORIZED
#define DSM_RC_ABORT_RULE_ALREADY_DEFED       DSM_RS_ABORT_RULE_ALREADY_DEFED
#define DSM_RC_ABORT_NO_STOR_SPACE_STOP       DSM_RS_ABORT_NO_STOR_SPACE_STOP

#define DSM_RC_ABORT_LICENSE_VIOLATION        DSM_RS_ABORT_LICENSE_VIOLATION
#define DSM_RC_ABORT_EXTOBJID_ALREADY_EXISTS DSM_RS_ABORT_EXTOBJID_ALREADY_EXISTS
#define DSM_RC_ABORT_DUPLICATE_OBJECT         DSM_RS_ABORT_DUPLICATE_OBJECT

#define DSM_RC_ABORT_INVALID_OFFSET           DSM_RS_ABORT_INVALID_OFFSET
#define DSM_RC_ABORT_INVALID_LENGTH           DSM_RS_ABORT_INVALID_LENGTH

#define DSM_RC_ABORT_STRING_ERROR             DSM_RS_ABORT_STRING_ERROR
#define DSM_RC_ABORT_NODE_NOT_AUTHORIZED      DSM_RS_ABORT_NODE_NOT_AUTHORIZED
#define DSM_RC_ABORT_RESTART_NOT_POSSIBLE     DSM_RS_ABORT_RESTART_NOT_POSSIBLE
#define DSM_RC_ABORT_RESTORE_IN_PROGRESS      DSM_RS_ABORT_RESTORE_IN_PROGRESS
#define DSM_RC_ABORT_SYNTAX_ERROR             DSM_RS_ABORT_SYNTAX_ERROR

#define DSM_RC_ABORT_DATA_SKIPPED             DSM_RS_ABORT_DATA_SKIPPED
#define DSM_RC_ABORT_EXCEED_MAX_MP            DSM_RS_ABORT_EXCEED_MAX_MP
#define DSM_RC_ABORT_NO_OBJSET_MATCH          DSM_RS_ABORT_NO_OBJSET_MATCH
#define DSM_RC_ABORT_PVR_ERROR                DSM_RS_ABORT_PVR_ERROR
#define DSM_RC_ABORT_BAD_RECOGTOKEN           DSM_RS_ABORT_BAD_RECOGTOKEN
#define DSM_RC_ABORT_MERGE_ERROR              DSM_RS_ABORT_MERGE_ERROR
#define DSM_RC_ABORT_FSRENAME_ERROR           DSM_RS_ABORT_FSRENAME_ERROR
#define DSM_RC_ABORT_INVALID_OPERATION        DSM_RS_ABORT_INVALID_OPERATION
#define DSM_RC_ABORT_STGPOOL_UNDEFINED        DSM_RS_ABORT_STGPOOL_UNDEFINED
#define DSM_RC_ABORT_INVALID_DATA_FORMAT      DSM_RS_ABORT_INVALID_DATA_FORMAT
#define DSM_RC_ABORT_DATAMOVER_UNDEFINED      DSM_RS_ABORT_DATAMOVER_UNDEFINED

#define DSM_RC_ABORT_INVALID_MOVER_TYPE       DSM_RS_ABORT_INVALID_MOVER_TYPE
#define DSM_RC_ABORT_ITEM_IN_USE              DSM_RS_ABORT_ITEM_IN_USE
#define DSM_RC_ABORT_LOCK_CONFLICT            DSM_RS_ABORT_LOCK_CONFLICT
#define DSM_RC_ABORT_SRV_PLUGIN_COMM_ERROR    DSM_RS_ABORT_SRV_PLUGIN_COMM_ERROR
#define DSM_RC_ABORT_SRV_PLUGIN_OS_ERROR      DSM_RS_ABORT_SRV_PLUGIN_OS_ERROR
#define DSM_RC_ABORT_CRC_FAILED               DSM_RS_ABORT_CRC_FAILED
#define DSM_RC_ABORT_INVALID_GROUP_ACTION     DSM_RS_ABORT_INVALID_GROUP_ACTION
#define DSM_RC_ABORT_DISK_UNDEFINED           DSM_RS_ABORT_DISK_UNDEFINED
#define DSM_RC_ABORT_BAD_DESTINATION          DSM_RS_ABORT_BAD_DESTINATION
#define DSM_RC_ABORT_DATAMOVER_NOT_AVAILABLE DSM_RS_ABORT_DATAMOVER_NOT_AVAILABLE
#define DSM_RC_ABORT_STGPOOL_COPY_CONT_NO     DSM_RS_ABORT_STGPOOL_COPY_CONT_NO
#define DSM_RC_ABORT_RETRY_SINGLE_TXN         DSM_RS_ABORT_RETRY_SINGLE_TXN
#define DSM_RC_ABORT_TOC_CREATION_FAIL        DSM_RS_ABORT_TOC_CREATION_FAIL
#define DSM_RC_ABORT_TOC_LOAD_FAIL            DSM_RS_ABORT_TOC_LOAD_FAIL
#define DSM_RC_ABORT_PATH_RESTRICTED          DSM_RS_ABORT_PATH_RESTRICTED
#define DSM_RC_ABORT_NO_LANFREE_SCRATCH       DSM_RS_ABORT_NO_LANFREE_SCRATCH
#define DSM_RC_ABORT_INSERT_NOT_ALLOWED       DSM_RS_ABORT_INSERT_NOT_ALLOWED
#define DSM_RC_ABORT_DELETE_NOT_ALLOWED       DSM_RS_ABORT_DELETE_NOT_ALLOWED
#define DSM_RC_ABORT_TXN_LIMIT_EXCEEDED       DSM_RS_ABORT_TXN_LIMIT_EXCEEDED
#define DSM_RC_ABORT_OBJECT_ALREADY_HELD      DSM_RS_ABORT_OBJECT_ALREADY_HELD
#define DSM_RC_ABORT_INVALID_CHUNK_REFERENCE DSM_RS_ABORT_INVALID_CHUNK_REFERENCE
#define DSM_RC_ABORT_DESTINATION_NOT_DEDUP    DSM_RS_ABORT_DESTINATION_NOT_DEDUP
#define DSM_RC_ABORT_DESTINATION_POOL_CHANGED DSM_RS_ABORT_DESTINATION_POOL_CHANGED
#define DSM_RC_ABORT_NOT_ROOT                 DSM_RS_ABORT_NOT_ROOT

#define DSM_RC_ABORT_CERTIFICATE_NOT_FOUND    DSM_RS_ABORT_CERTIFICATE_NOT_FOUND

/* 適用於伺服器登入拒絕程式碼的定義                                  */
/* 包含在範圍 (51 至 99) 的這些錯誤碼。                            */
#define DSM_RC_REJECT_NO_RESOURCES            51
#define DSM_RC_REJECT_VERIFIER_EXPIRED        52
#define DSM_RC_REJECT_ID_UNKNOWN              53
#define DSM_RC_REJECT_DUPLICATE_ID            54
#define DSM_RC_REJECT_SERVER_DISABLED         55
#define DSM_RC_REJECT_CLOSED_REGISTER         56
#define DSM_RC_REJECT_CLIENT_DOWNLEVEL        57
#define DSM_RC_REJECT_SERVER_DOWNLEVEL        58
#define DSM_RC_REJECT_ID_IN_USE               59
#define DSM_RC_REJECT_ID_LOCKED               61
#define DSM_RC_SIGNONREJECT_LICENSE_MAX       62
#define DSM_RC_REJECT_NO_MEMORY               63
#define DSM_RC_REJECT_NO_DB_SPACE             64
#define DSM_RC_REJECT_NO_LOG_SPACE            65
#define DSM_RC_REJECT_INTERNAL_ERROR          66
#define DSM_RC_SIGNONREJECT_INVALID_CLI       67 /* 未授權的用戶端類型 */
#define DSM_RC_CLIENT_NOT_ARCHRETPROT         68
#define DSM_RC_REJECT_LASTSESS_CANCELED       69
#define DSM_RC_REJECT_UNICODE_NOT_ALLOWED     70
#define DSM_RC_REJECT_NOT_AUTHORIZED          71
#define DSM_RC_REJECT_TOKEN_TIMEOUT           72
#define DSM_RC_REJECT_INVALID_NODE_TYPE       73
#define DSM_RC_REJECT_INVALID_SESSIONINIT     74
```

```
#define DSM_RC_REJECT_WRONG_PORT             75
#define DSM_RC_CLIENT_NOT_SPMRETPROT         79



#define DSM_RC_USER_ABORT          101 /* 使用者中斷處理         */
#define DSM_RC_NO_MEMORY           102 /* 無留存的 RAM 來完成要求   */
#define DSM_RC_TA_COMM_DOWN       2021 /* 不再使用                  */
#define DSM_RC_FILE_NOT_FOUND      104 /* 找不到指定的檔案          */
#define DSM_RC_PATH_NOT_FOUND      105 /* 指定的路徑不存在       */
#define DSM_RC_ACCESS_DENIED       106 /* 因為不正常的許可權而被拒絕 */
#define DSM_RC_NO_HANDLES          107 /* 無任何可用的檔案 handle   */
#define DSM_RC_FILE_EXISTS         108 /* 檔案已經存在              */
#define DSM_RC_INVALID_PARM        109 /* 傳送無效的參數。重大*/
#define DSM_RC_INVALID_HANDLE      110 /* 傳送無效的檔案 handle         */
#define DSM_RC_DISK_FULL           111 /* 不在磁碟空間                  */
#define DSM_RC_PROTOCOL_VIOLATION  113 /* 呼叫通訊協定違規。重大 */
#define DSM_RC_UNKNOWN_ERROR       114 /* 不明的系統錯誤。重大    */
#define DSM_RC_UNEXPECTED_ERROR    115 /* 非預期的錯誤。重大        */
#define DSM_RC_FILE_BEING_EXECUTED 116 /* 不接受寫入                */
#define DSM_RC_DIR_NO_SPACE        117 /* 無法展開目錄       */
#define DSM_RC_LOOPED_SYM_LINK     118 /* 在翻譯路徑中發現太多
                                          encountered in translating path.  */
#define DSM_RC_FILE_NAME_TOO_LONG  119 /* 檔案名稱太長                 */
#define DSM_RC_FILE_SPACE_LOCKED   120 /* 系統已鎖定檔案空間 */
#define DSM_RC_FINISHED            121 /* 完成處理                */
#define DSM_RC_UNKNOWN_FORMAT      122 /* 不明的格式                    */
#define DSM_RC_NO_AUTHORIZATION    123 /* 當用戶端沒有授權來讀取
                                          另一個主機的擁有者備份
                                          保存資料時，伺服器會回應  */
#define DSM_RC_FILE_SPACE_NOT_FOUND 124/* 找不到指定的檔案空間    */
#define DSM_RC_TXN_ABORTED         125 /* 異動中斷                */
#define DSM_RC_SUBDIR_AS_FILE      126 /* 子目錄名稱以檔案存在  */
#define DSM_RC_PROCESS_NO_SPACE    127 /* 程序沒有任何磁碟空間。   */
#define DSM_RC_PATH_TOO_LONG       128 /* 目錄路徑已建置
                                          變成太長                      */
#define DSM_RC_NOT_COMPRESSED      129 /* 經由壓縮的檔案
                                          實際上並不是                  */
#define DSM_RC_TOO_MANY_BITS       130 /* 使用比
                                          expander 能處理更多位元      */
#define DSM_RC_SYSTEM_ERROR        131 /* 資源以外內部系統錯誤          */
#define DSM_RC_NO_SERVER_RESOURCES 132 /* 伺服器的檔案。           */
#define DSM_RC_FS_NOT_KNOWN        133 /* 這是伺服器未知
                                          的檔案空間                        */
#define DSM_RC_NO_LEADING_DIRSEP   134 /* 無目錄分隔符號    */
#define DSM_RC_WILDCARD_DIR        135 /* 當不接受時，萬用字元
                                          正在目錄路徑中              */
#define DSM_RC_COMM_PROTOCOL_ERROR 136 /* 通訊協定錯誤     */
#define DSM_RC_AUTH_FAILURE        137 /* 鑑別失敗           */
#define DSM_RC_TA_NOT_VALID        138 /* TA 不是根目錄和/或 SUID 程式 */
#define DSM_RC_KILLED              139 /* 程序已刪除。                  */

#define DSM_RC_RETRY               143 /* 重試相同的作業        */

#define DSM_RC_WOULD_BLOCK         145 /* 作業可導致系統來
                                          block waiting for input.          */
#define DSM_RC_TOO_SMALL           146 /* 適用於編譯型樣的區域很小   */
#define DSM_RC_UNCLOSED              147 /* 在型樣中沒有關閉方括弧   */
#define DSM_RC_NO_STARTING_DELIMITER 148 /* 型樣必須以目錄定界符號
                                            來啟動              */
#define DSM_RC_NEEDED_DIR_DELIMITER 149 /* 立即需要目錄定界符號
                                           在
                                           「符合目錄」metastring
                                           ("...") ，但是找不到      */
#define DSM_RC_UNKNOWN_FILE_DATA_TYPE 150 /* 結構檔案資料類型
                                             不明                       */
#define DSM_RC_BUFFER_OVERFLOW       151 /* 資料緩衝區溢位            */


#define DSM_RC_NO_COMPRESS_MEMORY  154 /* 壓縮/延伸記憶體不足     */
#define DSM_RC_COMPRESS_GREW       155 /* 壓縮成長                  */
#define DSM_RC_INV_COMM_METHOD     156 /* 指定的無效 comm 方法     */
#define DSM_RC_WILL_ABORT          157 /* 將中斷異動        */
#define DSM_RC_FS_WRITE_LOCKED     158 /* 檔案空間寫入已鎖定       */
#define DSM_RC_SKIPPED_BY_USER     159 /* 已略過使用者需要的檔案
```

```
                                        以防 ABORT_DATA_OFFLINE        */
#define DSM_RC_TA_NOT_FOUND        160 /* 在其目錄下找不到 TA    */
#define DSM_RC_TA_ACCESS_DENIED    161 /* 拒絕到 TA 的存取權             */
#define DSM_RC_FS_NOT_READY        162 /* 檔案空間尚未備妥               */
#define DSM_RC_FS_IS_BAD           163 /* 檔案空間不正確的                  */
#define DSM_RC_FIO_ERROR           164 /* 檔案輸入/輸出錯誤            */
#define DSM_RC_WRITE_FAILURE       165 /* 寫入檔案發生錯誤              */
#define DSM_RC_OVER_FILE_SIZE_LIMIT 166 /* 系統上的檔案/使用者限制      */
#define DSM_RC_CANNOT_MAKE         167 /* 無法建立檔案/目錄，
                                        可為不正確的名稱                */
#define DSM_RC_NO_PASS_FILE        168 /* 需要密碼檔案及使用者
                                        不是根目錄                          */
#define DSM_RC_VERFILE_OLD         169 /* 儲存在本端環境的密碼不
                                        符合在主機上的那一個          */
#define DSM_RC_INPUT_ERROR         173 /* 無法讀取鍵盤輸入      */
#define DSM_RC_REJECT_PLATFORM_MISMATCH 174 /* 平台名稱不符
                                           適用於用戶端伺服器所
                                           說的平台 */
#define DSM_RC_TL_NOT_FILE_OWNER   175 /* 使用者嘗試備份不是
                                        the file's owner.                */
#define DSM_RC_COMPRESSED_DATA_CORRUPTED 176 /* 壓縮資料毀損     */
#define DSM_RC_UNMATCHED_QUOTE     177 /* 缺少啟動或引用尾端  */

#define DSM_RC_SIGNON_FAILOVER_MODE 178      /* Failed over to the replication server,
                                              running in failover mode */
#define DSM_RC_FAILOVER_MODE_FUNC_BLOCKED 179 /* function is blocked because
                                                 session is in failover mode */

/*-----------------------------------------------------------------------------*/
/* 回覆碼 180-199 已為「原則集」處理保留                 */
/*-----------------------------------------------------------------------------*/
#define DSM_RC_PS_MULTBCG          181 /* 多重備份副本群組在 1 MC*/
#define DSM_RC_PS_MULTACG          182 /* Multiple arch.  copy groups in 1 MC*/
#define DSM_RC_PS_NODFLTMC         183 /* 預設 MC 名稱不在原則集 */
#define DSM_RC_TL_NOBCG            184 /* 備份 req，無備份副本群組   */
#define DSM_RC_TL_EXCLUDED         185 /* 備份 req，由 in/ex 過濾器排除  */
#define DSM_RC_TL_NOACG            186 /* 保存 req，無保存副本群組 */
#define DSM_RC_PS_INVALID_ARCHMC   187 /* 無效 MC 名稱在保存置換*/
#define DSM_RC_NO_PS_DATA          188 /* 在伺服器上無原則集資料  */
#define DSM_RC_PS_INVALID_DIRMC    189 /* 指定的無效目錄 MC 在
                                        the options file.                */
#define DSM_RC_PS_NO_CG_IN_DIR_MC  190 /* 無備份副本群組在目錄 MC 上。
                                        必須使用 DieMC 選項來指定
                                        選項的設定。                          */

#define DSM_RC_WIN32_UNSUPPORTED_FILE_TYPE 280 /* 檔案不是
                                                  Win32 類型 FILE_TYPE_DISK  */

/*-----------------------------------------------------------------------------*/
/* 適用於「授信通訊代理程式」的回覆碼                          */
/*-----------------------------------------------------------------------------*/
#define DSM_RC_TCA_NOT_ROOT        161 /* 已拒絕到 TA 的存取權              */
#define DSM_RC_TCA_ATTACH_SHR_MEM_ERR  200 /* 連接共用記憶體的錯誤  */
#define DSM_RC_TCA_SHR_MEM_BLOCK_ERR   200 /* 共用記憶體鎖定錯誤      */
#define DSM_RC_TCA_SHR_MEM_IN_USE      200 /* 共用記憶體鎖定錯誤      */
#define DSM_RC_TCA_SHARED_MEMORY_ERROR 200 /* 共用記憶體鎖定錯誤       */
#define DSM_RC_TCA_SEGMENT_MISMATCH    200 /* 共用記憶體鎖定錯誤      */
#define DSM_RC_TCA_FORK_FAILED     292 /* 分岔關閉 TCA 處理發生錯誤      */
#define DSM_RC_TCA_DIED            294 /* 無預期 TCA 失效              */
#define DSM_RC_TCA_INVALID_REQUEST 295 /* 無效要求傳送至 TCA        */
#define DSM_RC_TCA_SEMGET_ERROR    297 /* 取得號誌時發生錯誤            */
#define DSM_RC_TCA_SEM_OP_ERROR    298 /* 設定或等待號誌時發生錯誤     */
#define DSM_RC_TCA_NOT_ALLOWED     299 /* 不接受 TCA ( 多重執行緒 )     */

/*-----------------------------------------------------------------------------*/
/* 400-430  適用於選項                                                  */
/*-----------------------------------------------------------------------------*/
#define DSM_RC_INVALID_OPT          400 /* 無效選項                  */
#define DSM_RC_NO_HOST_ADDR         405 /* 無 enuf 資訊來連接伺服器  */
#define DSM_RC_NO_OPT_FILE          406 /*無預設使用者配置檔*/
#define DSM_RC_MACHINE_SAME         408 /* -MACHINENAME 和實際的名稱相同   */
#define DSM_RC_INVALID_SERVER       409 /* 從用戶端的無效伺服器  */
#define DSM_RC_INVALID_KEYWORD      410 /* 無效選項關鍵字            */
#define DSM_RC_PATTERN_TOO_COMPLEX  411 /* 不符合併入/排除項目*/
#define DSM_RC_NO_CLOSING_BRACKET   412 /* 缺少關閉方括弧 inc/excl */
#define DSM_RC_OPT_CLIENT_NOT_ACCEPTING 417/* 用戶端不接受來自伺服器的這個
```

```
                                          選項                */
#define DSM_RC_OPT_CLIENT_DOES_NOT_WANT 418/* 用戶端不需要這個值
                                          選項                */
#define DSM_RC_OPT_NO_INCLEXCL_FILE     419/* 找不到併入/排除檔              */
#define DSM_RC_OPT_OPEN_FAILURE     420   /* 無法開啟檔案                    */
#define DSM_RC_OPT_INV_NODENAME     421/* 使用適用於 Windows 若節點名稱=本端
                                           機器當 CLUSTERNODE=是       */
#define DSM_RC_OPT_NODENAME_INVALID 423/* 同屬無效節點名稱           */
#define DSM_RC_OPT_ERRORLOG_CONFLICT 424/* 同時指定 logmax & 保留    */
#define DSM_RC_OPT_SCHEDLOG_CONFLICT 425/* 同時指定 logmax & 保留  */
#define DSM_RC_CANNOT_OPEN_TRACEFILE 426/* 無法開啟追蹤檔                      */
#define DSM_RC_CANNOT_OPEN_LOGFILE   427/* 無法開啟錯誤日誌檔         */
#define DSM_RC_OPT_SESSINIT_LF_CONFLICT 428/* 同時指定 sessioninit=server 和
                                          enablelanfree=yes*/
#define DSM_RC_OPT_OPTION_IGNORE     429/* 選項將被忽略             */
#define DSM_RC_OPT_DEDUP_CONFLICT    430/* 無法開啟錯誤日誌檔        */
#define DSM_RC_OPT_HSMLOG_CONFLICT 431/* both logmax & retention specified  */


/*----------------------------------------------------------------------------*/
/*  600 至 610 適用於標籤程式碼                                               */
/*----------------------------------------------------------------------------*/
#define DSM_RC_DUP_LABEL           600 /* 發現重複的磁區標籤                  */
#define DSM_RC_NO_LABEL            601 /* 磁碟沒有標籤                        */

/*----------------------------------------------------------------------------*/
/* 訊息檔處理的回覆碼                                                         */
/*----------------------------------------------------------------------------*/
#define DSM_RC_NLS_CANT_OPEN_TXT   610 /* 嘗試開啟 msg txt 檔案時發生錯誤 */
#define DSM_RC_NLS_CANT_READ_HDR   611 /* 嘗試讀取標頭發生錯誤        */
#define DSM_RC_NLS_INVALID_CNTL_REC 612 /* 無效控制記錄             */
#define DSM_RC_NLS_INVALID_DATE_FMT 613 /* 無效的預設日期格式       */
#define DSM_RC_NLS_INVALID_TIME_FMT 614 /* 無效的預設時間格式       */
#define DSM_RC_NLS_INVALID_NUM_FMT 615 /* 無效預設號碼格式      */

/*----------------------------------------------------------------------------*/
/* 已保留適用於訊息回覆碼的回覆碼 620-630            */
/*----------------------------------------------------------------------------*/
#define DSM_RC_LOG_CANT_BE_OPENED  620 /* 嘗試開啟錯誤日誌時發生錯誤     */
#define DSM_RC_LOG_ERROR_WRITING_TO_LOG 621 /* 寫入日誌檔時
                                           發生錯誤                        */
#define DSM_RC_LOG_NOT_SPECIFIED   622 /* 無指定的錯誤日誌檔    */


/*----------------------------------------------------------------------------*/
/* Return codes 900-999 IBM SPECTRUM PROTECT CLIENT ONLY                      */
/*----------------------------------------------------------------------------*/
#define DSM_RC_NOT_ADSM_AUTHORIZED  927 /* 必須授權 ADSM 來執行*/
                                        /* 動作：root 使用者或密碼 auth    */
#define DSM_RC_REJECT_USERID_UNKNOWN 940 /* 伺服器上不明的使用者 ID         */
#define DSM_RC_FILE_IS_SYMLINK      959 /* 錯誤日誌或追蹤是符號式
                                           鏈結
                                                                        */

#define DSM_RC_DIRECT_STORAGE_AGENT_UNSUPPORTED 961 /* 不支援直接連接至 SA  */
#define DSM_RC_FS_NAMESPACE_DOWNLEVEL 963 /* Long 名稱空間已移除從
                                             從 NetWare 磁區 */
#define DSM_RC_CONTINUE_NEW_CONSUMER   972 /* 使用新消費者繼續處理 */
#define DSM_RC_CONTINUE_NEW_CONSUMER_NODEDUP   973 /* 使用新消費者繼續處理，不刪除重複資料*/
#define DSM_RC_CONTINUE_NEW_CONSUMER_NOCOMPRESS 976 /* Continue processing using a new consumer no
compression */


#define DSM_RC_SERVER_SUPPORTS_FUNC    994 /* 伺服器支援這個功能  */
#define DSM_RC_SERVER_AND_SA_SUPPORT_FUNC 995 /* 伺服器及 SA 兩者皆支援 func */
#define DSM_RC_SERVER_DOWNLEVEL_FUNC   996 /* 伺服器是適用於 func 的下層    */
#define DSM_RC_STORAGEAGENT_DOWNLEVEL  997 /* 儲存體代理站是下層       */
#define DSM_RC_SERVER_AND_SA_DOWNLEVEL 998 /* 伺服器和 SA 兩者皆為下層       */


/* TCP/IP 錯誤碼 */
#define DSM_RC_TCPIP_FAILURE       -50 /* TCP/IP 通訊失敗      */
#define DSM_RC_CONN_TIMEDOUT       -51 /* TCP/IP 連接嘗試逾時 */
#define DSM_RC_CONN_REFUSED        -52 /* 主機拒絕 TCP/IP 連線  */
#define DSM_RC_BAD_HOST_NAME       -53 /* 指定的 TCP/IP 無效主機名稱 */
#define DSM_RC_NETWORK_UNREACHABLE -54 /* TCP/IP 主機名稱未達成       */
#define DSM_RC_WINSOCK_MISSING     -55 /* TCP/IP WINSOCK.DLL 遺失          */
```

```
#define DSM_RC_TCPIP_DLL_LOADFAILURE -56 /* 從 LoadLibrary 發生錯誤          */
#define DSM_RC_TCPIP_LOADFAILURE   -57 /* 從 GetProcAddress 發生錯誤         */
#define DSM_RC_TCPIP_USER_ABORT    -58 /* 使用者中斷當在 TCP/IP 層 */

/*-------------------------------------------------------------------------*/
/* 已保留適用於 CommTSM 錯誤碼的回覆碼 (-71)-(-90)        */
/*-------------------------------------------------------------------------*/
#define DSM_RC_TSM_FAILURE         -71 /* IBM Spectrum Protect comm failure  */
#define DSM_RC_TSM_ABORT           -72 /* 不尋常中斷階段作業         */

/*comm3270 錯誤碼 - 不再使用*/
#define DSM_RC_COMM_TIMEOUT       2021   /* 不再使用                          */
#define DSM_RC_EMULATOR_INACTIVE  2021   /* 不再使用                          */
#define DSM_RC_BAD_HOST_ID        2021   /* 不再使用                          */
#define DSM_RC_HOST_SESS_BUSY     2021   /* 不再使用                          */
#define DSM_RC_3270_CONNECT_FAILURE 2021 /* 不再使用                          */
#define DSM_RC_NO_ACS3ELKE_DLL    2021   /* 不再使用                          */
#define DSM_RC_EMULATOR_ERROR     2021   /* 不再使用                          */
#define DSM_RC_EMULATOR_BACKLEVEL 2021   /* 不再使用                          */
#define DSM_RC_CKSUM_FAILURE      2021   /* 不再使用                          */


/* The following Return codes are for EHLLAPI for Windows                   */
#define DSM_RC_3270COMMError_DLL        2021   /* no longer used             */
#define DSM_RC_3270COMMError_GetProc    2021   /* no longer used             */
#define DSM_RC_EHLLAPIError_DLL         2021   /* no longer used             */
#define DSM_RC_EHLLAPIError_GetProc     2021   /* no longer used             */
#define DSM_RC_EHLLAPIError_HostConnect 2021   /* no longer used             */
#define DSM_RC_EHLLAPIError_AllocBuff   2021   /* no longer used             */
#define DSM_RC_EHLLAPIError_SendKey     2021   /* no longer used             */
#define DSM_RC_EHLLAPIError_PacketChk   2021   /* no longer used             */
#define DSM_RC_EHLLAPIError_ChkSum      2021   /* no longer used             */
#define DSM_RC_EHLLAPIError_HostTimeOut 2021   /* no longer used             */
#define DSM_RC_EHLLAPIError_Send        2021   /* no longer used             */
#define DSM_RC_EHLLAPIError_Recv        2021   /* no longer used             */
#define DSM_RC_EHLLAPIError_General     2021   /* no longer used             */
#define DSM_RC_PC3270_MISSING_DLL       2021   /* no longer used             */
#define DSM_RC_3270COMM_MISSING_DLL     2021   /* no longer used             */


/* NETBIOS error codes */
#define DSM_RC_NETB_ERROR         -151 /* Could not add node to LAN          */
#define DSM_RC_NETB_NO_DLL        -152 /* The ACSNETB.DLL could not be loaded*/
#define DSM_RC_NETB_LAN_ERR       -155 /* LAN error detected                 */
#define DSM_RC_NETB_NAME_ERR      -158 /* Netbios error on Add Name          */
#define DSM_RC_NETB_TIMEOUT       -159 /* Netbios send timeout               */
#define DSM_RC_NETB_NOTINST       -160 /* Netbios not installed - DOS        */
#define DSM_RC_NETB_REBOOT        -161 /* Netbios config err - reboot DOS    */

/* Named Pipe error codes */
#define DSM_RC_NP_ERROR                      -190

/* CPIC error codes */
#define DSM_RC_CPIC_ALLOCATE_FAILURE          2021 /* 不再使用            */
#define DSM_RC_CPIC_TYPE_MISMATCH             2021 /* 不再使用            */
#define DSM_RC_CPIC_PIP_NOT_SPECIFY_ERR       2021 /* 不再使用            */
#define DSM_RC_CPIC_SECURITY_NOT_VALID        2021 /* 不再使用            */
#define DSM_RC_CPIC_SYNC_LVL_NO_SUPPORT       2021 /* 不再使用            */
#define DSM_RC_CPIC_TPN_NOT_RECOGNIZED        2021 /* 不再使用            */
#define DSM_RC_CPIC_TP_ERROR                  2021 /* 不再使用            */
#define DSM_RC_CPIC_PARAMETER_ERROR           2021 /* 不再使用            */
#define DSM_RC_CPIC_PROD_SPECIFIC_ERR         2021 /* 不再使用            */
#define DSM_RC_CPIC_PROGRAM_ERROR             2021 /* 不再使用            */
#define DSM_RC_CPIC_RESOURCE_ERROR            2021 /* 不再使用            */
#define DSM_RC_CPIC_DEALLOCATE_ERROR          2021 /* 不再使用            */
#define DSM_RC_CPIC_SVC_ERROR                 2021 /* 不再使用            */
#define DSM_RC_CPIC_PROGRAM_STATE_CHECK       2021 /* 不再使用            */
#define DSM_RC_CPIC_PROGRAM_PARAM_CHECK       2021 /* 不再使用            */
#define DSM_RC_CPIC_UNSUCCESSFUL              2021 /* 不再使用            */
#define DSM_RC_UNKNOWN_CPIC_PROBLEM           2021 /* 不再使用            */
#define DSM_RC_CPIC_MISSING_LU                2021 /* 不再使用            */
#define DSM_RC_CPIC_MISSING_TP                2021 /* 不再使用            */
#define DSM_RC_CPIC_SNA6000_LOAD_FAIL         2021 /* 不再使用            */
#define DSM_RC_CPIC_STARTUP_FAILURE           2021 /* 不再使用            */

/*-------------------------------------------------------------------------*/
/* Return codes -300 to -307 are reserved for IPX/SPX communications       */
/*-------------------------------------------------------------------------*/
```

```
#define DSM_RC_TLI_ERROR                        2021 /* 不再使用            */
#define DSM_RC_IPXSPX_FAILURE                   2021 /* 不再使用            */
#define DSM_RC_TLI_DLL_MISSING                  2021 /* 不再使用            */
#define DSM_RC_DLL_LOADFAILURE                  2021 /* 不再使用            */
#define DSM_RC_DLL_FUNCTION_LOADFAILURE         2021 /* 不再使用            */
#define DSM_RC_IPXCONN_REFUSED                  2021 /* 不再使用            */
#define DSM_RC_IPXCONN_TIMEDOUT                 2021 /* 不再使用            */
#define DSM_RC_IPXADDR_UNREACHABLE              2021 /* 不再使用            */
#define DSM_RC_CPIC_MISSING_DLL                 2021 /* 不再使用            */
#define DSM_RC_CPIC_DLL_LOADFAILURE             2021 /* 不再使用            */
#define DSM_RC_CPIC_FUNC_LOADFAILURE            2021 /* 不再使用            */

/*=== 共用記憶體通訊協定錯誤碼   ===*/
#define DSM_RC_SHM_TCPIP_FAILURE                -450
#define DSM_RC_SHM_FAILURE                      -451
#define DSM_RC_SHM_NOTAUTH                      -452

#define DSM_RC_NULL_OBJNAME         2000 /* 物件名稱指標是空值      */
#define DSM_RC_NULL_DATABLKPTR      2001 /* dataBlkPtr 是空值               */
#define DSM_RC_NULL_MSG             2002 /* 在 dsmRCMsg 的 msg parm 是空值     */

#define DSM_RC_NULL_OBJATTRPTR      2004 /* 物件 Attr 指標是空值      */

#define DSM_RC_NO_SESS_BLK          2006 /* 無伺服器階段作業資訊            */
#define DSM_RC_NO_POLICY_BLK        2007 /* 無原則 hdr 資訊           */
#define DSM_RC_ZERO_BUFLEN          2008 /* 適用於 dataBlkPtr 的 bufferLen 是零 */
#define DSM_RC_NULL_BUFPTR          2009 /* 適用於 dataBlkPtr 的 bufferPtr 是空值 */

#define DSM_RC_INVALID_OBJTYPE      2010 /* 無效的物件類型              */
#define DSM_RC_INVALID_VOTE         2011 /* 無效的 vote                     */
#define DSM_RC_INVALID_ACTION       2012 /* 無效的動作                    */
#define DSM_RC_INVALID_DS_HANDLE    2014 /* 無效的 ADSM 處理              */
#define DSM_RC_INVALID_REPOS        2015 /* 儲存庫的無效值     */
#define DSM_RC_INVALID_FSNAME       2016 /* fs 應該開始於 dir delim   */
#define DSM_RC_INVALID_OBJNAME      2017 /* 無效的完整路徑名稱            */
#define DSM_RC_INVALID_LLNAME       2018 /* ll 應該開始於 dir delim   */
#define DSM_RC_INVALID_OBJOWNER     2019 /* 無效的物件擁有者名稱          */
#define DSM_RC_INVALID_ACTYPE       2020 /* 無效的動作類型                */
#define DSM_RC_INVALID_RETCODE      2021 /* 在 dsmRCMsg 的 dsmRC 無效     */
#define DSM_RC_INVALID_SENDTYPE     2022 /* 無效的傳送類型              */
#define DSM_RC_INVALID_PARAMETER    2023 /* 無效參數                   */
#define DSM_RC_INVALID_OBJSTATE     2024 /* 作用中、非作用中或有其他符合？   */
#define DSM_RC_INVALID_MCNAME       2025 /* 找不到 Mgmt 類別名稱         */
#define DSM_RC_INVALID_DRIVE_CHAR   2026 /* 磁碟機代號不是英文字母     */
#define DSM_RC_NULL_FSNAME          2027 /* 檔案空間名稱是空值             */
#define DSM_RC_INVALID_HLNAME       2028 /* hl 應該開始於 dir delim   */

#define DSM_RC_NUMOBJ_EXCEED        2029 /* BeginGetData num 物件已超出   */

#define DSM_RC_NEWPW_REQD           2030 /* 新密碼是必要的          */
#define DSM_RC_OLDPW_REQD           2031 /* 舊密碼是必要的          */
#define DSM_RC_NO_OWNER_REQD        2032 /* 不接受擁有者。接受預設值 */
#define DSM_RC_NO_NODE_REQD         2033 /* 不接受節點 w/ pw=產生  */
#define DSM_RC_KEY_MISSING          2034 /* 找不到金鑰檔           */
#define DSM_RC_KEY_BAD              2035 /* 金鑰檔內容不正確        */

#define DSM_RC_BAD_CALL_SEQUENCE    2041 /* 不接受呼叫 DSM 順序 */
#define DSM_RC_INVALID_TSMBUFFER    2042 /* 適用於 tsmbuffhandle 或 dataPtr 的無效值*/
#define DSM_RC_TOO_MANY_BYTES       2043 /* 複製過多位元組至緩衝區    */
#define DSM_RC_MUST_RELEASE_BUFFER 2044 /* 無法結束 app 需要來釋放緩衝區 */
#define DSM_RC_BUFF_ARRAY_ERROR     2045 /* 內部緩衝區陣列發生錯誤          */
#define DSM_RC_INVALID_DATABLK      2046 /* 使用 tsmbuff datablk 應該是空值 */
#define DSM_RC_ENCR_NOT_ALLOWED     2047 /* 不接受使用 tsmbuffers 加密 */
#define DSM_RC_OBJ_COMPRESSED       2048 /* 無法使用在壓縮物件上的 tsmBuff 來還原  */
#define DSM_RC_OBJ_ENCRYPTED        2049 /* 無法使用在加密物件上的 tsmbuff 來還原  */
#define DSM_RC_WILDCHAR_NOTALLOWED 2050 /* Wild 卡片不適用於 hl,ll  */
#define DSM_RC_POR_NOT_ALLOWED      2051 /* 無法以 tsmBuffers 來使用部分物件還原 */
#define DSM_RC_NO_ENCRYPTION_KEY    2052 /* 找不到加密鍵 */
#define DSM_RC_ENCR_CONFLICT        2053 /* 互斥選項 */

#define DSM_RC_FSNAME_NOTFOUND      2060 /* 找不到檔案空間名稱          */
#define DSM_RC_FS_NOT_REGISTERED    2061 /* 無法登錄檔案空間名稱    */
#define DSM_RC_FS_ALREADY_REGED     2062 /* 檔案空間已登錄     */
#define DSM_RC_OBJID_NOTFOUND       2063 /* 無還原物件 ID           */
#define DSM_RC_WRONG_VERSION        2064 /* 錯誤層級程式碼              */
#define DSM_RC_WRONG_VERSION_PARM   2065 /* 錯誤參數結構層級  */
```

```
#define DSM_RC_NEEDTO_ENDTXN       2070 /* 需要呼叫 dsmEndTxn           */

#define DSM_RC_OBJ_EXCLUDED        2080 /* MC 已排除物件         */
#define DSM_RC_OBJ_NOBCG           2081 /* 物件無備份副本群組  */
#define DSM_RC_OBJ_NOACG           2082 /* 物件無保存副本群組 */

#define DSM_RC_APISYSTEM_ERROR     2090 /* API 內部錯誤               */

#define DSM_RC_DESC_TOOLONG        2100 /* 說明過長           */
#define DSM_RC_OBJINFO_TOOLONG     2101 /* 物件屬性 objinfo 過長     */
#define DSM_RC_HL_TOOLONG          2102 /* 高階限定元過長 */
#define DSM_RC_PASSWD_TOOLONG      2103 /* 密碼過長               */
#define DSM_RC_FILESPACE_TOOLONG   2104 /* 檔案空間名稱過長        */
#define DSM_RC_LL_TOOLONG          2105 /* 低階限定元過長  */
#define DSM_RC_FSINFO_TOOLONG      2106 /* 檔案空間長度過長      */
#define DSM_RC_SENDDATA_WITH_ZERO_SIZE 2107 /* 傳送資料 w/ 零 est        */

/*=== 新的回覆碼適用於 dsmaccess ===*/
#define DSM_RC_INVALID_ACCESS_TYPE 2110 /* 無效存取權類型               */
#define DSM_RC_QUERY_COMM_FAILURE  2111 /* 查詢期間通訊發生錯誤 */
#define DSM_RC_NO_FILES_BACKUP     2112 /* 無備份檔案適用於這個 fs   */
#define DSM_RC_NO_FILES_ARCHIVE    2113 /* 無保存檔案適用於這個 fs   */
#define DSM_RC_INVALID_SETACCESS   2114 /* 無效設定存取權格式        */

/*=== 新的回覆碼適用於 dsmaccess ===*/
#define DSM_RC_STRING_TOO_LONG     2120 /* 字串參數過長        */

#define DSM_RC_MORE_DATA           2200 /* 尚有資料要還原   */

#define DSM_RC_BUFF_TOO_SMALL      2210 /* 適用於 qry 的 DataBlk 緩衝區太小 */

#define DSM_RC_NO_API_CONFIGFILE   2228 /*找不到指定的 API 配置檔 */
#define DSM_RC_NO_INCLEXCL_FILE    2229 /* 找不到指定的 inclexcl 檔 */
#define DSM_RC_NO_SYS_OR_INCLEXCL  2230 /* 找不到在 dsm.sys 上指定的 either dsm.sys 或 inclexcl
                                           檔   */
#define DSM_RC_REJECT_NO_POR_SUPPORT 2231 /* 伺服器無 POR 支援*/

#define DSM_RC_NEED_ROOT           2300 /* API 呼叫者必須是根目錄           */
#define DSM_RC_NEEDTO_CALL_BINDMC  2301 /* dsmBindMC 必須最先呼叫   */
#define DSM_RC_CHECK_REASON_CODE   2302 /* 檢查從 dsmEndTxn 的原因碼 */
#define DSM_RC_NEEDTO_ENDTXN_DEDUP_SIZE_EXCEEDED 2303 /* 已超出 dedup 位元組數上限 */

/*=== 請參閱 agentrc.h，由 lic 檔案使用的回覆碼 2400 - 2410 ===*/

/*=== 請參閱 agentrc.h，由 Oracle 代理程式使用的回覆碼 2410 - 2430 ===*/

#define DSM_RC_ENC_WRONG_KEY       4580 /* 提供的金鑰不正確    */
#define DSM_RC_ENC_NOT_AUTHORIZED  4582 /* 不接受使用者解密    */
#define DSM_RC_ENC_TYPE_UNKNOWN    4584 /* 不明的加密類型         */

/*============================================================================
   已保留適用於叢集的回覆碼 (4600)-(4624)
============================================================================*/
#define DSM_RC_CLUSTER_INFO_LIBRARY_NOT_LOADED        4600
#define DSM_RC_CLUSTER_LIBRARY_INVALID                4601
#define DSM_RC_CLUSTER_LIBRARY_NOT_LOADED             4602
#define DSM_RC_CLUSTER_NOT_MEMBER_OF_CLUSTER          4603
#define DSM_RC_CLUSTER_NOT_ENABLED                    4604
#define DSM_RC_CLUSTER_NOT_SUPPORTED                  4605
#define DSM_RC_CLUSTER_UNKNOWN_ERROR                  4606

/*============================================================================
   Return codes (5200)-(5600) are reserved for new Server ABORT codes (dsmcomm.h)
============================================================================*/
#define DSM_RS_ABORT_CERTIFICATE_NOT_FOUND   5200

/*============================================================================
   已保留適用於 proxy 的回覆碼 (5701)-(5749)
============================================================================*/
#define DSM_RC_PROXY_REJECT_NO_RESOURCES              5702
#define DSM_RC_PROXY_REJECT_DUPLICATE_ID              5705
#define DSM_RC_PROXY_REJECT_ID_IN_USE                 5710
#define DSM_RC_PROXY_REJECT_INTERNAL_ERROR            5717
#define DSM_RC_PROXY_REJECT_NOT_AUTHORIZED            5722
#define DSM_RC_PROXY_INVALID_FROMNODE                 5746
#define DSM_RC_PROXY_INVALID_SERVERFREE               5747
#define DSM_RC_PROXY_INVALID_CLUSTER                  5748
```

```
#define DSM_RC_PROXY_INVALID_FUNCTION                 5749

/*============================================================================
   回覆碼 5801 - 5849 已保留用於加密法/安全
============================================================================*/

#define DSM_RC_CRYPTO_ICC_ERROR                       5801
#define DSM_RC_CRYPTO_ICC_CANNOT_LOAD                 5802
#define DSM_RC_SSL_NOT_SUPPORTED                      5803
#define DSM_RC_SSL_INIT_FAILED                        5804
#define DSM_RC_SSL_KEYFILE_OPEN_FAILED                5805
#define DSM_RC_SSL_KEYFILE_BAD_PASSWORD               5806
#define DSM_RC_SSL_BAD_CERTIFICATE                    5807

/*============================================================================
   回覆碼 6300 - 6399 已保留用於用戶端刪除重複資料
============================================================================*/
#define DSM_RC_DIGEST_VALIDATION_ERROR                6300  /* 端對端摘要驗證錯誤  */
#define DSM_RC_DATA_FINGERPRINT_ERROR                 6301 /* Rabin 指紋辨識失敗    */
#define DSM_RC_DATA_DEDUP_ERROR                       6302 /* 將資料轉換至區塊時發生錯誤 */


#endif /* _H_DSMRC */
```

相關參考

API 回覆碼

附錄 B API 類型定義原始檔

本附錄包含結構定義、類型定義以及 API 的常數。

- 第一個標頭檔 `dsmapitd.h` 和 `tsmapitd.h` 說明所有作業系統通用的定義。
- 第二個標頭檔 `dsmapips.h` 提供特定作業系統專用的定義範例；在本例中是使用 Windows 平台。
- 第三個標頭檔 `release.h` 則包含版本和版次資訊。

在這裡提供的資訊包含使用 API 配送的檔案的時間點副本。檢視 API 配送套件中的檔案以取得最新版本。

dsmapitd.h

```
/***********************************************************************
*       IBM Spectrum Protect             *
* API Client Component                                                 *
*                                                                      *
* IBM Confidential                                                     *
* (IBM Confidential-Restricted when combined with the Aggregated OCO   *
* source modules for this program)                                     *
*                                                                      *
* OCO Source Materials                                                 *
*                                                                      *
* 5648-020 (C) Copyright IBM Corporation 1993, 2018                    *
***********************************************************************/

/**************************************************************************
* 標頭檔名稱：dsmapitd.h
*
* 環境：            ***********************************************
*                   ** 這是一個 platform-independent 原始檔       **
*
*                   ***********************************************
*
* 設計注意事項：   這個檔案包含基本資料類型以及可由所有用戶端來源檔案
*                   併入的常數。 應該要適當地設定此檔案內的常數，*                  使它們適用於要在其上
*                   執行用戶端軟體
*                   的特定機器及作業系統。
*
*                   dsmapips.h 內含特定的平台定義
*
* 敘述名稱：IBM Spectrum Protect API 常數的定義
*-------------------------------------------------------------------------*/

#ifndef _H_DSMAPITD
#define _H_DSMAPITD


#include "dsmapips.h"      /* Platform specific definitions*/
#include "release.h"

/*=== 設定結構對齊來壓縮結構 ===*/
#if (_OPSYS_TYPE == DS_WINNT) && !defined(_WIN64)
#pragma pack(1)
#endif

#ifdef _MAC
/*=============================================================================
  選項有：
http://developer.apple.com/documentation/DeveloperTools/Conceptual/PowerPCRuntime/Data/
chapter_2_section_3.html

#pragma option align=<mode>
其中 <mode> 為 power、mac68k、natural 或 packed。
=============================================================================*/
#pragma options align = packed
#endif

typedef char osChar_t;

/*<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>*/
/*                    D E F I N E S                                       */
/*<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>*/
```

```
/*-------------------------------------------------------------------------+
|  API 版本，版次及層級用來使用 dsmInit() 上的 dsmApiVersion    |
+-------------------------------------------------------------------------*/
#define DSM_API_VERSION     COMMON_VERSION
#define DSM_API_RELEASE     COMMON_RELEASE
#define DSM_API_LEVEL       COMMON_LEVEL
#define DSM_API_SUBLEVEL    COMMON_SUBLEVEL

/*-------------------------------------------------------------------------+
| 最大欄位長度                                                             |
+-------------------------------------------------------------------------*/
#define DSM_MAX_CG_DEST_LENGTH        30      /* 副本群組定義 */
#define DSM_MAX_CG_NAME_LENGTH        30      /* 副本群組名稱         */
#define DSM_MAX_DESCR_LENGTH          255     /* 保存說明     */
#define DSM_MAX_DOMAIN_LENGTH         30      /* 原則網域名稱      */
#define DSM_MAX_FSINFO_LENGTH         500       /* 檔案空間資訊         */
#define DSM_MAX_USER_FSINFO_LENGTH    480     /* 最大使用者檔案空間資訊*/
#define DSM_MAX_FSNAME_LENGTH         1024    /* 檔案空間名稱          */
#define DSM_MAX_FSTYPE_LENGTH         32      /* 檔案空間類型          */
#define DSM_MAX_HL_LENGTH             1024    /* 物件高階名稱 */
#define DSM_MAX_ID_LENGTH             64      /* 階段作業節點名稱       */
#define DSM_MAX_LL_LENGTH             256     /* 物件低層級名稱 */
#define DSM_MAX_MC_NAME_LENGTH        30      /* 管理類別名稱  */
#define DSM_MAX_OBJINFO_LENGTH        255       /* 物件資訊              */
#define DSM_MAX_EXT_OBJINFO_LENGTH    1500      /* 延伸的物件資訊   */
#define DSM_MAX_OWNER_LENGTH          64      /* 物件擁有者名稱       */
#define DSM_MAX_PLATFORM_LENGTH       16      /* 應用程式類型        */
#define DSM_MAX_PS_NAME_LENGTH        30      /* 原則集名稱         */
#define DSM_MAX_SERVERTYPE_LENGTH     32      /* 伺服器平台類型   */
#define DSM_MAX_VERIFIER_LENGTH       64      /* 密碼                 */
#define DSM_PATH_MAX                  1024    /* API 配置檔路徑   */
#define DSM_NAME_MAX                  255     /* API 配置檔名稱   */
#define DSM_MAX_NODE_LENGTH           64      /* 節點/機器名稱       */
#define DSM_MAX_RC_MSG_LENGTH         1024      /* 適用於 dsmRCMsg 的 msg parm */
#define DSM_MAX_SERVER_ADDRESS        1024            /* 伺服器位址 */

#define DSM_MAX_MC_DESCR_LENGTH     DSM_MAX_DESCR_LENGTH /* mgmt 類別  */
#define DSM_MAX_SERVERNAME_LENGTH   DSM_MAX_ID_LENGTH /* 伺服器名稱    */
#define DSM_MAX_GET_OBJ               4080     /*  BeginGetData 上的最大物件 */
#define DSM_MAX_PARTIAL_GET_OBJ       1300 /* BeginGetData 上的最大部分物件*/
#define DSM_MAX_COMPRESSTYPE_LENGTH   32       /* 最大壓縮演算法名稱  */

/*-------------------------------------------------------------------------+
| 最小欄位長度                                                             |
+-------------------------------------------------------------------------*/
#define DSM_MIN_COMPRESS_SIZE  2048 /* 物件的最小位元組數量 */
                                    /* 接受壓縮之前的需要*/

/*-------------------------------------------------------------------------+
|  mtFlag 在 dsmSetup 的呼叫值                                            |
+-------------------------------------------------------------------------*/
#define DSM_MULTITHREAD      bTrue
#define DSM_SINGLETHREAD     bFalse

/*-------------------------------------------------------------------------+
|  物件類型在 dsmObjName 結構的值                                          |
|  附註：這些值必須保留在 sync with dsmcomm.h                              |
+-------------------------------------------------------------------------*/
#define DSM_OBJ_FILE                  0x01 /*物件有屬性資訊 & 資料*/
#define DSM_OBJ_DIRECTORY             0x02 /*物件僅有屬性資訊 */
#define DSM_OBJ_RESERVED1             0x04 /* 供未來使用                  */
#define DSM_OBJ_RESERVED2             0x05 /* 供未來使用                  */
#define DSM_OBJ_RESERVED3             0x06 /* 供未來使用                  */
#define DSM_OBJ_WILDCARD    0xFE  /* 任何物件類型                    */
#define DSM_OBJ_ANY_TYPE    0xFF  /* 供未來使用                      */

/*-------------------------------------------------------------------------+
| QryResp 中 compressedState 的類型定義                                    |
+-------------------------------------------------------------------------*/
#define DSM_OBJ_COMPRESSED_UNKNOWN    0
#define DSM_OBJ_COMPRESSED_YES        1
#define DSM_OBJ_COMPRESSED_NO         2

/*-------------------------------------------------------------------------+
| 「群組類型」欄位在 tsmGrouphandlerIn_t 的定義           |
+-------------------------------------------------------------------------*/
```

```
#define  DSM_GROUPTYPE_NONE         0x00   /* 不是群組成員                  */
#define  DSM_GROUPTYPE_RESERVED1    0x01   /* 供未來使用                    */
#define  DSM_GROUPTYPE_PEER         0x02   /* 對等群組                      */
#define  DSM_GROUPTYPE_RESERVED2    0x03   /* 供未來使用                    */

/*-------------------------------------------------------------------------+
| 「成員類型」欄位在 tsmGrouphandlerIn_t 的定義          |
+-------------------------------------------------------------------------*/

#define  DSM_MEMBERTYPE_LEADER      0x01   /* 群組前導  */
#define  DSM_MEMBERTYPE_MEMBER      0x02   /* 群組成員  */

/*-------------------------------------------------------------------------+
| 「作業類型」欄位在 tsmGrouphandlerIn_t 的定義         |
+-------------------------------------------------------------------------*/
#define DSM_GROUP_ACTION_BEGIN         0x01
#define DSM_GROUP_ACTION_OPEN          0x02 /* 建立新群組                    */
#define DSM_GROUP_ACTION_CLOSE         0x03 /* 確定並儲存開啟的群組          */
#define DSM_GROUP_ACTION_ADD           0x04 /* 附加到群組 */
#define DSM_GROUP_ACTION_ASSIGNTO      0x05 /* 指定至另一個群組 */
#define DSM_GROUP_ACTION_REMOVE        0x06 /* 從群組中移除成員 */

/*-------------------------------------------------------------------------+
| 查詢 Mgmt 類別回應 copySer 在 DetailCG 結構的值   |
+-------------------------------------------------------------------------*/
#define Copy_Serial_Static          1  /*複製序列化靜態         */
#define Copy_Serial_Shared_Static   2 /*複製序列化共用靜態 */
#define Copy_Serial_Shared_Dynamic  3  /*複製序列化共用動態 */
#define Copy_Serial_Dynamic         4  /*複製序列化動態         */

/*-------------------------------------------------------------------------+
| 查詢 Mgmt 類別回應 copyMode 在 DetailCG 結構的值 |
+-------------------------------------------------------------------------*/
#define Copy_Mode_Modified          1   /*修改複製模式              */
#define Copy_Mode_Absolute          2   /*絕對複製模式              */

/*-------------------------------------------------------------------------+
|  objState 在 qryBackupData 結構的值                          |
+-------------------------------------------------------------------------*/
#define DSM_ACTIVE            0x01      /* 僅查詢作用中的物件      */
#define DSM_INACTIVE          0x02      /* 僅查詢非作用的物件    */
#define DSM_ANY_MATCH         0xFF      /* 查詢所有備份物件       */

/*-------------------------------------------------------------------------+
| dsmDate.year 欄位在 qryArchiveData 結構的界限值       |
+-------------------------------------------------------------------------*/
#define DATE_MINUS_INFINITE         0x0000      /* 最低的界限         */
#define DATE_PLUS_INFINITE          0xFFFF      /* 最高的上層界限 */

/*-------------------------------------------------------------------------+
| 在 dsmUpdateFS() 上更新動作參數的位元遮罩                |
+-------------------------------------------------------------------------*/
#define DSM_FSUPD_FSTYPE              ((unsigned) 0x00000002)
#define DSM_FSUPD_FSINFO              ((unsigned) 0x00000004)
#define DSM_FSUPD_BACKSTARTDATE       ((unsigned) 0x00000008)
#define DSM_FSUPD_BACKCOMPLETEDATE    ((unsigned) 0x00000010)
#define DSM_FSUPD_OCCUPANCY           ((unsigned) 0x00000020)
#define DSM_FSUPD_CAPACITY            ((unsigned) 0x00000040)
#define DSM_FSUPD_RESERVED1           ((unsigned) 0x00000100)

/*-------------------------------------------------------------------------+
| 在 dsmUpdateObj() 上備份更新動作參數的位元遮罩          |
+-------------------------------------------------------------------------*/
#define DSM_BACKUPD_OWNER             ((unsigned) 0x00000001)
#define DSM_BACKUPD_OBJINFO           ((unsigned) 0x00000002)
#define DSM_BACKUPD_MC                ((unsigned) 0x00000004)

#define DSM_ARCHUPD_OWNER             ((unsigned) 0x00000001)
#define DSM_ARCHUPD_OBJINFO           ((unsigned) 0x00000002)
#define DSM_ARCHUPD_DESCR             ((unsigned) 0x00000004)

/*-------------------------------------------------------------------------+
|  在 dsmDeleteFS() 上儲存庫參數的值                        |
+-------------------------------------------------------------------------*/
#define DSM_ARCHIVE_REP       0x0A      /* 保存儲存庫             */
#define DSM_BACKUP_REP        0x0B      /* 備份儲存庫              */
#define DSM_REPOS_ALL         0x01      /* 所有儲存庫類型          */
```

```
/*--------------------------------------------------------------------------+
|  在 dsmEndTxn() 上投票參數的值                                            |
+--------------------------------------------------------------------------*/
#define DSM_VOTE_COMMIT  1             /* 確定目前的異動       */
#define DSM_VOTE_ABORT   2             /* 回轉目前的異動    */

/*--------------------------------------------------------------------------+
|  採用 ApiSessInfo 結構傳回各旗標的值。             |
+--------------------------------------------------------------------------*/
/* 用戶端壓縮欄位碼 */
#define COMPRESS_YES    1    /* 用戶端必須壓縮資料              */
#define COMPRESS_NO     2    /* 用戶端「不可以」壓縮資料          */
#define COMPRESS_CD     3    /* 用戶端判斷                        */

/* 保存刪除許可權碼。 */
#define ARCHDEL_YES     1    /* 接受保存刪除                 */
#define ARCHDEL_NO      2    /* 「不接受」保存刪除            */

/* 備份刪除權限代碼。 */
#define BACKDEL_YES     1    /* 接受備份刪除                  */
#define BACKDEL_NO      2    /* 「不接受」備份刪除             */


/*--------------------------------------------------------------------------+
  多種旗標返回在 optStruct 結構上的值。             |
---------------------------------------------------------------------------*/
#define DSM_PASSWD_GENERATE  1
#define DSM_PASSWD_PROMPT    0

#define DSM_COMM_TCP        1    /* tcpip         */
#define DSM_COMM_NAMEDPIPE  2    /* 指定的管道   */
#define DSM_COMM_SHM        3    /* 共用記憶體 */

/* 已作廢的 commmethods */
#define DSM_COMM_PVM_IUCV   12
#define DSM_COMM_3270       12
#define DSM_COMM_IUCV       12
#define DSM_COMM_PWSCS      12
#define DSM_COMM_SNA_LU6_2  12
#define DSM_COMM_IPXSPX     12    /* 適用於 IPX/SPX 支援 */
#define DSM_COMM_NETBIOS    12    /* NETBIOS */
#define DSM_COMM_400COMM    12
#define DSM_COMM_CLIO      12     /* CLIO/S */
/*--------------------------------------------------------------------------+
|  可供未來使用 userNameAuthorities 在 dsmInitEx 上的值              |
+--------------------------------------------------------------------------*/
#define DSM_USERAUTH_NONE     ((dsInt16_t)0x0000)
#define DSM_USERAUTH_ACCESS   ((dsInt16_t)0x0001)
#define DSM_USERAUTH_OWNER    ((dsInt16_t)0x0002)
#define DSM_USERAUTH_POLICY   ((dsInt16_t)0x0004)
#define DSM_USERAUTH_SYSTEM   ((dsInt16_t)0x0008)

/*--------------------------------------------------------------------------+
|  dsmEndSendObjEx，queryResp 上的 encryptionType 值                       |
+--------------------------------------------------------------------------*/
#define DSM_ENCRYPT_NO              ((dsUint8_t)0x00)
#define DSM_ENCRYPT_USER            ((dsUint8_t)0x01)
#define DSM_ENCRYPT_CLIENTENCRKEY   ((dsUint8_t)0x02)
#define DSM_ENCRYPT_DES_56BIT       ((dsUint8_t)0x04)
#define DSM_ENCRYPT_AES_128BIT      ((dsUint8_t)0x08)
#define DSM_ENCRYPT_AES_256BIT      ((dsUint8_t)0x10)

/*--------------------------------------------------------------------+
| mediaClass 欄位的定義。                                          |
+--------------------------------------------------------------------*/
/*
 *  下列的常數定義媒體存取權類別的階層。
 *  低值指出可以提供較快存取權至資料的媒體。
 */

/* 已修正：代表線上的類別，已修正的媒體 (如
           硬碟)。 */
#define  MEDIA_FIXED          0x10

/* 程式庫：代表可經由機械裝載裝置來存取
              的可裝載媒體類別。 */
#define  MEDIA_LIBRARY        0x20

/* 未來使用 */
```

```
#define  MEDIA_NETWORK        0x30

/* 未來使用 */
#define  MEDIA_SHELF          0x40

/* 未來使用 */
#define  MEDIA_OFFSITE        0x50

/* 未來使用 */
#define  MEDIA_UNAVAILABLE    0xF0


/*------------------------------------------------------------------------+
| dsmBeginGetData() 部分物件資料的類型定義           |
+------------------------------------------------------------------------*/
typedef struct
{
    dsUint16_t     stVersion;           /* 結構版本                 */
    dsStruct64_t   partialObjOffset; /* 偏移至物件來開始讀取*/
    dsStruct64_t   partialObjLength;   /* 讀取的物件數量         */
} PartialObjData ;               /* 部分物件資料                    */

#define PartialObjDataVersion 1  /*                                        */

/*------------------------------------------------------------------------+
日期結構的類型定義                                    |
+------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t    year;                  /* 年，16 位元 整數 (例如，1990)  */
   dsUint8_t     month;              /* 月，8 位元整數 (1 - 12)       */
   dsUint8_t     day;                   /* 日。8 位元整數 (1 - 31)        */
   dsUint8_t     hour;                  /* 小時，8 位元整數 (0 - 23)      */
   dsUint8_t     minute;             /* 分鐘，8 位元整數 (0 - 59)    */
   dsUint8_t     second;             /* 秒，b 位元整數 (0 - 59)     */
}dsmDate ;

/*------------------------------------------------------------------------+
|  「物件 ID」在 dsmGetObj() 及 dsmGetList structure| 上的類型定義
+------------------------------------------------------------------------*/
typedef dsStruct64_t  ObjID ;

/*------------------------------------------------------------------------+
| dsmQueryBuff 在 dsmBeginQuery() 上的類型定義                      |
+------------------------------------------------------------------------*/
typedef void dsmQueryBuff ;

/*------------------------------------------------------------------------+
| dsmGetType 參數在 dsmBeginGetData() 上的類型定義           |
+------------------------------------------------------------------------*/
typedef enum
{
        gtBackup = 0x00,                    /* 備份處理類型    */
        gtArchive                           /* 保存處理類型   */
} dsmGetType ;

/*------------------------------------------------------------------------+
|  dsmQueryType 參數在 dsmBeginQuery() 上的類型定義          |
+------------------------------------------------------------------------*/
typedef enum
{
   qtArchive = 0x00,                              /* 保存查詢類型 */
   qtBackup,                                       /* 備份查詢類型 */
   qtBackupActive,                 /* 作用中備份檔的快取查詢 */
   qtFilespace,                                   /* 檔案空間查詢類型 */
   qtMC,                           /* 管理類別查詢類型              */
   qtReserved1,                    /* 未來使用                          */
   qtReserved2,                    /* 未來使用                          */
   qtReserved3,                    /* 未來使用                          */
   qtReserved4,                    /* 未來使用                          */
   qtBackupGroups,                 /* 在特定的檔案的群組前導      */
   qtOpenGroups,                   /* 開啟特定的檔案中的群組        */
   qtReserved5,                    /* 未來使用                          */
   qtProxyNodeAuth,                /* 節點可 proxy 至    */
   qtProxyNodePeer,                /* 具有相同目標的對等節點    */
   qtReserved6,                    /* 未來使用                          */
   qtReserved7,                    /* 未來使用                          */
   qtReserved8                     /* 未來使用                          */
```

```
}dsmQueryType ;

/*------------------------------------------------------------------------+
|  sendType 參數在 dsmBindMC() and dsmSendObj() 上的類型定義     |
+------------------------------------------------------------------------*/
typedef enum
{
   stBackup = 0x00,                          /* 備份處理類型    */
   stArchive,                                /* 保存處理類型   */
   stBackupMountWait,           /* 備份處理和 mountwait 上   */
   stArchiveMountWait           /* 保存處理和 mountwait 上  */
}dsmSendType ;

/*------------------------------------------------------------------------+
|  delType 參數在 dsmDeleteObj() 上的類型定義                |
+------------------------------------------------------------------------*/
typedef enum
{
   dtArchive = 0x00,                                /* 保存刪除類型 */
   dtBackup,                          /* 備份刪除(關閉)類型 */
   dtBackupID                        /* 備份刪除(移除)      類型 */
}dsmDelType ;

/*------------------------------------------------------------------------+
|  sendType 參數在 dsmSetAccess () 上的類型定義                   |
+------------------------------------------------------------------------*/
typedef enum
{
   atBackup = 0x00,                             /* 備份處理類型 */
   atArchive                                   /* 保存處理類型 */
}dsmAccessType;

/*------------------------------------------------------------------------+
|  「API 版本」在 dsmInit() 及 dsmQueryApiVersion() 上的類型定義  |
+------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t version;       /* API 版本                      */
   dsUint16_t release;       /* API 版次                      */
   dsUint16_t level;         /* API 層級                        */
}dsmApiVersion;

/*------------------------------------------------------------------------+
|  「API 版本」在 dsmInit() 及 dsmQueryApiVersion() 上的類型定義  |
+------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t     stVersion;           /* 結構版本                */
   dsUint16_t version;       /* API 版本                      */
   dsUint16_t release;       /* API 版次                      */
   dsUint16_t level;         /* API 層級                        */
   dsUint16_t subLevel;      /* API 子層                     */
   dsmBool_t  unicode;       /* API Unicode?                     */
}dsmApiVersionEx;


#define apiVersionExVer       2

/*------------------------------------------------------------------------+
|  應用程式版本在 dsmInit() 上的類型定義                                  |
+------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t     stVersion;           /* 結構版本                */
   dsUint16_t   applicationVersion;   /* 應用程式版本號碼    */
   dsUint16_t   applicationRelease;   /* 應用程式版次號碼    */
   dsUint16_t   applicationLevel;     /* 應用程式層次號碼      */
   dsUint16_t   applicationSubLevel;  /* 應用程式子層次號碼 */
} dsmAppVersion;

#define appVersionVer      1

/*------------------------------------------------------------------------+
|  在「BindMC、傳送、刪除及查詢」上所使用物件名稱的類型定義                |
+------------------------------------------------------------------------*/
typedef struct S_dsmObjName
{
   char       fs[DSM_MAX_FSNAME_LENGTH + 1] ;              /* 檔案空間名稱 */
   char       hl[DSM_MAX_HL_LENGTH + 1] ;                 /* 高階名稱 */
```

```
   char       ll[DSM_MAX_LL_LENGTH + 1] ;                    /* 低階名稱 */
   dsUint8_t  objType;          /* 如需物件類型值，請參閱上述的定義 */
}dsmObjName;


/*-----------------------------------------------------------------------+
|  「備份」刪除資訊在 dsmDeleteObj() 上的類型定義                        |
+-----------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t        stVersion ;                    /* 結構版本      */
   dsmObjName       *objNameP ;                    /* 物件名稱 */
   dsUint32_t       copyGroup ;                    /* 副本群組             */
}delBack ;

#define  delBackVersion    1

/*-----------------------------------------------------------------------+
|  「保存」刪除資訊在 dsmDeleteObj() 上的類型定義                        |
+-----------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t        stVersion ;                    /* 結構版本      */
   dsStruct64_t       objId ;                        /* 物件 ID         */
}delArch ;

#define  delArchVersion  1

/*-----------------------------------------------------------------------+
|  「備份 ID」刪除資訊在 dsmDeleteObj() 上的類型定義                     |
+-----------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t        stVersion ;                    /* 結構版本      */
   dsStruct64_t       objId ;                        /* 物件 ID         */
}delBackID;

#define  delBackIDVersion  1

/*-----------------------------------------------------------------------+
|  刪除資訊在 dsmDeleteObj() 上的類型定義                                |
+-----------------------------------------------------------------------*/
typedef union
{
   delBack   backInfo ;
   delArch   archInfo ;
   delBackID backIDInfo ;
}dsmDelInfo ;

/*-----------------------------------------------------------------------+
|  「物件屬性」參數在 dsmSendObj() 上的類型定義                          |
+-----------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t     stVersion;           /* 結構版本                */
   char         owner[DSM_MAX_OWNER_LENGTH + 1];      /* 物件擁有者 */
   dsStruct64_t sizeEstimate;             /* 物件的大小預估 (以位元組為單位) */
   dsmBool_t    objCompressed;                  /* 物件是否已壓縮？ */
   dsUint16_t   objInfoLength;             /* object-dependent 資訊長度   */
   char         *objInfo;      /* 物件導向的資訊 */
   char         *mcNameP;              /* 用來置換的 mgmnt 類別名稱  */
   dsmBool_t    disableDeduplication;         /* 不對此物件強制執行刪除重複資料    */
   dsmBool_t    useExtObjInfo;                /* 使用延伸物件資訊，最多 1536 */
}ObjAttr;

#define ObjAttrVersion  4



/*-----------------------------------------------------------------------+
| mcBindKey 傳回在 dsmBindMC() 上的類型定義                              |
+-----------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t     stVersion;   /* 結構版本  */
   char        mcName[DSM_MAX_MC_NAME_LENGTH + 1];
                                       /* 連結至物件的 MC 名稱。 */
   dsmBool_t   backup_cg_exists;                          /* 是/否 */
   dsmBool_t   archive_cg_exists;                         /* 是/否 */
```

```
   char        backup_copy_dest[DSM_MAX_CG_DEST_LENGTH + 1];
                                                /* 備份副本目的地名稱 */
   char        archive_copy_dest[DSM_MAX_CG_DEST_LENGTH + 1];
                                                  /* 保存副本 dest.name */
}mcBindKey;

#define mcBindKeyVersion  1


/*-------------------------------------------------------------------------+
| 物件清單在 dsmBeginGetData() 上的類型定義                    |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t        stVersion ;                    /* 結構版本      */
   dsUint32_t       numObjId ;       /* 清單中物件數 ID */
   ObjID           *objId ;             /* 物件 ID 的清單還原*/
   PartialObjData  *partialObjData;     /*部分物件資料資訊的清單 */
}dsmGetList ;

#define dsmGetListVersion     2  /* 如果未使用部分物件資料的預設值 */
#define dsmGetListPORVersion  3  /* 如果使用部分物件資料的版本     */

/*-------------------------------------------------------------------------+
|  使用的 DataBlk 來「取得」或「傳送」資料的類型定義                    |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t        stVersion ;                    /* 結構版本      */
   dsUint32_t bufferLen;                /* 緩衝區傳送長度如下  */
   dsUint32_t numBytes;                  /* 讀取實際位元組數量從 */
                                         /* 或寫入至緩衝區 */
   char       *bufferPtr;                             /* 資料緩衝區 */
   dsUint32_t numBytesCompressed;        /* 在傳送實際位元組壓縮 */
   dsUint16_t reserved;                  /* 供未來使用                  */
}DataBlk;

#define DataBlkVersion  3


/*-------------------------------------------------------------------------+
|  Mgmt 類別 queryBuffer 在 dsmBeginQuery() 上的類型定義                  |
+-------------------------------------------------------------------------*/
typedef struct S_qryMCData
{
   dsUint16_t     stVersion;   /* 結構版本  */
   char         *mcName;                              /* Mgmt 類別名稱 */
                        /* 使用單一名稱來取得一個，或使用空字串來產生全部*/
   dsmBool_t    mcDetail;                              /* 是否需要詳細資訊？ */
}qryMCData;

#define qryMCDataVersion  1

/*=== 適用於 RETINIT 的值 ===*/
#define ARCH_RETINIT_CREATE 0
#define ARCH_RETINIT_EVENT  1

/*-------------------------------------------------------------------------+
|  「保存副本群組」詳細資訊在查詢 MC 回應上的類型定義                    |
+-------------------------------------------------------------------------*/
typedef struct S_archDetailCG
{
   char         cgName[DSM_MAX_CG_NAME_LENGTH + 1];       /* 副本群組名稱 */
   dsUint16_t   frequency;                         /* 複製 (保存) 頻率 */
   dsUint16_t   retainVers;                              /* 保留版本 */
   dsUint8_t    copySer;       /* 如需複製序列化值，請參閱定義 */
   dsUint8_t    copyMode;         /* 如需複製模式值，請參閱上述的定義 */
   char         destName[DSM_MAX_CG_DEST_LENGTH + 1];      /* 複製 dest 名稱 */
   dsmBool_t    bLanFreeDest;           /* 目的地是否有不需 LAN 路徑？    */
   dsmBool_t    reserved;                /* 目前未使用                 */
   dsUint8_t    retainInit;             /* 如需可能的值，請參閱上述的定義     */
   dsUint16_t   retainMin;              /* 若 retInit 是事件日期數 */
   dsmBool_t    bDeduplicate;            /* 目的地已啟用刪除重複資料    */
}archDetailCG;

/*-------------------------------------------------------------------------+
|  「備份副本群組」詳細資訊在查詢 MC 回應上的類型定義                    |
+-------------------------------------------------------------------------*/
```

```
typedef struct S_backupDetailCG
{
   char         cgName[DSM_MAX_CG_NAME_LENGTH + 1];       /* 副本群組名稱 */
   dsUint16_t   frequency;                                /* 備份頻率 */
   dsUint16_t   verDataExst;                             /* 現存資料的版本數 */
   dsUint16_t   verDataDltd;                            /* 刪除資料的版本數 */
   dsUint16_t   retXtraVers;                            /* 保留額外版本 */
   dsUint16_t   retOnlyVers;                             /* 保留唯一版本 */
   dsUint8_t    copySer;       /* 如需複製序列化值，請參閱定義 */
   dsUint8_t    copyMode;         /* 如需複製模式值，請參閱上述的定義 */
   char         destName[DSM_MAX_CG_DEST_LENGTH + 1];      /* 複製 dest 名稱 */
   dsmBool_t    bLanFreeDest;           /* 目的地是否有不需 LAN 路徑？    */
   dsmBool_t    reserved;               /* 目前未使用                */
   dsmBool_t    bDeduplicate;           /* 目的地已啟用刪除重複資料    */
}backupDetailCG;


/*--------------------------------------------------------------------------+
|  「查詢 Mgmt 類別」詳細資訊回應在 dsmGetNextQObj()| 上的類型定義         |
+--------------------------------------------------------------------------*/
typedef struct S_qryRespMCDetailData
{
   dsUint16_t     stVersion;   /* 結構版本  */
   char            mcName[DSM_MAX_MC_NAME_LENGTH + 1];            /* mc 名稱 */
   char            mcDesc[DSM_MAX_MC_DESCR_LENGTH + 1]; /*mc 說明 */
   archDetailCG    archDet;                 /* 保存副本群組詳細資訊 */
   backupDetailCG  backupDet;               /* 備份副本群組詳細資訊 */
}qryRespMCDetailData;

#define qryRespMCDetailDataVersion  4

/*--------------------------------------------------------------------------+
| 「查詢 Mgmt 類別」彙總回應在 dsmGetNextQObj()| 上的類型定義              |
+--------------------------------------------------------------------------*/
typedef struct S_qryRespMCData
{
   dsUint16_t     stVersion;   /* 結構版本  */
   char            mcName[DSM_MAX_MC_NAME_LENGTH + 1];           /* mc 名稱 */
   char         mcDesc[DSM_MAX_MC_DESCR_LENGTH + 1];       /* mc 說明 */
}qryRespMCData;

#define qryRespMCDataVersion  1



/*--------------------------------------------------------------------------+
|  保存 queryBuffer 在 dsmBeginQuery() 上的類型定義                        |
+--------------------------------------------------------------------------*/
typedef struct S_qryArchiveData
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsmObjName   *objName;                  /* 完整 dsm 物件名稱 */
   char        *擁有者；            /* 擁有者名稱 */
                     /* 如需日期界限最大值，請參閱上述的定義 */
   dsmDate      insDateLowerBound;      /* 低界限保存插入日期 */
   dsmDate      insDateUpperBound;       /* 高界限保存插入日期 */
   dsmDate      expDateLowerBound;          /* 低界限到期日 */
   dsmDate      expDateUpperBound;           /* 高界限到期日 */
   char         *descr;                    /* 保存說明 */
} qryArchiveData;

#define qryArchiveDataVersion  1



/*=== retentionInitiated 欄位的值 ===*/
#define DSM_ARCH_RETINIT_UNKNOWN 0 /* 不明的 ret init (下層 srv)  */
#define DSM_ARCH_RETINIT_STARTED 1 /* 保留時鐘已啟動             */
#define DSM_ARCH_RETINIT_PENDING 2 /* 保留時鐘未啟動        */


/*=== objHeld 的值 ===*/
#define DSM_ARCH_HELD_UNKNOWN 0    /* 不明的保留狀態 (下層 srv)  */
#define DSM_ARCH_HELD_FALSE   1    /* 物件「沒有」在刪除保留的狀態  */
#define DSM_ARCH_HELD_TRUE    2    /* 物件在刪除保留的狀態       */

/*--------------------------------------------------------------------------+
| 「查詢保存」回應在 dsmGetNextQObj() 上的類型定義                          |
```

```
+------------------------------------------------------------------------*/
typedef struct S_qryRespArchiveData
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsmObjName      objName;                  /* 檔案空間名稱限定元 */
   dsUint32_t      copyGroup;                          /* 副本群組號碼 */
   char            mcName[DSM_MAX_MC_NAME_LENGTH + 1];            /* mc 名稱 */
   char            owner[DSM_MAX_OWNER_LENGTH + 1];           /* 擁有者名稱 */
   dsStruct64_t    objId;                                  /* 唯一的副本 ID */
   dsStruct64_t    reserved;                        /* 舊版相容性 */
   dsUint8_t       mediaClass;                          /* 媒體存取權類別 */
   dsmDate         insDate;                        /* 保存插入日期 */
   dsmDate         expDate;                     /* 物件到期日 */
   char            descr[DSM_MAX_DESCR_LENGTH + 1];   /* 保存說明 */
   dsUint16_t      objInfolen;             /* object-dependent 資訊長度*/
   char            reservedObjInfo[DSM_MAX_OBJINFO_LENGTH];  /*相依於物件的資訊 */
   dsUint160_t     restoreOrderExt;                          /* 還原次序 */
   dsStruct64_t    sizeEstimate;              /* 使用者儲存預估大小 */
   dsUint8_t       compressType;                        /* 壓縮旗標*/
   dsUint8_t       retentionInitiated;   /* 物件在保留事件上等待*/
   dsUint8_t       objHeld;  /*物件在保留上「保留」，請參閱上述的值*/
   dsUint8_t       encryptionType;          /* 加密類型 */
   dsmBool_t       clientDeduplicated;      /* API 對物件刪除重複資料*/
   char            objInfo[DSM_MAX_EXT_OBJINFO_LENGTH];  /*相依於物件的資訊 */
   char            compressAlg[DSM_MAX_COMPRESSTYPE_LENGTH + 1];  /* 壓縮演算法名稱 */
}qryRespArchiveData;

#define qryRespArchiveDataVersion  7

/*------------------------------------------------------------------------+
|  保存 sendBuff 參數在 dsmSendObj() 上的類型定義                          |
+------------------------------------------------------------------------*/
typedef struct S_sndArchiveData
{
   dsUint16_t     stVersion;   /* 結構版本  */
   char         *descr;                    /* 保存說明 */
}sndArchiveData;

#define sndArchiveDataVersion  1

/*------------------------------------------------------------------------+
|  備份 queryBuffer 在 dsmBeginQuery() 上的類型定義                        |
+------------------------------------------------------------------------*/
typedef struct S_qryBackupData
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsmObjName  *objName;          /* 完整 dsm 物件名稱 */
   char        *擁有者；          /* 擁有者名稱 */
   dsUint8_t   objState;          /* 物件狀態選擇器 */
   dsmDate     pitDate;           /* 時間點還原的日期值 */
                                  /* 如需可能的值，請參閱上述的定義 */
}qryBackupData;

#define qryBackupDataVersion  2

typedef struct
{
  dsUint8_t    reserved1;
  dsStruct64_t reserved2;
} reservedInfo_t;                /* 供未來使用 */

/*------------------------------------------------------------------------+
| 「查詢備份」回應在 dsmGetNextQObj() 上的類型定義                         |
+------------------------------------------------------------------------*/
typedef struct S_qryRespBackupData
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsmObjName  objName;                        /* 完整 dsm 物件名稱 */
   dsUint32_t      copyGroup;                          /* 副本群組號碼 */
   char            mcName[DSM_MAX_MC_NAME_LENGTH + 1];            /* mc 名稱 */
   char            owner[DSM_MAX_OWNER_LENGTH + 1];           /* 擁有者名稱 */
   dsStruct64_t    objId;                                  /* 唯一的物件 ID */
   dsStruct64_t    reserved;                        /* 舊版相容性 */
   dsUint8_t       mediaClass;                          /* 媒體存取權類別 */
   dsUint8_t       objState;                       /* 物件狀態，作用中等等。 */
   dsmDate         insDate;                          /* 備份插入日期 */
   dsmDate         expDate;                     /* 物件到期日 */
```

```
   dsUint16_t      objInfolen;             /* object-dependent 資訊長度*/
   char            reservedObjInfo[DSM_MAX_OBJINFO_LENGTH];  /*相依於物件的資訊 */
   dsUint160_t     restoreOrderExt;                          /* 還原次序 */
   dsStruct64_t    sizeEstimate;             /* 使用者儲存的預估大小 */
   dsStruct64_t    baseObjId;
   dsUint16_t      baseObjInfolen;             /* 基本 object-dependent 資訊長度*/
   dsUint8_t       baseObjInfo[DSM_MAX_OBJINFO_LENGTH];  /* 基本 object-dependent 資訊 */
   dsUint160_t     baseRestoreOrder;                         /* 還原次序 */
   dsUint32_t     fsID;
   dsUint8_t       compressType;
   dsmBool_t       isGroupLeader;
   dsmBool_t       isOpenGroup;
   dsUint8_t       reserved1;               /* 供未來使用 */
   dsmBool_t       reserved2;                  /* 供未來使用 */
   dsUint16_t      reserved3;               /* 供未來使用 */
   reservedInfo_t  *reserved4;              /* 供未來使用 */
   dsUint8_t       encryptionType;          /* 加密類型 */
   dsmBool_t       clientDeduplicated;      /* API 對物件刪除重複資料*/
   char            objInfo[DSM_MAX_EXT_OBJINFO_LENGTH];  /*相依於物件的資訊 */
   char            compressAlg[DSM_MAX_COMPRESSTYPE_LENGTH + 1];  /* 壓縮演算法名稱 */
}qryRespBackupData;

#define qryRespBackupDataVersion  8

/*------------------------------------------------------------------------+
|  作用中備份 queryBuffer 在 dsmBeginQuery() 上的類型定義
|
|  附註：  如需作用中備份查詢，僅需要設定 objName 的檔案空間及 objType 欄位。
|          objType 僅能設定為
|          DSM_OBJ_FILE 或 DSM_OBJ_DIRECTORY。  DSM_OBJ_ANY_TYPE 將無法
|          在查詢上尋找符合項目。
+------------------------------------------------------------------------*/
typedef struct S_qryABackupData
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsmObjName      *objName;                /* 僅使用 fs 及 objtype */
}qryABackupData;

#define qryABackupDataVersion  1

/*------------------------------------------------------------------------+
| 「查詢作用中備份」回應在 dsmGetNextQObj() 上的類型定義                  |
+------------------------------------------------------------------------*/
typedef struct S_qryARespBackupData
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsmObjName  objName;                     /* 完整 dsm 物件名稱 */
   dsUint32_t      copyGroup;                          /* 副本群組號碼 */
   char        mcName[DSM_MAX_MC_NAME_LENGTH + 1];/*管理類別名稱*/
   char            owner[DSM_MAX_OWNER_LENGTH + 1];            /* 擁有者名稱 */
   dsmDate         insDate;                       /* 備份插入日期 */
   dsUint16_t  objInfolen;             /* object-dependent 資訊長度*/
   char        reservedObjInfo[DSM_MAX_OBJINFO_LENGTH];  /*相依於物件的資訊 */
   char        objInfo[DSM_MAX_EXT_OBJINFO_LENGTH];  /*相依於物件的資訊 */
}qryARespBackupData;

#define qryARespBackupDataVersion  2

/*------------------------------------------------------------------------+
|  備份 queryBuffer 在 dsmBeginQuery() 上的類型定義                       |
+------------------------------------------------------------------------*/
typedef struct qryBackupGroups
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsUint8_t    groupType;
   char         *fsName;
   char           *owner;
   dsStruct64_t groupLeaderObjId;
   dsUint8_t    objType;
   dsmBool_t    noRestoreOrder;
   dsmBool_t    noGroupInfo;
   char         *hl;
}qryBackupGroups;

#define qryBackupGroupsVersion 3


/*------------------------------------------------------------------------+
```

```
|  proxynode queryBuffer 在 dsmBeginQuery() 上的類型定義                    |
+--------------------------------------------------------------------------*/
typedef struct qryProxyNodeData
{
   dsUint16_t     stVersion;   /* 結構版本  */
   char        *targetNodeName;             /* 目標節點名稱  */
}qryProxyNodeData;

#define qryProxyNodeDataVersion  1

/*--------------------------------------------------------------------------+
| qryRespProxyNodeData 使用的參數在 dsmGetNextQObj()| 上的類型定義         |
+--------------------------------------------------------------------------*/

typedef struct
{
   dsUint16_t        stVersion ;                    /* 結構版本      */
   char            targetNodeName[DSM_MAX_ID_LENGTH+1]; /* 目標節點名稱  */
   char            peerNodeName[DSM_MAX_ID_LENGTH+1];   /* 對等節點名稱    */
   char            hlAddress[DSM_MAX_ID_LENGTH+1];      /* 同層級 hlAddress    */
   char            llAddress[DSM_MAX_ID_LENGTH+1];      /* 同層級 hlAddress    */
}qryRespProxyNodeData;

#define qryRespProxyNodeDataVersion  1


/*--------------------------------------------------------------------------+
|  WINNT 及 OS/2 檔案空間屬性的類型定義                                     |
+--------------------------------------------------------------------------*/
typedef struct
{
   char         driveLetter ;          /* 檔案空間的磁碟機代號    */
   dsUint16_t   fsInfoLength;          /* 使用的 fsInfo 長度            */
   char         fsInfo[DSM_MAX_FSINFO_LENGTH];/*caller-determined 資料  */
}dsmDosFSAttrib ;

/*--------------------------------------------------------------------------+
|  UNIX 檔案空間屬性的類型定義                                              |
+--------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t   fsInfoLength;          /* 使用的 fsInfo 長度            */
   char         fsInfo[DSM_MAX_FSINFO_LENGTH];/*caller-determined 資料  */
}dsmUnixFSAttrib ;

/*--------------------------------------------------------------------------+
|  「NetWare 檔案空間」屬性的類型定義                                       |
+--------------------------------------------------------------------------*/
typedef dsmUnixFSAttrib dsmNetwareFSAttrib;

/*--------------------------------------------------------------------------+
|  「檔案空間」屬性在所有「檔案空間」呼叫上的類型定義                       |
+--------------------------------------------------------------------------*/
typedef union
{
   dsmNetwareFSAttrib  netwareFSAttr;
   dsmUnixFSAttrib     unixFSAttr ;
   dsmDosFSAttrib      dosFSAttr ;
}dsmFSAttr ;

/*--------------------------------------------------------------------------+
|  fsUpd 參數在 dsmUpdateFS() 上的類型定義
+--------------------------------------------------------------------------*/
typedef struct S_dsmFSUpd
{
   dsUint16_t        stVersion ;                    /* 結構版本      */
   char            *fsType ;           /* 檔案空間類型                */
   dsStruct64_t    occupancy ;             /* 佔用預估              */
   dsStruct64_t    capacity  ;             /* 容量預估               */
   dsmFSAttr       fsAttr ;                    /* 平台特定的屬性 */
}dsmFSUpd ;

#define dsmFSUpdVersion  1

/*--------------------------------------------------------------------------+
|  檔案空間 queryBuffer 在 dsmBeginQuery() 上的類型定義                     |
+--------------------------------------------------------------------------*/
typedef struct S_qryFSData
{
```

```
   dsUint16_t     stVersion;   /* 結構版本  */
   char        *fsName;                   /* 檔案空間名稱 */
}qryFSData;

#define qryFSDataVersion  1

/*-------------------------------------------------------------------------+
| 「查詢檔案空間」回應在 dsmGetNextQObj() 上的類型定義                     |
+-------------------------------------------------------------------------*/
typedef struct S_qryRespFSData
{
   dsUint16_t     stVersion;   /* 結構版本  */
   char           fsName[DSM_MAX_FSNAME_LENGTH + 1];          /* 檔案空間名稱 */
   char           fsType[DSM_MAX_FSTYPE_LENGTH + 1] ;         /* 檔案空間類型 */
   dsStruct64_t   occupancy;                          /* 佔用預估 ( 以位元組為單位 ) 。*/
   dsStruct64_t   capacity;                           /* 容量預估 ( 以位元組為單位 ) 。 */
   dsmFSAttr       fsAttr ;                    /* 平台特定的屬性 */
   dsmDate        backStartDate;            /* 啟動備份日期               */
   dsmDate        backCompleteDate;         /* 結束備份日期                 */
   dsmDate        reserved1;                    /* 供未來使用                 */
   dsmDate        lastReplStartDate;         /* 前次開始抄寫的時間 */
   dsmDate        lastReplCmpltDate;         /* 前次完成抄寫的時間    */
                                             /*    ( 可能失敗 ,            */
                                             /*     但仍然完成了 )             */
   dsmDate        lastBackOpDateFromServer; /* 前次在伺服器上儲存用戶端  */
                                             /*     的儲存時間戳記                 */
   dsmDate        lastArchOpDateFromServer; /* 前次在伺服器上儲存用戶端  */
                                             /*     的儲存時間戳記                 */
   dsmDate        lastSpMgOpDateFromServer; /* 前次在伺服器上儲存用戶端  */
                                             /*     的儲存時間戳記                 */
   dsmDate        lastBackOpDateFromLocal;  /* 前次在本地儲存用戶端  */
                                             /*     的儲存時間戳記                  */
   dsmDate        lastArchOpDateFromLocal;  /* 前次在本地儲存用戶端  */
                                             /*     的儲存時間戳記                  */
   dsmDate        lastSpMgOpDateFromLocal;  /* 前次在本地儲存用戶端  */
                                             /*     的儲存時間戳記                  */
   dsInt32_t      failOverWriteDelay;       /* 等待分鐘數 , 此後才容許將用戶端  */
                                             /* 儲存到此 Repl srvr , 特殊代碼 :  */
                                             /* NO_ACCESS(-1) 和 ACCESS_RDONLY (-2)           */
}qryRespFSData;

#define qryRespFSDataVersion 4

/*-------------------------------------------------------------------------+
|  regFilespace 參數在 dsmRegisterFS() 上的類型定義
+-------------------------------------------------------------------------*/
typedef struct S_regFSData
{
   dsUint16_t     stVersion;   /* 結構版本  */
   char            *fsName;                              /* 檔案空間名稱 */
   char            *fsType;                              /* 檔案空間類型 */
   dsStruct64_t    occupancy;                        /* 佔用預估 ( 以位元組為單位 ) 。 */
   dsStruct64_t    capacity;                          /* 容量預估 ( 以位元組為單位 ) 。 */
   dsmFSAttr       fsAttr ;                    /* 平台特定的屬性 */
}regFSData;

#define regFSDataVersion  1

/*-------------------------------------------------------------------------+
| 使用的 dedupType 在 apisessInfo 上的類型定義                             |
+-------------------------------------------------------------------------*/
typedef enum
{
   dedupServerOnly= 0x00,             /* 僅在伺服器上執行刪除重複資料     */
   dedupClientOrServer               /* 可在用戶端或伺服器上執行刪除重複資料 */
}dsmDedupType ;

/*-------------------------------------------------------------------------+
 | 失效接手配置和狀態的類型定義
 -------------------------------------------------------------------------*/
typedef enum
{
   failOvrNotConfigured = 0x00,
   failOvrConfigured,
   failOvrConnectedToReplServer
}dsmFailOvrCfgType ;

/*-------------------------------------------------------------------------+
```

```
|  階段作業資訊回應在 dsmQuerySessionInfo() 上的類型定義                    |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t     stVersion;           /* 結構版本                 */
      /*-------------------------------------------------------------------*/
      /*           伺服器資訊                                              */
      /*-------------------------------------------------------------------*/
   char           serverHost[DSM_MAX_SERVERNAME_LENGTH+1];
                                  /* DSM 伺服器的網路主機名稱  */
   dsUint16_t     serverPort;           /* 伺服器 comm 埠在主機上          */
   dsmDate        serverDate;           /* 伺服器的日期 / 時間                */
   char           serverType[DSM_MAX_SERVERTYPE_LENGTH+1];
                                 /* 伺服器的執行平台       */
   dsUint16_t     serverVer;            /* 伺服器的版本號碼           */
   dsUint16_t         serverRel;     /* 伺服器的版本號碼           */
   dsUint16_t         serverLev;     /* 伺服器的層級號碼             */
   dsUint16_t         serverSubLev;  /* 伺服器的次層級號碼          */
      /*-------------------------------------------------------------------*/
      /*           用戶端預設值                                            */
      /*-------------------------------------------------------------------*/
   char           nodeType[DSM_MAX_PLATFORM_LENGTH+1]; /*節點 / 應用程式類型*/
   char           fsdelim;              /* 檔案空間定界符號          */
   char           hldelim;              /* 介於 highlev & lowlev 的定界符號 */
   dsUint8_t      compression;          /* 壓縮旗標                   */
   dsUint8_t      archDel;              /* 保存刪除許可權         */
   dsUint8_t      backDel;              /* 備份刪除許可權         */
   dsUint32_t     maxBytesPerTxn;       /* 供未來使用                    */
   dsUint16_t     maxObjPerTxn;         /* 在 txn 中可接受的最大物件 */
      /*-------------------------------------------------------------------*/
      /*           階段作業資訊                                            */
      /*-------------------------------------------------------------------*/
   char       id[DSM_MAX_ID_LENGTH+1];    /* 登入 ID 節點名稱          */
   char       owner[DSM_MAX_OWNER_LENGTH+1]; /* 登入擁有者              */
                                 /*   ( 適用於多使用者平台 )     */
   char       confFile[DSM_PATH_MAX + DSM_NAME_MAX +1];
                                 /* len 是平台 dep              */
                                 /* appl 配置檔 dsInit 名稱  */
   dsUint8_t      opNoTrace;            /* dsInit 選項 - NOTRACE = 1      */
      /*-------------------------------------------------------------------*/
      /*           原則資料                                                */
      /*-------------------------------------------------------------------*/
   char           domainName[DSM_MAX_DOMAIN_LENGTH+1]; /* 網域名稱          */
   char           policySetName[DSM_MAX_PS_NAME_LENGTH+1];
                                 /* 作用原則集名稱            */
   dsmDate        polActDate;           /* 原則集啟動日期       */
   char           dfltMCName[DSM_MAX_MC_NAME_LENGTH+1];/* 預設值 Mgmt 類別  */
   dsUint16_t     gpBackRetn;           /* 寬限期備份保留     */
   dsUint16_t     gpArchRetn;           /* 寬限期保存保留    */
   char           adsmServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* ADSM 伺服器名稱 */
   dsmBool_t      archiveRetentionProtection; /* 是否啟用伺服器保留保護 */
   dsStruct64_t   maxBytesPerTxn_64;       /* 供未來使用                   */
   dsmBool_t      lanFreeEnabled;        /* 已設定不需 LAN 選項          */
   dsmDedupType   dedupType;             /* 伺服器或 clientOrServer       */
   char           accessNode[DSM_MAX_ID_LENGTH+1];    /* 作為節點名稱          */

   /*-------------------------------------------------------------------*/
   /*           抄寫和失效接手資訊                                      */
   /*-------------------------------------------------------------------*/
   dsmFailOvrCfgType failOverCfgType; /* 失效接手狀態 */
   char           replServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* 抄寫伺服器名稱 */
   char           homeServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* 起始伺服器名稱 */
   char           replServerHost[DSM_MAX_SERVERNAME_LENGTH+1]; /* DSM 伺服器的網路主機名稱  */
   dsInt32_t      replServerPort;                              /* 主機上的伺服器通訊埠          */

}ApiSessInfo;

#define ApiSessInfoVersion  6



/*-------------------------------------------------------------------------+
| 「查詢保存」回應在 dsmQueryCliOptions()                                   |
|      和 dsmQuerySessOptions() 上的類型定義                                |
+-------------------------------------------------------------------------*/

typedef struct
```

```
{
   char            dsmiDir[DSM_PATH_MAX + DSM_NAME_MAX +1];
   char            dsmiConfig[DSM_PATH_MAX + DSM_NAME_MAX +1];
   char        serverName[DSM_MAX_SERVERNAME_LENGTH+1];
   dsInt16_t   commMethod;
   char        serverAddress[DSM_MAX_SERVER_ADDRESS];
   char        nodeName[DSM_MAX_NODE_LENGTH+1];
   dsmBool_t   compression;
   dsmBool_t   compressalways;
   dsmBool_t   passwordAccess;
}optStruct ;

/*-------------------------------------------------------------------------+
| 使用的 LogType 在 logInfo 上的類型定義                                    |
+-------------------------------------------------------------------------*/
typedef enum
{
   logServer = 0x00,                    /* 僅日誌 msg 至伺服器    */
   logLocal,                       /* 僅日誌 msg 至本端錯誤日誌 */
   logBoth,                         /* 將訊息記錄到伺服器和本端錯誤日誌 */
   logNone
}dsmLogType ;

/*-------------------------------------------------------------------------+
| logInfo 使用的參數在 dsmLogEvent() 上的類型定義                           |
+-------------------------------------------------------------------------*/

typedef struct
{
   char        *message;   /* 記載的訊息文字 */
   dsmLogType  logType;    /* 日誌類型：本端、伺服器或兩者 */
}logInfo;

/*-------------------------------------------------------------------------+
| 使用的 qryRespAccessData 參數在 dsmQueryAccess() 上的類型定義             |
+-------------------------------------------------------------------------*/

typedef struct
{
   dsUint16_t        stVersion ;                    /* 結構版本      */
   char              node[DSM_MAX_ID_LENGTH+1];     /* 節點名稱              */
   char              owner[DSM_MAX_OWNER_LENGTH+1]; /* 擁有者                   */
   dsmObjName      objName ;                          /* 物件名稱 */
   dsmAccessType     accessType;                    /* 保存或備份     */
   dsUint32_t        ruleNumber ;                   /* 存取權規則 ID         */
}qryRespAccessData;

#define qryRespAccessDataVersion  1

/*-------------------------------------------------------------------------+
|  envSetUp 參數在 dsmSetUp() 上的類型定義
+-------------------------------------------------------------------------*/
typedef struct S_envSetUp
{
   dsUint16_t     stVersion;   /* 結構版本  */
   char            dsmiDir[DSM_PATH_MAX + DSM_NAME_MAX +1];
   char            dsmiConfig[DSM_PATH_MAX + DSM_NAME_MAX +1];
   char            dsmiLog[DSM_PATH_MAX + DSM_NAME_MAX +1];
   char            **argv;  /* 適用於可執行檔名稱 argv[0] */
   char               logName[DSM_NAME_MAX +1];
   dsmBool_t       reserved1;                  /* 供未來使用 */
   dsmBool_t       reserved2;                  /* 供未來使用 */
}envSetUp;

#define envSetUpVersion  4

/*-------------------------------------------------------------------------+
|  適用於 dsmInitExIn_t 的類型定義
+-------------------------------------------------------------------------*/
typedef struct dsmInitExIn_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsmApiVersionEx    *apiVersionExP;
   char               *clientNodeNameP;
   char               *clientOwnerNameP;
   char               *clientPasswordP;
   char               *userNameP;
   char               *userPasswordP;
   char               *applicationTypeP;
   char               *configfile;
```

```
   char               *options;
   char               dirDelimiter;
   dsmBool_t          useUnicode;
   dsmBool_t          bCrossPlatform;
   dsmBool_t          bService;
   dsmBool_t          bEncryptKeyEnabled;
   char               *encryptionPasswordP;
   dsmBool_t          useTsmBuffers;
   dsUint8_t          numTsmBuffers;
   dsmAppVersion      *appVersionP;
}dsmInitExIn_t;

#define dsmInitExInVersion 5

/*-------------------------------------------------------------------------+
|  適用於 dsmInitExOut_t 的類型定義
+-------------------------------------------------------------------------*/
typedef struct dsmInitExOut_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsInt16_t          userNameAuthorities;
   dsInt16_t          infoRC;         /* 若發現的話，會有錯誤回覆碼 */
   char               adsmServerName[DSM_MAX_SERVERNAME_LENGTH+1];
   dsUint16_t     serverVer;              /* 伺服器的版本號碼          */
   dsUint16_t         serverRel;     /* 伺服器的版本號碼          */
   dsUint16_t         serverLev;     /* 伺服器的層級號碼            */
   dsUint16_t         serverSubLev;  /* 伺服器的次層級號碼         */

   dsmBool_t          bIsFailOverMode; /* 如果發生了失效接手則為 true */
   char               replServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* 抄寫伺服器名稱 */
   char               homeServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* 起始伺服器名稱 */

}dsmInitExOut_t;

#define dsmInitExOutVersion 3

/*-------------------------------------------------------------------------+
| 使用的 LogType 在 logInfo 上的類型定義                                   |
+-------------------------------------------------------------------------*/
typedef enum
{
   logSevInfo = 0x00,       /* 資訊 ANE4991 */
   logSevWarning,           /* 警告     ANE4992 */
   logSevError,             /* 錯誤       ANE4993 */
   logSevSevere,            /* 嚴重      ANE4994 */
   logSevLicense,           /* 授權     ANE4995 */
   logSevTryBuy             /* 嘗試「購買」      ANE4996 */
}dsmLogSeverity ;


/*-------------------------------------------------------------------------+
|  適用於 dsmLogExIn_t 的類型定義
+-------------------------------------------------------------------------*/
typedef struct dsmLogExIn_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsmLogSeverity     severity;
   char               appMsgID[8];
   dsmLogType         logType;    /* 日誌類型：本端，伺服器或兩者 */
   char               *message;   /* 記載的訊息文字 */
   char               appName[DSM_MAX_PLATFORM_LENGTH];
   char               osPlatform[DSM_MAX_PLATFORM_LENGTH];
   char               appVersion[DSM_MAX_PLATFORM_LENGTH];
}dsmLogExIn_t;

#define dsmLogExInVersion 2

/*-------------------------------------------------------------------------+
|  適用於 dsmlogExOut_t 的類型定義
+-------------------------------------------------------------------------*/
typedef struct dsmLogExOut_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
}dsmLogExOut_t;

#define dsmLogExOutVersion 1

/*-------------------------------------------------------------------------+
|  適用於 dsmRenameIn_t 的類型定義
```

```
+------------------------------------------------------------------------*/
typedef struct dsmRenameIn_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsUint32_t     dsmHandle;     /* 要處理的階段作業 */
   dsUint8_t        repository;                    /* 備份或保存  */
   dsmObjName       *objNameP ;                    /* 物件名稱 */
   char             newHl[DSM_MAX_HL_LENGTH + 1];  /* 新的高階名稱 */
   char             newLl[DSM_MAX_LL_LENGTH + 1];  /* 新的低階名稱 */
   dsmBool_t        合併;                        /* 合併至現存名稱 */
   ObjID            objId;                         /* 保存的 objId */
}dsmRenameIn_t;

#define dsmRenameInVersion 1

/*------------------------------------------------------------------------+
|  適用於 dsmRenameOut_t 的類型定義
+------------------------------------------------------------------------*/
typedef struct dsmRenameOut_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
}dsmRenameOut_t;

#define dsmRenameOutVersion 1

/*------------------------------------------------------------------------+
|  適用於 dsmEndSendObjExIn_t 的類型定義
+------------------------------------------------------------------------*/
typedef struct dsmEndSendObjExIn_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsUint32_t     dsmHandle;     /* 要處理的階段作業 */
}dsmEndSendObjExIn_t;

#define dsmEndSendObjExInVersion 1

/*------------------------------------------------------------------------+
|  適用於 dsmEndSendObjExOut_t 的類型定義
+------------------------------------------------------------------------*/
typedef struct dsmEndSendObjExOut_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsStruct64_t       totalBytesSent;      /* 從 app 讀取總位元組數 */
   dsmBool_t          objCompressed;       /* 物件是否已壓縮            */
   dsStruct64_t       totalCompressSize;   /* 壓縮之後的大小總計 */
   dsStruct64_t       totalLFBytesSent;    /* 不需 LAN 的總送出位元組數 */
   dsUint8_t          encryptionType;      /* 使用的加密類型   */
   dsmBool_t          objDeduplicated;     /* 物件是否已經過刪除重複資料處理 */
   dsStruct64_t       totalDedupSize;      /* 刪除重複資料之後的大小總計 */
}dsmEndSendObjExOut_t;

#define dsmEndSendObjExOutVersion 3
/*------------------------------------------------------------------------+
|  適用於 dsmGroupHandlerIn_t 的類型定義
+------------------------------------------------------------------------*/
typedef struct dsmGroupHandlerIn_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsUint32_t     dsmHandle;     /* 要處理的階段作業 */
   dsUint8_t       groupType;        /* 群組類型                              */
   dsUint8_t        actionType;       /* 群組作業類型              */
   dsUint8_t        memberType;       /* 成員類型：主導器或成員    */
   dsStruct64_t    leaderObjId;      /*  當操作成員時的群組前導的 OBJID */
   char            *uniqueGroupTagP; /* 唯一的群組 ID                    */
   dsmObjName      *objNameP ;       /* 群組前導物件名稱 */
   dsmGetList       memberObjList;    /* 移除及指定的物件清單   */
}dsmGroupHandlerIn_t;

#define dsmGroupHandlerInVersion 1

/*------------------------------------------------------------------------+
|  適用於 dsmGroupHandlerExOut_t 的類型定義
+------------------------------------------------------------------------*/
typedef struct dsmGroupHandlerOut_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
}dsmGroupHandlerOut_t;
```

```
#define dsmGroupHandlerOutVersion 1

/*--------------------------------------------------------------------------+
|  適用於 dsmEndTxnExIn_t 的類型定義
+--------------------------------------------------------------------------*/
typedef struct dsmEndTxnExIn_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsUint32_t     dsmHandle;     /* 要處理的階段作業 */
   dsUint8_t        vote;
}dsmEndTxnExIn_t;

#define dsmEndTxnExInVersion 1

/*--------------------------------------------------------------------------+
|  適用於 dsmEndTxnExOut_t 的類型定義
+--------------------------------------------------------------------------*/
typedef struct dsmEndTxnExOut_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsUint16_t         原因；                 /* 原因碼                      */
   dsStruct64_t       groupLeaderObjId;        /* groupLeader 物件 ID 返回 */
                                               /* DSM_ACTION_OPEN               */
   dsUint8_t          reserved1;               /* 供未來使用                  */
   dsUint16_t         reserved2;               /* 供未來使用                  */
}dsmEndTxnExOut_t;

#define dsmEndTxnExOutVersion 1

/*--------------------------------------------------------------------------+
|  適用於 dsmEndGetDataExIn_t 的類型定義
+--------------------------------------------------------------------------*/
typedef struct dsmEndGetDataExIn_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsUint32_t     dsmHandle;     /* 要處理的階段作業 */
}dsmEndGetDataExIn_t;

#define dsmEndGetDataExInVersion 1

/*--------------------------------------------------------------------------+
|  適用於 dsmEndGetDataExOut_t 的類型定義
+--------------------------------------------------------------------------*/
typedef struct dsmEndGetDataExOut_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsUint16_t     reason;            /* 原因碼                    */
   dsStruct64_t   totalLFBytesRecv;  /* 已接收的不需 LAN 位元組總數 */
}dsmEndGetDataExOut_t;

#define dsmEndGetDataExOutVersion 1

/*--------------------------------------------------------------------------+
| 物件清單在 dsmRetentionEvent() 上的類型定義                               |
+--------------------------------------------------------------------------*/
typedef struct dsmObjList
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsUint32_t       numObjId ;       /* 清單內的物件 ID 數*/
   ObjID            *objId;            /* 物件 ID 的清單至信號     */
}dsmObjList_t ;

#define dsmObjlistVersion 1

/*--------------------------------------------------------------------------+
| 使用事件類型在 dsmRetentionEvent 上的類型定義                             |
+--------------------------------------------------------------------------*/
typedef enum
{
   eventRetentionActivate = 0x00,   /* 發信號給發生事件的伺服器 */
   eventHoldObj,                    /* 暫停刪除 / 物件截止          */
   eventReleaseObj                  /* 回復一般刪除 / 截止處理       */
}dsmEventType_t;

/*--------------------------------------------------------------------------+
|  在 dsmRetentionEvent() 上的類型定義                                      |
+--------------------------------------------------------------------------*/
typedef struct dsmRetentionEventIn_t
{
```

```
   dsUint16_t     stVersion;   /* 結構版本  */
   dsUint32_t         dsmHandle;             /* 階段作業 Handle     */
   dsmEventType_t     eventType;                   /* 事件類型            */
   dsmObjList_t       objList;                     /* 物件 ID             */
}dsmRetentionEventIn_t;

#define dsmRetentionEventInVersion 1

/*--------------------------------------------------------------------------+
|  在 dsmRetentionEvent() 上的類型定義                                      |
+--------------------------------------------------------------------------*/
typedef struct dsmRetentionEventOut_t
{
   dsUint16_t        stVersion ;                  /* 結構版本      */
}dsmRetentionEventOut_t;

#define dsmRetentionEventOutVersion 1


/*--------------------------------------------------------------------------+
|  在 dsmRequestBuffer() 上的類型定義                                       |
+--------------------------------------------------------------------------*/
typedef struct requestBufferIn_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsUint32_t         dsmHandle;             /* 階段作業 Handle     */
}requestBufferIn_t;

#define requestBufferInVersion 1

/*--------------------------------------------------------------------------+
|  在 dsmRequestBuffer() 上的類型定義                                       |
+--------------------------------------------------------------------------*/
typedef struct requestBufferOut_t
{
   dsUint16_t        stVersion ;                  /* 結構版本      */
   dsUint8_t          tsmBufferHandle;       /* 處理至 tsm 資料緩衝區 */
   char               *dataPtr;                    /* 寫入資料的位址 */
   dsUint32_t       bufferLen;                 /* 寫入的最大資料長度 */
}requestBufferOut_t;

#define requestBufferOutVersion 1

/*--------------------------------------------------------------------------+
|  在 dsmReleaseBuffer() 上的類型定義                                       |
+--------------------------------------------------------------------------*/
typedef struct releaseBufferIn_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsUint32_t         dsmHandle;             /* 階段作業 Handle     */
   dsUint8_t          tsmBufferHandle;       /* 處理至 tsm 資料緩衝區 */
   char               *dataPtr;                    /* 寫入資料的位址 */
}releaseBufferIn_t;

#define releaseBufferInVersion 1

/*--------------------------------------------------------------------------+
|  在 dsmReleaseBuffer() 上的類型定義                                       |
+--------------------------------------------------------------------------*/
typedef struct releaseBufferOut_t
{
   dsUint16_t        stVersion ;                  /* 結構版本      */
}releaseBufferOut_t;

#define releaseBufferOutVersion 1

/*--------------------------------------------------------------------------+
|  在 dsmGetBufferData() 上的類型定義                                       |
+--------------------------------------------------------------------------*/
typedef struct getBufferDataIn_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsUint32_t         dsmHandle;             /* 階段作業 Handle     */
}getBufferDataIn_t;

#define getBufferDataInVersion 1

/*--------------------------------------------------------------------------+
|  在 dsmGetBufferData() 上的類型定義                                       |
```

```
+------------------------------------------------------------------------*/
typedef struct getBufferDataOut_t
{
   dsUint16_t        stVersion ;                    /* 結構版本       */
   dsUint8_t          tsmBufferHandle;       /* 處理至 tsm 資料緩衝區 */
   char              *dataPtr;              /* 讀取實際資料的位址 */
   dsUint32_t       numBytes;              /* 從 dataPtr 實際位元組數讀取 */
}getBufferDataOut_t;

#define getBufferDataOutVersion 1

/*-------------------------------------------------------------------------+
|  在 dsmSendBufferData() 上的類型定義                                     |
+-------------------------------------------------------------------------*/
typedef struct sendBufferDataIn_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsUint32_t         dsmHandle;             /* 階段作業 Handle     */
   dsUint8_t          tsmBufferHandle;       /* 處理至 tsm 資料緩衝區 */
   char               *dataPtr;              /* 傳送實際資料的位址 */
   dsUint32_t         numBytes;              /* 從 dataPtr 實際位元組數傳送 */
}sendBufferDataIn_t;

#define sendBufferDataInVersion 1

/*-------------------------------------------------------------------------+
|  在 dsmSendBufferData() 上的類型定義                                     |
+-------------------------------------------------------------------------*/
typedef struct sendBufferDataOut_t
{
   dsUint16_t        stVersion ;                    /* 結構版本       */
}sendBufferDataOut_t;

#define sendBufferDataOutVersion 1

/*-------------------------------------------------------------------------+
|  適用於 dsmUpdateObjExIn_t 的類型定義
+-------------------------------------------------------------------------*/
typedef struct dsmUpdateObjExIn_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsUint32_t         dsmHandle;             /* 階段作業 Handle     */
   dsmSendType     sendType;              /* 傳送類型備份/保存 */
   char            *descrP;               /* 保存說明            */
   dsmObjName      *objNameP;             /* 物件名稱            */
   ObjAttr         *objAttrPtr;           /* 屬性                */
   dsUint32_t      objUpdAct;             /* 更新動作            */
   ObjID           archObjId;             /* 保存的 objId        */
}dsmUpdateObjExIn_t;

#define dsmUpdateObjExInVersion 1

/*-------------------------------------------------------------------------+
|  適用於 dsmUpdateObjExOut_t 的類型定義
+-------------------------------------------------------------------------*/
typedef struct dsmUpdateObjExOut_t
{
    dsUint16_t     stVersion;   /* 結構版本  */
}dsmUpdateObjExOut_t;

#define dsmUpdateObjExOutVersion 1

#if (_OPSYS_TYPE == DS_WINNT) && !defined(_WIN64)
#pragma pack()
#endif


#ifdef _MAC
#pragma options align = reset
#endif
#endif /* _H_DSMAPITD */
```

tsmapitd.h

```
/*****************************************************************************
*       IBM Spectrum Protect             *
* API Client Component                                                       *
*                                                                            *
* IBM Confidential                                                           *
* (IBM Confidential-Restricted when combined with the Aggregated OCO   *
* source modules for this program)                                           *
*                                                                            *
* OCO Source Materials                                                       *
*                                                                            *
* 5648-020 (C) Copyright IBM Corporation 1993, 2018                    *
*****************************************************************************/

/*****************************************************************************
 * 標頭檔名稱：tsmapitd.h
 *
 * 環境：            ***********************************************
 *                    ** 這是一個 platform-independent 原始檔       **
 *
 *                    ***********************************************
 *
 * 設計注意事項：    這個檔案包含基本資料類型以及可由所有用戶端來源檔案
*                    併入的常數。 應該要適當地設定此檔案內的常數，*                    使它們適用於要在其上
*                    執行用戶端軟體
*                    的特定機器及作業系統。
 *
 *                    dsmapips.h 內含特定的平台定義
 *
 * 敘述名稱：IBM Spectrum Protect API 常數的定義
 *---------------------------------------------------------------------------*/

#ifndef _H_TSMAPITD
#define _H_TSMAPITD


/*=== 設定結構對齊來壓縮結構 ===*/
#if _OPSYS_TYPE == DS_WINNT
#ifdef _WIN64
#pragma pack(8)
#else
#pragma pack(1)
#endif
#endif

#ifdef _MAC
#pragma options align = packed
#endif

/*============================================================
 使用 tsm 介面的 Win32 應用程式，在編譯期間必須使用
   -DUNICODE 旗標。
 ============================================================*/
#if _OPSYS_TYPE == DS_WINNT && !defined(DSMAPILIB)
#ifndef UNICODE
#error "Win32 applications using the TSM interface MUST be compiled with the -DUNICODE flag"
#endif
#endif

/*============================================================
 使用 tsm 介面的 Mac OS X 應用程式，在編譯期間必須使用
   -DUNICODE 旗標。
 ============================================================*/
#if _OPSYS_TYPE == DS_MACOS && !defined(DSMAPILIB)
#ifndef UNICODE
#error "使用 TSM 介面的 Mac OS X 應用程式，必須使用 -DUNICODE 旗標來編譯"
#endif
#endif

/*---------------------------------------------------------------------------+
 | dsmGetType 參數在 tsmBeginGetData() 上的類型定義                          |
 +---------------------------------------------------------------------------*/
typedef enum
{
   gtTsmBackup = 0x00,                    /* 備份處理類型     */
   gtTsmArchive                           /* 保存處理類型    */
} tsmGetType ;
```

```
/*-------------------------------------------------------------------------+
 |  dsmQueryType 參數在 tsmBeginQuery() 上的類型定義                       |
 +-------------------------------------------------------------------------*/
typedef enum
{
   qtTsmArchive = 0x00,                     /* 保存查詢類型       */
   qtTsmBackup,                             /* 備份查詢類型         */
   qtTsmBackupActive,                       /* 作用中備份檔的快速查詢 */
   qtTsmFilespace,                          /* 檔案空間查詢類型     */
   qtTsmMC,                                 /* 管理類別查詢類型    */
    qtTsmReserved1,                          /* 可供未來使用                */
    qtTsmReserved2,                          /* 可供未來使用                */
    qtTsmReserved3,                          /* 可供未來使用                */
    qtTsmReserved4,                          /* 可供未來使用                */
   qtTsmBackupGroups,                       /* 在特定的檔案空間中的所有群組前導 */
   qtTsmOpenGroups,                         /* 所有與主導器相關的群組成員 */
    qtTsmReserved5,                          /* 可供未來使用                */
   qtTsmProxyNodeAuth,                      /* 可供這個節點 proxy 至的節點   */
    qtTsmProxyNodePeer,                      /* 此目標節點下的對等節點  */
    qtTsmReserved6,                          /* 未來使用                  */
    qtTsmReserved7,                          /* 未來使用                  */
    qtTsmReserved8                           /* 未來使用                  */
} tsmQueryType ;

/*-------------------------------------------------------------------------+
 |  sendType 參數在 tsmBindMC() 及 tsmSendObj() 上的類型定義               |
 +-------------------------------------------------------------------------*/
typedef enum
{
   stTsmBackup = 0x00,                        /* 備份處理類型     */
   stTsmArchive,                              /* 保存處理類型    */
   stTsmBackupMountWait,           /* 以 mountwait 上處理的備份    */
   stTsmArchiveMountWait           /* 以 mountwait 上處理的保存   */
} tsmSendType ;

/*-------------------------------------------------------------------------+
 |  delType 參數在 tsmDeleteObj() 上的類型定義                             |
 +-------------------------------------------------------------------------*/
typedef enum
{
   dtTsmArchive = 0x00,                              /* 保存刪除類型 */
   dtTsmBackup,                           /* 備份刪除 (關閉) 類型 */
   dtTsmBackupID                         /* 備份刪除 (移除) 類型 */
} tsmDelType ;

/*-------------------------------------------------------------------------+
 |  sendType 參數在 tsmSetAccess() 上的類型定義                            |
 +-------------------------------------------------------------------------*/
typedef enum
{
   atTsmBackup = 0x00,                         /* 備份處理類型    */
   atTsmArchive                               /* 保存處理類型    */
}tsmAccessType;

/*-------------------------------------------------------------------------+
 |  改寫參數在 tsmSendObj() 上的類型定義                                   |
 +-------------------------------------------------------------------------*/
typedef enum
{
   owIGNORE = 0x00,
   owYES,
   owNO
}tsmOwType;


/*-------------------------------------------------------------------------+
 |  API 版本在 tsmInit() 及 tsmQueryApiVersion() 上的類型定義              |
 +-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t     stVersion;           /* 結構版本                */
   dsUint16_t version;       /* API 版本                      */
   dsUint16_t release;       /* API 版次                      */
   dsUint16_t level;         /* API 層級                        */
   dsUint16_t subLevel;      /* API 子層                     */
   dsmBool_t  unicode;       /* API Unicode?                    */
} tsmApiVersionEx;
```

```
#define tsmApiVersionExVer       2

/*-------------------------------------------------------------------------+
|  應用程式版本在 tsmInit() 上的類型定義                                    |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t     stVersion;           /* 結構版本                */
   dsUint16_t   applicationVersion;   /* 應用程式版本號碼    */
   dsUint16_t   applicationRelease;   /* 應用程式版次號碼    */
   dsUint16_t   applicationLevel;     /* 應用程式層次號碼     */
   dsUint16_t   applicationSubLevel;  /* 應用程式子層次號碼 */
} tsmAppVersion;

#define tsmAppVersionVer      1


/*-------------------------------------------------------------------------+
 |  在「BindMC、傳送、刪除及查詢」上所使用物件名稱的類型定義               |
 +-------------------------------------------------------------------------*/

typedef struct tsmObjName
{
   dsChar_t   fs[DSM_MAX_FSNAME_LENGTH + 1] ;              /* 檔案空間名稱 */
   dsChar_t   hl[DSM_MAX_HL_LENGTH + 1] ;                 /* 高階名稱 */
   dsChar_t   ll[DSM_MAX_LL_LENGTH + 1] ;                  /* 低階名稱 */
   dsUint8_t  objType;          /* 如需物件類型值，請參閱上述的定義 */
   dsChar_t           dirDelimiter;
} tsmObjName;

/*-------------------------------------------------------------------------+
 |  「備份」刪除資訊在 dsmDeleteObj() 上的類型定義                          |
 +-------------------------------------------------------------------------*/
typedef struct tsmDelBack
{
   dsUint16_t        stVersion ;                    /* 結構版本      */
   tsmObjName       *objNameP ;                    /* 物件名稱 */
   dsUint32_t       copyGroup ;                    /* 副本群組             */
} tsmDelBack ;

#define  tsmDelBackVersion    1

/*-------------------------------------------------------------------------+
 |  「保存」刪除資訊在 dsmDeleteObj() 上的類型定義                          |
 +-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t        stVersion ;                    /* 結構版本      */
   dsStruct64_t       objId ;                        /* 物件 ID         */
} tsmDelArch ;

#define  tsmDelArchVersion  1

/*-------------------------------------------------------------------------+
 |  「備份 ID」刪除資訊在 dsmDeleteObj() 上的類型定義                       |
 +-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t        stVersion ;                    /* 結構版本      */
   dsStruct64_t       objId ;                        /* 物件 ID         */
} tsmDelBackID;

#define  tsmDelBackIDVersion  1


/*-------------------------------------------------------------------------+
 |  刪除資訊在 dsmDeleteObj() 上的類型定義                                  |
 +-------------------------------------------------------------------------*/
typedef union
{
   tsmDelBack   backInfo ;
   tsmDelArch   archInfo ;
   tsmDelBackID backIDInfo;
} tsmDelInfo ;

/*-------------------------------------------------------------------------+
 |  「物件屬性」參數在 dsmSendObj() 上的類型定義                            |
 +-------------------------------------------------------------------------*/
typedef struct tsmObjAttr
```

```
{
   dsUint16_t     stVersion;           /* 結構版本                */
   dsChar_t     owner[DSM_MAX_OWNER_LENGTH + 1];      /* 物件擁有者          */
   dsStruct64_t sizeEstimate;             /* 物件的大小預估 (以位元組為單位) */
   dsmBool_t    objCompressed;                 /* 物件是否已壓縮?     */
   dsUint16_t   objInfoLength;            /* object-dependent 資訊長度   */
   char         *objInfo;                 /* object-dependent 資訊位元組緩衝區 */
   dsChar_t     *mcNameP;                 /* 置換的 mgmnt 類別名稱    */
   tsmOwType    reserved1;                /* 供未來使用                  */
   tsmOwType    reserved2;                /* 供未來使用                  */
   dsmBool_t    disableDeduplication;         /* 不對此物件強制執行刪除重複資料    */
   dsmBool_t    useExtObjInfo;             /* 使用延伸物件資訊,最多 1536        */
} tsmObjAttr;

#define tsmObjAttrVersion 5

/*-------------------------------------------------------------------------+
 | mcBindKey 傳回在 dsmBindMC() 上的類型定義                               |
 +-------------------------------------------------------------------------*/
typedef struct tsmMcBindKey
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsChar_t    mcName[DSM_MAX_MC_NAME_LENGTH + 1];
   /* 連結至物件的 MC 名稱。 */
   dsmBool_t   backup_cg_exists;                          /* 是/否 */
   dsmBool_t   archive_cg_exists;                         /* 是/否 */
   dsChar_t    backup_copy_dest[DSM_MAX_CG_DEST_LENGTH + 1];
   /* 備份副本目的地名稱 */
   dsChar_t    archive_copy_dest[DSM_MAX_CG_DEST_LENGTH + 1];
   /* 保存副本 dest.name */
} tsmMcBindKey;

#define tsmMcBindKeyVersion  1


/*-------------------------------------------------------------------------+
 |  Mgmt 類別 queryBuffer 在 dsmBeginQuery() 上的類型定義                  |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryMCData
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsChar_t     *mcName;                        /* Mgmt 類別名稱 */
   /* 使用單一名稱來取得一個,或使用空字串來產生全部*/
   dsmBool_t    mcDetail;                           /* 是否需要詳細資訊? */
} tsmQryMCData;

#define tsmQryMCDataVersion  1


/*-------------------------------------------------------------------------+
 |  「保存副本群組」詳細資訊在查詢 MC 回應上的類型定義                     |
 +-------------------------------------------------------------------------*/
typedef struct tsmArchDetailCG
{
   dsChar_t     cgName[DSM_MAX_CG_NAME_LENGTH + 1];       /* 副本群組名稱 */
   dsUint16_t   frequency;                        /* 複製 (保存) 頻率 */
   dsUint16_t   retainVers;                          /* 保留版本 */
   dsUint8_t    copySer;       /* 如需複製序列化值,請參閱定義 */
   dsUint8_t    copyMode;         /* 如需複製模式值,請參閱上述的定義 */
   dsChar_t     destName[DSM_MAX_CG_DEST_LENGTH + 1];      /* 複製 dest 名稱 */
   dsmBool_t    bLanFreeDest;           /* 目的地是否有不需 LAN 路徑?    */
   dsmBool_t    reserved;                          /* 目前未使用 */
   dsUint8_t    retainInit;             /* 如需可能的值,請參閱 dsmapitd.h  */
   dsUint16_t   retainMin;              /* 若 retInit 是事件日期數 */
   dsmBool_t    bDeduplicate;             /* 目的地已啟用刪除重複資料   */
}tsmArchDetailCG;

/*-------------------------------------------------------------------------+
 |  「備份副本群組」詳細資訊在查詢 MC 回應上的類型定義                     |
 +-------------------------------------------------------------------------*/
typedef struct tsmBackupDetailCG
{
   dsChar_t     cgName[DSM_MAX_CG_NAME_LENGTH + 1];       /* 副本群組名稱 */
   dsUint16_t   frequency;                           /* 備份頻率 */
   dsUint16_t   verDataExst;                        /* 現存資料的版本數 */
   dsUint16_t   verDataDltd;                       /* 刪除資料的版本數 */
   dsUint16_t   retXtraVers;                       /* 保留額外版本 */
   dsUint16_t   retOnlyVers;                        /* 保留唯一版本 */
```

```
   dsUint8_t    copySer;       /* 如需複製序列化值，請參閱定義 */
   dsUint8_t    copyMode;         /* 如需複製模式值，請參閱上述的定義 */
   dsChar_t     destName[DSM_MAX_CG_DEST_LENGTH + 1];      /* 複製 dest 名稱 */
   dsmBool_t    bLanFreeDest;           /* 目的地是否有不需 LAN 路徑？    */
   dsmBool_t    reserved;                              /* 目前未使用 */
   dsmBool_t    bDeduplicate;             /* 目的地已啟用刪除重複資料   */
}tsmBackupDetailCG;


/*-------------------------------------------------------------------------+
 |  「查詢 Mgmt 類別」詳細資訊回應在 dsmGetNextQObj()| 上的類型定義          |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryRespMCDetailData
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsChar_t        mcName[DSM_MAX_MC_NAME_LENGTH + 1];            /* mc 名稱 */
   dsChar_t        mcDesc[DSM_MAX_MC_DESCR_LENGTH + 1]; /*mc 說明 */
   archDetailCG    archDet;                 /* 保存副本群組詳細資訊 */
   backupDetailCG  backupDet;                /* 備份副本群組詳細資訊 */
} tsmQryRespMCDetailData;

#define tsmQryRespMCDetailDataVersion  4

/*-------------------------------------------------------------------------+
 | 「查詢 Mgmt 類別」彙總回應在 dsmGetNextQObj()| 上的類型定義               |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryRespMCData
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsChar_t        mcName[DSM_MAX_MC_NAME_LENGTH + 1];            /* mc 名稱 */
   dsChar_t     mcDesc[DSM_MAX_MC_DESCR_LENGTH + 1];       /* mc 說明 */
}tsmQryRespMCData;

#define tsmQryRespMCDataVersion  1

/*-------------------------------------------------------------------------+
 |  保存 queryBuffer 在 tsmBeginQuery() 上的類型定義                       |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryArchiveData
{
   dsUint16_t     stVersion;   /* 結構版本  */
   tsmObjName   *objName;                  /* 完整 dsm 物件名稱 */
   dsChar_t    *擁有者；            /* 擁有者名稱 */
   /* 如需日期界限最大值，請參閱上述的定義 */
   dsmDate      insDateLowerBound;      /* 低界限保存插入日期 */
   dsmDate      insDateUpperBound;       /* 高界限保存插入日期 */
   dsmDate      expDateLowerBound;          /* 低界限到期日 */
   dsmDate      expDateUpperBound;           /* 高界限到期日 */
   dsChar_t     *descr;                    /* 保存說明 */
} tsmQryArchiveData;

#define tsmQryArchiveDataVersion  1

/*-------------------------------------------------------------------------+
| 「查詢保存」回應在 dsmGetNextQObj() 上的類型定義                         |
+-------------------------------------------------------------------------*/
typedef struct tsmQryRespArchiveData
{
    dsUint16_t     stVersion;   /* 結構版本  */
    tsmObjName      objName;                       /* 檔案空間名稱限定元 */
    dsUint32_t      copyGroup;                          /* 副本群組號碼 */
    dsChar_t        mcName[DSM_MAX_MC_NAME_LENGTH + 1];            /* mc 名稱 */
    dsChar_t    owner[DSM_MAX_OWNER_LENGTH + 1];             /* 擁有者名稱 */
    dsStruct64_t    objId;                                   /* 唯一的副本 ID */
    dsStruct64_t    reserved;                          /* 舊版相容性 */
    dsUint8_t       mediaClass;                            /* 媒體存取權類別 */
    dsmDate         insDate;                         /* 保存插入日期 */
    dsmDate         expDate;                       /* 物件到期日 */
    dsChar_t        descr[DSM_MAX_DESCR_LENGTH + 1];   /* 保存說明 */
    dsUint16_t      objInfolen;             /* object-dependent 資訊長度*/
    dsUint8_t       reservedObjInfo[DSM_MAX_OBJINFO_LENGTH];  /*相依於物件的資訊 */
    dsUint160_t     restoreOrderExt;                           /* 還原次序 */
    dsStruct64_t    sizeEstimate;              /* 使用者儲存預估大小 */
    dsUint8_t       compressType;                          /* 壓縮旗標 */
    dsUint8_t       retentionInitiated;   /* 物件在保留事件上等待*/
    dsUint8_t       objHeld; /* 物件在「保留」上，如需值請參閱 dsmapitd.h */
    dsUint8_t       encryptionType;          /* 加密類型 */
```

```
    dsmBool_t       clientDeduplicated;      /* API 對物件刪除重複資料*/
    dsUint8_t       objInfo[DSM_MAX_EXT_OBJINFO_LENGTH];  /*相依於物件的資訊 */
    dsChar_t        compressAlg[DSM_MAX_COMPRESSTYPE_LENGTH + 1];  /* 壓縮演算法名稱 */
} tsmQryRespArchiveData;

#define tsmQryRespArchiveDataVersion  7

/*-------------------------------------------------------------------------+
 |  保存 sendBuff 參數在 dsmSendObj() 上的類型定義                          |
 +-------------------------------------------------------------------------*/
typedef struct tsmSndArchiveData
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsChar_t     *descr;                    /* 保存說明 */
} tsmSndArchiveData;

#define tsmSndArchiveDataVersion  1

/*-------------------------------------------------------------------------+
 |  備份 queryBuffer 在 dsmBeginQuery() 上的類型定義                        |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryBackupData
{
   dsUint16_t     stVersion;   /* 結構版本  */
   tsmObjName  *objName;          /* 完整 dsm 物件名稱 */
   dsChar_t    *owner;            /* 擁有者名稱 */
   dsUint8_t   objState;          /* 物件狀態選擇器 */
   dsmDate     pitDate;           /* 時間點還原的日期值 */
   /* 如需可能的值，請參閱上述的定義 */
    dsUint32_t  reserved1;
    dsUint32_t  reserved2;
} tsmQryBackupData;

#define tsmQryBackupDataVersion  3

/*-------------------------------------------------------------------------+
 |「查詢備份」回應在 dsmGetNextQObj() 上的類型定義                          |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryRespBackupData
{
   dsUint16_t     stVersion;   /* 結構版本  */
   tsmObjName  objName;                      /* 完整 dsm 物件名稱 */
   dsUint32_t      copyGroup;                    /* 副本群組號碼 */
   dsChar_t        mcName[DSM_MAX_MC_NAME_LENGTH + 1];           /* mc 名稱 */
   dsChar_t    owner[DSM_MAX_OWNER_LENGTH + 1];           /* 擁有者名稱 */
   dsStruct64_t    objId;                                /* 唯一的物件 ID */
   dsStruct64_t    reserved;                        /* 舊版相容性 */
   dsUint8_t       mediaClass;                          /* 媒體存取權類別 */
   dsUint8_t       objState;                       /* 物件狀態，作用中等等。 */
   dsmDate         insDate;                         /* 備份插入日期 */
   dsmDate         expDate;                     /* 物件到期日 */
   dsUint16_t      objInfolen;             /* object-dependent 資訊長度*/
   dsUint8_t       reservedObjInfo[DSM_MAX_OBJINFO_LENGTH];  /*相依於物件的資訊 */
   dsUint160_t     restoreOrderExt;                          /* 還原次序 */
   dsStruct64_t    sizeEstimate;             /* 使用者儲存的預估大小 */
   dsStruct64_t    baseObjId;
   dsUint16_t      baseObjInfolen;             /* 基本 object-dependent 資訊長度*/
   dsUint8_t       baseObjInfo[DSM_MAX_OBJINFO_LENGTH];  /* 基本 object-dependent 資訊 */
   dsUint160_t     baseRestoreOrder;                          /* 還原次序 */
   dsUint32_t     fsID;
   dsUint8_t       compressType;
   dsmBool_t       isGroupLeader;
   dsmBool_t       isOpenGroup;
   dsUint8_t       reserved1;               /* 供未來使用 */
   dsmBool_t       reserved2;                  /* 供未來使用 */
   dsUint16_t      reserved3;               /* 供未來使用 */
   reservedInfo_t  *reserved4;              /* 供未來使用 */
   dsUint8_t       encryptionType;          /* 加密類型 */
   dsmBool_t       clientDeduplicated;      /* API 對物件刪除重複資料*/
   dsUint8_t       objInfo[DSM_MAX_EXT_OBJINFO_LENGTH];  /*相依於物件的資訊 */
   dsChar_t        compressAlg[DSM_MAX_COMPRESSTYPE_LENGTH + 1];  /* 壓縮演算法名稱 */
} tsmQryRespBackupData;

#define tsmQryRespBackupDataVersion  8
/*-------------------------------------------------------------------------+
 |  作用中備份 queryBuffer 在 dsmBeginQuery() 上的類型定義
 |
 |  附註：  如需作用中備份查詢，僅需要設定 objName 的檔案空間及 objType 欄位。
```

```
 |          objType 僅能設定為
 |          DSM_OBJ_FILE 或 DSM_OBJ_DIRECTORY。  DSM_OBJ_ANY_TYPE 將無法
 |          在查詢上尋找符合項目。
 +----------------------------------------------------------------------------*/
typedef struct tsmQryABackupData
{
   dsUint16_t     stVersion;   /* 結構版本  */
   tsmObjName      *objName;                    /* 僅使用 fs 及 objtype  */
} tsmQryABackupData;

#define tsmQryABackupDataVersion  1

/*----------------------------------------------------------------------------+
 | 「查詢作用中備份」回應在 dsmGetNextQObj() 上的類型定義                    |
 +----------------------------------------------------------------------------*/
typedef struct tsmQryARespBackupData
{
   dsUint16_t     stVersion;   /* 結構版本  */
   tsmObjName  objName;                       /* 完整 dsm 物件名稱 */
   dsUint32_t      copyGroup;                          /* 副本群組號碼 */
   dsChar_t    mcName[DSM_MAX_MC_NAME_LENGTH + 1];/*管理類別名稱*/
   dsChar_t    owner[DSM_MAX_OWNER_LENGTH + 1];            /* 擁有者名稱 */
   dsmDate         insDate;                          /* 備份插入日期 */
   dsUint16_t  objInfolen;             /* object-dependent 資訊長度*/
   dsUint8_t   reservedObjInfo[DSM_MAX_OBJINFO_LENGTH];  /*相依於物件的資訊 */
   dsUint8_t  objInfo[DSM_MAX_EXT_OBJINFO_LENGTH];  /*相依於物件的資訊 */
} tsmQryARespBackupData;

#define tsmQryARespBackupDataVersion  2

/*----------------------------------------------------------------------------+
 |  備份 queryBuffer 在 dsmBeginQuery() 上的類型定義                          |
 +----------------------------------------------------------------------------*/
typedef struct tsmQryBackupGroups
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsUint8_t    groupType;
   dsChar_t     *fsName;
   dsChar_t     *owner;
   dsStruct64_t groupLeaderObjId;
   dsUint8_t    objType;
   dsUint32_t   reserved1;
   dsUint32_t   reserverd2;
   dsmBool_t    noRestoreOrder;
   dsmBool_t    noGroupInfo;
   dsChar_t     *hl;
} tsmQryBackupGroups;

#define tsmQryBackupGroupsVersion 4

/*----------------------------------------------------------------------------+
 |  proxy 節點 queryBuffer 在 tsmBeginQuery() 上的類型定義                    |
 +----------------------------------------------------------------------------*/
typedef struct tsmQryProxyNodeData
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsChar_t    *targetNodeName;             /* 目標節點名稱      */
}tsmQryProxyNodeData;

#define tsmQryProxyNodeDataVersion  1

/*----------------------------------------------------------------------------+
 | 使用的 qryRespProxyNodeData 參數在 tsmGetNextQObj()| 上的類型定義        |
 +----------------------------------------------------------------------------*/

typedef struct tsmQryRespProxyNodeData
{
   dsUint16_t        stVersion ;                    /* 結構版本      */
   dsChar_t          targetNodeName[DSM_MAX_ID_LENGTH+1];  /* 目標節點名稱  */
   dsChar_t          peerNodeName[DSM_MAX_ID_LENGTH+1];    /* 對等節點名稱    */
   dsChar_t          hlAddress[DSM_MAX_ID_LENGTH+1];       /* 同層級 hlAddress    */
   dsChar_t          llAddress[DSM_MAX_ID_LENGTH+1];       /* 同層級 llAddress    */
}tsmQryRespProxyNodeData;

#define tsmQryRespProxyNodeDataVersion  1

/*----------------------------------------------------------------------------+
 |  WINNT 及 OS/2 檔案空間屬性的類型定義                                      |
 +----------------------------------------------------------------------------*/
```

```
typedef struct tsmDosFSAttrib
{
   osChar_t     driveLetter ;          /* 適用於檔案空間的磁碟機代號    */
   dsUint16_t   fsInfoLength;          /* 使用的 fsInfo 長度           */
   osChar_t     fsInfo[DSM_MAX_FSINFO_LENGTH];/*caller-determined 資料  */
} tsmDosFSAttrib ;

/*-------------------------------------------------------------------------+
 |  UNIX 檔案空間屬性的類型定義                                             |
 +-------------------------------------------------------------------------*/
typedef struct tsmUnixFSAttrib
{
   dsUint16_t   fsInfoLength;          /* 使用的 fsInfo 長度            */
   osChar_t     fsInfo[DSM_MAX_FSINFO_LENGTH];/*caller-determined 資料  */
} tsmUnixFSAttrib ;

/*-------------------------------------------------------------------------+
 |   「NetWare 檔案空間」屬性的類型定義                                    |
 +-------------------------------------------------------------------------*/
typedef tsmUnixFSAttrib tsmNetwareFSAttrib;

/*-------------------------------------------------------------------------+
 |   「檔案空間」屬性在所有「檔案空間」呼叫上的類型定義                    |
 +-------------------------------------------------------------------------*/
typedef union
{
   tsmNetwareFSAttrib  netwareFSAttr;
   tsmUnixFSAttrib     unixFSAttr ;
   tsmDosFSAttrib      dosFSAttr ;
} tsmFSAttr ;

/*-------------------------------------------------------------------------+
 |  fsUpd 參數在 dsmUpdateFS() 上的類型定義
 +-------------------------------------------------------------------------*/
typedef struct   tsmFSUpd
{
   dsUint16_t        stVersion ;                  /* 結構版本      */
   dsChar_t        *fsType ;              /* 檔案空間類型                 */
   dsStruct64_t    occupancy ;            /* 佔用預估              */
   dsStruct64_t    capacity  ;            /* 容量預估               */
   tsmFSAttr       fsAttr ;               /* 平台特定的屬性 */
} tsmFSUpd ;

#define tsmFSUpdVersion  1

/*-------------------------------------------------------------------------+
 |  檔案空間 queryBuffer 在 dsmBeginQuery() 上的類型定義                   |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryFSData
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsChar_t    *fsName;                  /* 檔案空間名稱 */
} tsmQryFSData;

#define tsmQryFSDataVersion  1

/*-------------------------------------------------------------------------+
 | 「查詢檔案空間」回應在 dsmGetNextQObj() 上的類型定義                    |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryRespFSData
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsChar_t       fsName[DSM_MAX_FSNAME_LENGTH + 1];     /* 檔案空間名稱 */
   dsChar_t       fsType[DSM_MAX_FSTYPE_LENGTH + 1] ;     /* 檔案空間類型 */
   dsStruct64_t    occupancy;                  /* 佔用預估 ( 以位元組為單位 )。              */
   dsStruct64_t    capacity;                    /* 容量預估 ( 以位元組為單位 )。               */
   tsmFSAttr       fsAttr ;               /* 平台特定的屬性 */
   dsmDate        backStartDate;          /* 啟動備份日期             */
   dsmDate        backCompleteDate;       /* 結束備份日期               */
    dsmDate        reserved1    ;          /* 供未來使用                */
   dsmBool_t      bIsUnicode;
   dsUint32_t     fsID;
   dsmDate        lastReplStartDate;       /* 前次開始抄寫的時間 */
   dsmDate        lastReplCmpltDate;       /* 前次完成抄寫的時間   */
                                           /*    ( 可能失敗 ,           */
                                           /*     但仍然完成了 )            */
   dsmDate        lastBackOpDateFromServer; /* 前次在伺服器上儲存用戶端  */
                                           /*     的儲存時間戳記                */
```

```
   dsmDate        lastArchOpDateFromServer; /* 前次在伺服器上儲存用戶端  */
                                            /*    的儲存時間戳記                 */
   dsmDate        lastSpMgOpDateFromServer; /* 前次在伺服器上儲存用戶端  */
                                            /*    的儲存時間戳記                 */
   dsmDate        lastBackOpDateFromLocal;  /* 前次在本地儲存用戶端  */
                                            /*    的儲存時間戳記                  */
   dsmDate        lastArchOpDateFromLocal;  /* 前次在本地儲存用戶端  */
                                            /*    的儲存時間戳記                  */
   dsmDate        lastSpMgOpDateFromLocal;  /* 前次在本地儲存用戶端  */
                                            /*    的儲存時間戳記                  */
   dsInt32_t      failOverWriteDelay;       /* 等待分鐘數，此後才容許將用戶端  */
                                            /* 儲存到此 Repl srvr，特殊代碼：  */
                                            /* NO_ACCESS(-1) 和 ACCESS_RDONLY (-2)          */
} tsmQryRespFSData;

#define tsmQryRespFSDataVersion 5

/*-------------------------------------------------------------------------+
 |  regFilespace 參數在 dsmRegisterFS() 上的類型定義
 +-------------------------------------------------------------------------*/
typedef struct tsmRegFSData
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsChar_t        *fsName;                        /* 檔案空間名稱 */
   dsChar_t        *fsType;                        /* 檔案空間類型 */
   dsStruct64_t    occupancy;                  /* 佔用預估 ( 以位元組為單位 )。 */
   dsStruct64_t    capacity;                    /* 容量預估 ( 以位元組為單位 )。 */
   tsmFSAttr       fsAttr ;                /* 平台特定的屬性 */
} tsmRegFSData;

#define tsmRegFSDataVersion  1

/*-------------------------------------------------------------------------+
 |  階段作業資訊回應在 dsmQuerySessionInfo() 上的類型定義                  |
 +-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t     stVersion;           /* 結構版本                */
   /*-----------------------------------------------------------------*/
   /*           伺服器資訊                                          */
   /*-----------------------------------------------------------------*/
   dsChar_t       serverHost[DSM_MAX_SERVERNAME_LENGTH+1];
   /* DSM 伺服器的網路主機名稱  */
   dsUint16_t     serverPort;             /* 伺服器 comm 埠在主機上          */
   dsmDate        serverDate;             /* 伺服器的日期 / 時間               */
   dsChar_t       serverType[DSM_MAX_SERVERTYPE_LENGTH+1];
   /* 伺服器的執行平台      */
   dsUint16_t     serverVer;              /* 伺服器的版本號碼           */
   dsUint16_t         serverRel;     /* 伺服器的版本號碼           */
   dsUint16_t         serverLev;     /* 伺服器的層級號碼             */
   dsUint16_t         serverSubLev;  /* 伺服器的次層級號碼          */
   /*-----------------------------------------------------------------*/
   /*           用戶端預設值                                          */
   /*-----------------------------------------------------------------*/
   dsChar_t       nodeType[DSM_MAX_PLATFORM_LENGTH+1]; /*節點/應用程式類型*/
   dsChar_t       fsdelim;                /* 檔案空間定界符號          */
   dsChar_t       hldelim;                /* 介於 highlev 和 lowlev 的定界字元  */
   dsUint8_t      compression;            /* 壓縮旗標                  */
   dsUint8_t      archDel;                /* 保存刪除許可權         */
   dsUint8_t      backDel;                /* 備份刪除許可權         */
   dsUint32_t     maxBytesPerTxn;         /* 供未來使用                      */
   dsUint16_t     maxObjPerTxn;           /* 在 txn 中可接受的最大物件 */
   /*-----------------------------------------------------------------*/
   /*           階段作業資訊                                        */
   /*-----------------------------------------------------------------*/
   dsChar_t       id[DSM_MAX_ID_LENGTH+1];    /* 登入 ID 節點名稱          */
   dsChar_t       owner[DSM_MAX_OWNER_LENGTH+1]; /* 登入擁有者              */
   /*    ( 適用於多使用者平台 )      */
   dsChar_t       confFile[DSM_PATH_MAX + DSM_NAME_MAX +1];
   /* len 是平台 dep              */
   /* appl 配置檔 dsInit 名稱  */
   dsUint8_t      opNoTrace;              /* dsInit 選項 - NOTRACE = 1      */
   /*-----------------------------------------------------------------*/
   /*           原則資料                                             */
   /*-----------------------------------------------------------------*/
   dsChar_t       domainName[DSM_MAX_DOMAIN_LENGTH+1]; /* 網域名稱         */
   dsChar_t       policySetName[DSM_MAX_PS_NAME_LENGTH+1];
```

```
   /* 作用原則集名稱           */
   dsmDate        polActDate;             /* 原則集啟動日期       */
   dsChar_t       dfltMCName[DSM_MAX_MC_NAME_LENGTH+1];/* 預設管理類別  */
   dsUint16_t     gpBackRetn;             /* 寬限期備份保留     */
   dsUint16_t     gpArchRetn;             /* 寬限期保存保留    */
   dsChar_t       adsmServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* ADSM 伺服器名稱 */
  dsmBool_t      archiveRetentionProtection; /* 是否啟用伺服器保留保護 */
   dsUint64_t     maxBytesPerTxn_64;         /* 供未來使用                    */
   dsmBool_t      lanFreeEnabled;            /* 已設定不需 LAN 選項         */
   dsmDedupType   dedupType;                 /* 伺服器或 clientOrServer       */
   dsChar_t       accessNode[DSM_MAX_ID_LENGTH+1];    /* 作為節點名稱         */

   /*-------------------------------------------------------------------*/
   /*           抄寫和失效接手資訊                                      */
   /*-------------------------------------------------------------------*/
   dsmFailOvrCfgType failOverCfgType; /* 失效接手狀態 */
   dsChar_t       replServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* 抄寫伺服器名稱 */
   dsChar_t       homeServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* 起始伺服器名稱 */
   dsChar_t       replServerHost[DSM_MAX_SERVERNAME_LENGTH+1]; /* DSM 伺服器的網路主機名稱  */
   dsInt32_t      replServerPort;                              /* 主機上的伺服器通訊埠          */

} tsmApiSessInfo;

#define tsmApiSessInfoVersion  6

/*-------------------------------------------------------------------------+
 | 「查詢保存」回應在 dsmQueryCliOptions()                                  |
 |      和 dsmQuerySessOptions() 上的類型定義                               |
 +-------------------------------------------------------------------------*/

typedef struct
{
   dsUint16_t  stVersion;
   dsChar_t        dsmiDir[DSM_PATH_MAX + DSM_NAME_MAX +1];
   dsChar_t        dsmiConfig[DSM_PATH_MAX + DSM_NAME_MAX +1];
   dsChar_t    serverName[DSM_MAX_SERVERNAME_LENGTH+1];
   dsInt16_t   commMethod;
   dsChar_t    serverAddress[DSM_MAX_SERVER_ADDRESS];
   dsChar_t    nodeName[DSM_MAX_NODE_LENGTH+1];
   dsmBool_t   compression;
   dsmBool_t   compressalways;
   dsmBool_t   passwordAccess;
}tsmOptStruct ;

#define tsmOptStructVersion  1

/*-------------------------------------------------------------------------+
 | 使用的 qryRespAccessData 參數在 dsmQueryAccess() 上的類型定義           |
 +-------------------------------------------------------------------------*/

typedef struct
{
   dsUint16_t        stVersion ;                  /* 結構版本      */
   dsChar_t          node[DSM_MAX_ID_LENGTH+1];     /* 節點名稱               */
   dsChar_t          owner[DSM_MAX_OWNER_LENGTH+1]; /* 擁有者                   */
   tsmObjName        objName ;                    /* 物件名稱             */
   dsmAccessType     accessType;                  /* 保存或備份     */
   dsUint32_t        ruleNumber ;                 /* 存取權規則 ID         */

}tsmQryRespAccessData;

#define tsmQryRespAccessDataVersion  1

/*-------------------------------------------------------------------------+
 |  envSetUp 參數在 dsmSetUp() 上的類型定義
 +-------------------------------------------------------------------------*/
typedef struct tsmEnvSetUp
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsChar_t        dsmiDir[DSM_PATH_MAX + DSM_NAME_MAX +1];
   dsChar_t        dsmiConfig[DSM_PATH_MAX + DSM_NAME_MAX +1];
   dsChar_t        dsmiLog[DSM_PATH_MAX + DSM_NAME_MAX +1];
   char            **argv;  /* 適用於可執行檔名稱 argv[0] */
   dsChar_t        logName[DSM_NAME_MAX +1];
   dsmBool_t       reserved1;                  /* 供未來使用 */
   dsmBool_t       reserved2;                  /* 供未來使用 */
} tsmEnvSetUp;

#define tsmEnvSetUpVersion  4
```

```
/*-------------------------------------------------------------------------+
 |  適用於 dsmInitExIn_t 的類型定義
 +-------------------------------------------------------------------------*/
typedef struct tsmInitExIn_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
   tsmApiVersionEx    *apiVersionExP;
   dsChar_t           *clientNodeNameP;
   dsChar_t           *clientOwnerNameP;
   dsChar_t           *clientPasswordP;
   dsChar_t           *userNameP;
   dsChar_t           *userPasswordP;
   dsChar_t           *applicationTypeP;
   dsChar_t           *configfile;
   dsChar_t           *options;
   dsChar_t           dirDelimiter;
   dsmBool_t          useUnicode;
   dsmBool_t          bCrossPlatform;
   dsmBool_t          bService;
   dsmBool_t          bEncryptKeyEnabled;
   dsChar_t           *encryptionPasswordP;
   dsmBool_t          useTsmBuffers;
   dsUint8_t          numTsmBuffers;
   tsmAppVersion      appVersionP;
} tsmInitExIn_t;

#define tsmInitExInVersion 5

/*-------------------------------------------------------------------------+
 |  適用於 dsmInitExOut_t 的類型定義
 +-------------------------------------------------------------------------*/
typedef struct tsmInitExOut_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsInt16_t          userNameAuthorities;
   dsInt16_t          infoRC;        /* 若發現的話，會有錯誤回覆碼 */
   /* ADSM 伺服器名稱                */
   dsChar_t           adsmServerName[DSM_MAX_SERVERNAME_LENGTH+1];
   dsUint16_t     serverVer;              /* 伺服器的版本號碼          */
   dsUint16_t         serverRel;     /* 伺服器的版本號碼          */
   dsUint16_t         serverLev;     /* 伺服器的層級號碼            */
   dsUint16_t         serverSubLev;  /* 伺服器的次層級號碼         */
   dsmBool_t          bIsFailOverMode; /* 如果發生了失效接手則為 true */
   dsChar_t           replServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* 抄寫伺服器名稱 */
   dsChar_t           homeServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* 起始伺服器名稱 */
} tsmInitExOut_t;

#define tsmInitExOutVersion 3

/*-------------------------------------------------------------------------+
 |  適用於 dsmLogExIn_t 的類型定義
 +-------------------------------------------------------------------------*/
typedef struct tsmLogExIn_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsmLogSeverity     severity;
   dsChar_t           appMsgID[8];
   dsmLogType         logType;    /* 日誌類型：本端，伺服器或兩者 */
   dsChar_t           *message;   /* 要記載的訊息文字 */
   dsChar_t           appName[DSM_MAX_PLATFORM_LENGTH];
   dsChar_t           osPlatform[DSM_MAX_PLATFORM_LENGTH];
   dsChar_t           appVersion[DSM_MAX_PLATFORM_LENGTH];

} tsmLogExIn_t;

#define tsmLogExInVersion 2

/*-------------------------------------------------------------------------+
 |  適用於 dsmlogExOut_t 的類型定義
 +-------------------------------------------------------------------------*/
typedef struct tsmLogExOut_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
} tsmLogExOut_t;

#define tsmLogExOutVersion 1

/*-------------------------------------------------------------------------+
 |  適用於 dsmRenameIn_t 的類型定義
```

```
 +-------------------------------------------------------------------------*/
typedef struct tsmRenameIn_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsUint32_t       tsmHandle;                    /* 階段作業的控點 */
   dsUint8_t        repository;                   /* 備份或保存  */
   tsmObjName       *objNameP ;                   /* 物件名稱 */
   dsChar_t         newHl[DSM_MAX_HL_LENGTH + 1];  /* 新的高階名稱 */
   dsChar_t         newLl[DSM_MAX_LL_LENGTH + 1];  /* 新的低階名稱 */
   dsmBool_t        合併;                       /* 合併至現存名稱 */
   ObjID            objId;                        /* 保存的 objId */
} tsmRenameIn_t;

#define tsmRenameInVersion 1

/*-------------------------------------------------------------------------+
 |  適用於 dsmRenameOut_t 的類型定義
 +-------------------------------------------------------------------------*/
typedef struct tsmRenameOut_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
} tsmRenameOut_t;

#define tsmRenameOutVersion 1

/*-------------------------------------------------------------------------+
 |  適用於 tsmEndSendObjExIn_t 的類型定義
 +-------------------------------------------------------------------------*/
typedef struct tsmEndSendObjExIn_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsUint32_t       tsmHandle;                    /* 階段作業的控點 */
} tsmEndSendObjExIn_t;

#define tsmEndSendObjExInVersion 1

/*-------------------------------------------------------------------------+
 |  適用於 dsmEndSendObjExOut_t 的類型定義
 +-------------------------------------------------------------------------*/
typedef struct tsmEndSendObjExOut_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsStruct64_t       totalBytesSent;      /* 從 app 讀取總位元組數 */
   dsmBool_t          objCompressed;       /* 物件是否已壓縮             */
   dsStruct64_t       totalCompressSize;   /* 壓縮之後的大小總計 */
   dsStruct64_t       totalLFBytesSent;    /* 不需 LAN 的總送出位元組數 */
   dsUint8_t          encryptionType;      /* 使用的加密類型    */
   dsmBool_t          objDeduplicated;     /* 物件是否已經過刪除重複資料處理 */
   dsStruct64_t       totalDedupSize;      /* 刪除重複資料之後的大小總計 */
}tsmEndSendObjExOut_t;

#define tsmEndSendObjExOutVersion 3

/*-------------------------------------------------------------------------+
 |  適用於 tsmGroupHandlerIn_t 的類型定義
 +-------------------------------------------------------------------------*/
typedef struct tsmGroupHandlerIn_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsUint32_t       tsmHandle;        /* 階段作業的控點                          */
   dsUint8_t        groupType;        /* 群組類型                                */
   dsUint8_t        actionType;       /* 群組作業類型              */
   dsUint8_t        memberType;       /* 成員類型：主導器或成員    */
   dsStruct64_t     leaderObjId;      /* 群組前導的 OBJID            */
   dsChar_t         *uniqueGroupTagP; /* 唯一的群組 ID             */
   tsmObjName       *objNameP ;       /* 群組前導物件名稱            */
   dsmGetList       memberObjList;    /* 移除及指定的物件清單    */
} tsmGroupHandlerIn_t;

#define tsmGroupHandlerInVersion 1

/*-------------------------------------------------------------------------+
 |  適用於 tsmGroupHandlerExOut_t 的類型定義
 +-------------------------------------------------------------------------*/
typedef struct tsmGroupHandlerOut_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
} tsmGroupHandlerOut_t;
```

```
#define tsmGroupHandlerOutVersion 1

/*-------------------------------------------------------------------------+
 |  適用於 tsmEndTxnExIn_t 的類型定義
 +-------------------------------------------------------------------------*/
typedef struct tsmEndTxnExIn_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsUint32_t       tsmHandle;                     /* 階段作業的控點 */
   dsUint8_t        vote;
} tsmEndTxnExIn_t;

#define tsmEndTxnExInVersion 1

/*-------------------------------------------------------------------------+
 |  適用於 tsmEndTxnExOut_t 的類型定義
 +-------------------------------------------------------------------------*/
typedef struct tsmEndTxnExOut_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsUint16_t         原因 ;                  /* 原因碼                      */
   dsStruct64_t       groupLeaderObjId;         /* groupLeader 物件 ID 返回 */
   /* DSM_ACTION_OPEN                */
    dsUint8_t          reserved1;               /* 供未來使用                      */
    dsUint16_t         reserved2;               /* 供未來使用                      */
} tsmEndTxnExOut_t;

#define tsmEndTxnExOutVersion 1

/*-------------------------------------------------------------------------+
 |  適用於 tsmEndGetDataExIn_t 的類型定義
 +-------------------------------------------------------------------------*/
typedef struct tsmEndGetDataExIn_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsUint32_t     tsmHandle;   /* 階段作業的控點 */
}tsmEndGetDataExIn_t;

#define tsmEndGetDataExInVersion 1

/*-------------------------------------------------------------------------+
 |  適用於 tsmEndGetDataExOut_t 的類型定義
 +-------------------------------------------------------------------------*/
typedef struct tsmEndGetDataExOut_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsUint16_t     reason;            /* 原因碼                     */
   dsStruct64_t   totalLFBytesRecv;  /* 已接收的不需 LAN 位元組總數 */
}tsmEndGetDataExOut_t;

#define tsmEndGetDataExOutVersion 1

/*-------------------------------------------------------------------------+
 |  適用於在 tsmRetentionEvent() 上的類型定義              |
 +-------------------------------------------------------------------------*/
typedef struct tsmRetentionEventIn_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsUint32_t         tsmHandle;                    /* 階段作業 Handle     */
   dsmEventType_t     eventType;                    /* 事件類型            */
   dsmObjList_t       objList;                      /* 物件 ID             */
}tsmRetentionEventIn_t;

#define tsmRetentionEventInVersion 1

/*-------------------------------------------------------------------------+
 |  適用於在 tsmRetentionEvent() 上的類型定義              |
 +-------------------------------------------------------------------------*/
typedef struct tsmRetentionEventOut_t
{
   dsUint16_t        stVersion ;                    /* 結構版本      */
}tsmRetentionEventOut_t;

#define tsmRetentionEventOutVersion 1

/*-------------------------------------------------------------------------+
 |  適用於 tsmUpdateObjExIn_t 的類型定義
 +-------------------------------------------------------------------------*/
```

```
typedef struct tsmUpdateObjExIn_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
   dsUint32_t         tsmHandle;                    /* 階段作業 Handle     */
   tsmSendType     sendType;             /* 傳送類型備份/保存 */
   dsChar_t        *descrP;              /* 保存說明           */
   tsmObjName      *objNameP;            /* 物件名稱           */
   tsmObjAttr      *objAttrPtr;          /* 屬性               */
   dsUint32_t      objUpdAct;            /* 更新動作           */
   ObjID           archObjId;            /* 保存的 objId        */
}tsmUpdateObjExIn_t;

#define tsmUpdateObjExInVersion 1

/*-------------------------------------------------------------------------+
 |  適用於 tsmUpdateObjExOut_t 的類型定義
 +-------------------------------------------------------------------------*/
typedef struct tsmUpdateObjExOut_t
{
   dsUint16_t     stVersion;   /* 結構版本  */
}tsmUpdateObjExOut_t;

#define tsmUpdateObjExOutVersion 1

#if _OPSYS_TYPE == DS_WINNT
#pragma pack()
#endif


#ifdef _MAC
#pragma options align = reset
#endif
#endif /* _H_TSMAPITD */
```

dsmapips.h

```
/***********************************************************************
*       IBM Spectrum Protect             *
* API Client Component                                                 *
*                                                                      *
* IBM Confidential                                                     *
* (IBM Confidential-Restricted when combined with the Aggregated OCO   *
* source modules for this program)                                     *
*                                                                      *
* OCO Source Materials                                                 *
*                                                                      *
* 5648-020 (C) Copyright IBM Corporation 1993, 2016                    *
***********************************************************************/

/**************************************************************************
* 標頭檔名稱：dsmapips.h
*
* 環境              *********************************************
*                    ** 這是適用於 Windows NT 的特定平台        **
*                    ** 來源檔版本                              **
*
*                    *********************************************
*
* 設計注意事項：     這個檔案包括平台相依定義
*
* 敘述名稱：IBM Spectrum Protect typedefs 和 LINKAGE 的定義
*-------------------------------------------------------------------------*/

#ifndef _H_DSMAPIPS
#define _H_DSMAPIPS


#ifndef _WIN64
#pragma pack(1)
#endif

/*<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>*/
/*                      T Y P E D E F S                                   */
/*<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>*/
```

```
/* 新的 typedef 檔案適用於版本 3   */


#if !defined(DSMAPILIB) || defined (XOPEN_BUILD)

/* 適用於支援鏈結 */
#include <windows.h>
#define DSMLINKAGE WINAPI


#define DS_WINNT    22
#define _OPSYS_TYPE DS_WINNT

   typedef signed   char  dsInt8_t;
   typedef unsigned char  dsUint8_t;
   typedef signed   short dsInt16_t;
   typedef unsigned short dsUint16_t;
   typedef signed   long  dsInt32_t;
   typedef unsigned long  dsUint32_t;

/*=== 字元及字串類型 ===*/
#ifdef UNICODE
  typedef wchar_t dsChar_t;
  #define dsTEXT(x)       L##x
#else
  typedef char dsChar_t;
  #define dsTEXT(x)       x
#endif  /* !UNICODE */

/*=== 共同 typedefs 及定義衍生從 dsChar_t ===*/
typedef dsChar_t            *dsString_t;

  /* 為延伸還原次序新增 */
   typedef struct
   {
     dsUint32_t top;
     dsUint32_t hi_hi;
     dsUint32_t hi_lo;
     dsUint32_t lo_hi;
     dsUint32_t lo_lo;
   } dsUint160_t ;

#if defined(_LONG_LONG)
   typedef __int64              dsInt64_t;
   typedef unsigned __int64     dsUint64_t;
   /*=== A "true" unsigned 64-bit integer ===*/
   typedef __int64              dsLongLong_t;
#else
typedef struct tagUINT64_t
   {
      dsUint32_t hi;        /* 最大有效的 32 位元。 */
      dsUint32_t lo;        /* 最小有效的 32 位元。 */
   } dsUint64_t;
#endif

/*-------------------------------------------------------------------------+
| 適用於 bool_t 的類型定義                                                 |
+-------------------------------------------------------------------------*/
/*
 *  必須建立沒有和任何其他預先定義的作業系統
 *  或 Windows 系統相衝突的布林版本。
 */
typedef enum
{
   dsmFalse = 0x00,
   dsmTrue  = 0x01
}dsmBool_t ;


/*===  適用於往回相容性  ===*/
#define uint8    dsUint8_t
#define int8     dsInt8_t
#define uint16   dsUint16_t
#define int16    dsInt16_t
#define uint32   dsUint32_t
#define int32    dsInt32_t
#define uint64   dsStruct64_t
#define bool_t   dsBool_t
#define dsBool_t dsmBool_t
#define bTrue    dsmTrue
#define bFalse   dsmFalse
```

```
typedef struct
{
   dsUint32_t hi;        /* 最大有效的 32 位元。 */
   dsUint32_t lo;        /* 最小有效的 32 位元。 */
}dsStruct64_t ;

#endif /* DSMAPILIB */



#ifndef _WIN64
#pragma pack()
#endif
#endif  /* _H_DSMAPIPS */
```

release.h

```
/*****************************************************************************
*       IBM Spectrum Protect             *
* Common Source Component                                                    *
*                                                                            *
* (C) Copyright IBM Corporation 1993,2018                                    *
*****************************************************************************/

/*****************************************************************************
* 標頭檔名稱：release.h
*
* 環境：            ************************************************
*                    ** 這是一個 platform-independent 原始檔       **
*                    ************************************************
*
* 設計注意事項：    這個檔案包含有關於實際 version.release.level.sublevel
*                    的一般相關資訊
*
* 敘述名稱：IBM Spectrum Protect 版本的定義
*
* 附註：這個檔案不應包含 LOG 或 CMVC 資訊。It is
*       這是隨 API 程式碼出貨。
*
*----------------------------------------------------------------------------*/

#ifndef _H_RELEASE
#define _H_RELEASE

#define COMMON_VERSION         8
#define COMMON_RELEASE         1
#define COMMON_LEVEL           12
#define COMMON_SUBLEVEL        0
#define COMMON_DRIVER  dsTEXT("")

#define COMMON_VERSIONTXT "8.1.12.0"

#define SHIPYEARTXT  ""
#define SHIPYEARTXTW dsTEXT("2021")
#define TSMPRODTXT  "IBM Spectrum Protect"

/*============================================================================
   下列字串定義用於 VERSION 資訊
   而且不應轉換成 dsTEXT 或 osTEXT。只有在
   鏈結時才使用它們。

   當 Jar 檔是建置於 Unix 時也會使用這些。  請參閱
   perl script tools/unx/mzbuild/createReleaseJava
============================================================================*/
#define COMMON_VERSION_STR    "8"
#define COMMON_RELEASE_STR    "1"
#define COMMON_LEVEL_STR      "12"
#define COMMON_SUBLEVEL_STR   "0"
#define COMMON_DRIVER_STR     ""

/*=== 產品名稱定義 ===*/
#define COMMON_NAME_DFDSM      1
#define COMMON_NAME_ADSM       2
#define COMMON_NAME_TSM        3
#define COMMON_NAME_ITSM       4
#define COMMON_NAME            COMMON_NAME_ITSM
```

```
/*============================================================================
       內部版本、版次和層次 ( 建置 ) 版本。對於
    產品的每一個 version+release+ptf 來說，這必須是唯一的。
    本資訊記錄於檔案屬性和資料
    串流以供診斷之用。
    附註：請勿修改這些值。只能新增項目！
============================================================================*/
#define COMMON_BUILD_TSM_510   1
#define COMMON_BUILD_TSM_511   2
#define COMMON_BUILD_TSM_515   3
#define COMMON_BUILD_TSM_516   4
#define COMMON_BUILD_TSM_520   5
#define COMMON_BUILD_TSM_522   6
#define COMMON_BUILD_TSM_517   7
#define COMMON_BUILD_TSM_523   8
#define COMMON_BUILD_TSM_530   9
#define COMMON_BUILD_TSM_524   10
#define COMMON_BUILD_TSM_532   11
#define COMMON_BUILD_TSM_533   12
#define COMMON_BUILD_TSM_525   13
#define COMMON_BUILD_TSM_534   14
#define COMMON_BUILD_TSM_540   15
#define COMMON_BUILD_TSM_535   16
#define COMMON_BUILD_TSM_541   17
#define COMMON_BUILD_TSM_550   18
#define COMMON_BUILD_TSM_542   19
#define COMMON_BUILD_TSM_551   20
#define COMMON_BUILD_TSM_610   21
#define COMMON_BUILD_TSM_552   22
#define COMMON_BUILD_TSM_611   23
#define COMMON_BUILD_TSM_543   24
#define COMMON_BUILD_TSM_620   25
#define COMMON_BUILD_TSM_612   26
#define COMMON_BUILD_TSM_553   27
#define COMMON_BUILD_TSM_613   28
#define COMMON_BUILD_TSM_621   29
#define COMMON_BUILD_TSM_622   30
#define COMMON_BUILD_TSM_614   31
#define COMMON_BUILD_TSM_623   32
#define COMMON_BUILD_TSM_630   33
#define COMMON_BUILD_TSM_615   34
#define COMMON_BUILD_TSM_624   35
#define COMMON_BUILD_TSM_631   36
#define COMMON_BUILD_TSM_640   37
#define COMMON_BUILD_TSM_710   38
#define COMMON_BUILD_TSM_625   39
#define COMMON_BUILD_TSM_641   40
#define COMMON_BUILD_TSM_711   41
#define COMMON_BUILD_TSM_712   42
#define COMMON_BUILD_TSM_713   43
#define COMMON_BUILD_TSM_714   44
#define COMMON_BUILD_TSM_720   45
#define COMMON_BUILD_TSM_721   46
#define COMMON_BUILD_TSM_642   47
#define COMMON_BUILD_TSM_643   48
#define COMMON_BUILD_TSM_715   49
#define COMMON_BUILD_TSM_716   50
#define COMMON_BUILD_TSM_810   51
#define COMMON_BUILD_TSM_811   52
#define COMMON_BUILD_TSM_812   53
#define COMMON_BUILD_TSM_718   54
#define COMMON_BUILD_TSM_814   55
#define COMMON_BUILD_TSM_816   56
#define COMMON_BUILD_TSM_817   57
#define COMMON_BUILD_TSM_818   58
#define COMMON_BUILD_TSM_819   59
#define COMMON_BUILD_TSM_8110  60
#define COMMON_BUILD_TSM_8111  61
#define COMMON_BUILD_TSM_8112  62
#define COMMON_BUILD           COMMON_BUILD_TSM_8112

/*=== 將 VRL 定義為 Int 以進行點陣圖版本比較 ====*/
static const  int  VRL_712 = 712;
static const  int  VRL_713 = 713;
static const  int  VRL_714 = 714;
static const  int  VRL_715 = 715;
static const  int  VRL_716 = 716;
static const  int  VRL_718 = 718;
static const  int  VRL_810 = 810;
static const  int  VRL_811 = 811;
static const  int  VRL_812 = 812;
```

```
static const  int  VRL_814 = 814;
static const  int  VRL_816 = 816;
static const  int  VRL_817 = 817;
static const  int  VRL_818 = 818;
static const  int  VRL_819 = 819;
static const  int  VRL_8110 = 8110;
static const  int  VRL_8111 = 8111;
static const  int  VRL_8112 = 8112;

#define   TDP4VE_PLATFORM_STRING_MBCS     "TDP VMware"
#define   TDP4VE_PLATFORM_STRING          dsTEXT("TDP VMware")

#define   TDP4HYPERV_PLATFORM_STRING_MBCS "TDP HyperV"
#define   TDP4HYPERV_PLATFORM_STRING      dsTEXT("TDP HyperV")

#endif /* _H_RELEASE */
```

附錄 C API 函數定義原始檔

本附錄包含 dsmapifp.h 標頭檔，因此您可以查看 API 的函數定義。

註: DSMLINKAGE 在 每一個作業系統上的定義並不同。 請參閱您的作業系統專用的 dsmapips.h 檔案中的定義。

在這裡提供的資訊包含使用 API 配送的檔案的時間點副本。檢視 API 配送套件中的檔案以取得最新版本。

dsmapifp.h

```
/*************************************************************************
*       IBM Spectrum Protect            *
* API Client Component                                                   *
*                                                                        *
* IBM Confidential                                                       *
* (IBM Confidential-Restricted when combined with the Aggregated OCO   *
* source modules for this program)                                       *
*                                                                        *
* OCO Source Materials                                                   *
*                                                                        *
* 5648-020 (C) Copyright IBM Corporation 1993, 2016                     *
*************************************************************************/

/*************************************************************************/
/* 標頭檔名稱：dsmapifp.h                                                */
/*                                                                       */
/* 敘述名稱：IBM Spectrum Protect API 函數原型                          */
/*************************************************************************/
#ifndef _H_DSMAPIFP
#define _H_DSMAPIFP

#if defined(__cplusplus)
extern "C" {
#endif

#ifdef DYNALOAD_DSMAPI

/* 將會動態載入函數 */
#include "dsmapidl.h"

#else

/* 將從程式庫隱含地載入函數 */

/*************************************************************************/
/*               P U B L I C   F U N C T I O N S                         */
/*************************************************************************/


extern dsInt16_t DSMLINKAGE  dsmBeginGetData(
       dsUint32_t             dsmHandle,
       dsBool_t               mountWait,
       dsmGetType             getType,
       dsmGetList             *dsmGetObjListP
);

extern dsInt16_t DSMLINKAGE  dsmBeginQuery(
       dsUint32_t             dsmHandle,
       dsmQueryType           queryType,
       dsmQueryBuff           *queryBuffer
);

extern dsInt16_t DSMLINKAGE  dsmBeginTxn(
       dsUint32_t             dsmHandle
);

extern dsInt16_t DSMLINKAGE  dsmBindMC(
       dsUint32_t             dsmHandle,
       dsmObjName             *objNameP,
       dsmSendType            sendType,
       mcBindKey              *mcBindKeyP
);
```

```
extern dsInt16_t DSMLINKAGE  dsmChangePW(
       dsUint32_t             dsmHandle,
       char                  *oldPW,
       char                  *newPW
);

extern dsInt16_t DSMLINKAGE dsmCleanUp(
       dsBool_t               mtFlag
);


extern dsInt16_t DSMLINKAGE  dsmDeleteAccess(
       dsUint32_t             dsmHandle,
       dsUint32_t            ruleNum
);

extern dsInt16_t DSMLINKAGE  dsmDeleteObj(
       dsUint32_t             dsmHandle,
       dsmDelType            delType,
       dsmDelInfo            delInfo
);

extern dsInt16_t DSMLINKAGE  dsmDeleteFS(
       dsUint32_t             dsmHandle,
       char                  *fsName,
       dsUint8_t             repository
);

extern dsInt16_t DSMLINKAGE  dsmEndGetData(
       dsUint32_t            dsmHandle
);

extern dsInt16_t DSMLINKAGE  dsmEndGetDataEx(
       dsmEndGetDataExIn_t  *dsmEndGetDataExInP,
       dsmEndGetDataExOut_t *dsmEndGetDataExOutP
);

extern dsInt16_t DSMLINKAGE  dsmEndGetObj(
       dsUint32_t            dsmHandle
);

extern dsInt16_t DSMLINKAGE  dsmEndQuery(
       dsUint32_t            dsmHandle
);

extern dsInt16_t DSMLINKAGE  dsmEndSendObj(
       dsUint32_t            dsmHandle
);

extern dsInt16_t DSMLINKAGE  dsmEndSendObjEx(
       dsmEndSendObjExIn_t   *dsmEndSendObjExInP,
       dsmEndSendObjExOut_t  *dsmEndSendObjExOutP
);

extern dsInt16_t DSMLINKAGE  dsmEndTxnEx(
       dsmEndTxnExIn_t      *dsmEndTxnExInP,
       dsmEndTxnExOut_t     *dsmEndTxnExOutP
);

extern dsInt16_t DSMLINKAGE  dsmEndTxn(
       dsUint32_t             dsmHandle,
       dsUint8_t             vote,
       dsUint16_t            *reason
);

extern dsInt16_t DSMLINKAGE  dsmGetData(
       dsUint32_t             dsmHandle,
       DataBlk                *dataBlkPtr
);

extern dsInt16_t DSMLINKAGE  dsmGetBufferData(
       getBufferDataIn_t     *dsmGetBufferDataInP,
       getBufferDataOut_t    *dsmGetBufferDataOutP
);

extern dsInt16_t DSMLINKAGE  dsmGetNextQObj(
       dsUint32_t             dsmHandle,
       DataBlk               *dataBlkPtr
) ;

extern dsInt16_t DSMLINKAGE  dsmGetObj(
       dsUint32_t             dsmHandle,
```

```
       ObjID                *objIdP,
       DataBlk               *dataBlkPtr
);

extern dsInt16_t DSMLINKAGE  dsmGroupHandler(
       dsmGroupHandlerIn_t   *dsmGroupHandlerInP,
       dsmGroupHandlerOut_t  *dsmGroupHandlerOutP
);

extern dsInt16_t DSMLINKAGE  dsmInit(
       dsUint32_t            *dsmHandle,
       dsmApiVersion         *dsmApiVersionP,
       char                  *clientNodeNameP,
       char                  *clientOwnerNameP,
       char                  *clientPasswordP,
       char                  *applicationType,
       char                  *configfile,
       char                  *options
);

extern dsInt16_t DSMLINKAGE  dsmInitEx(
       dsUint32_t            *dsmHandleP,
       dsmInitExIn_t         *dsmInitExInP,
       dsmInitExOut_t        *dsmInitExOutP
);


extern dsInt16_t DSMLINKAGE  dsmLogEvent(
       dsUint32_t             dsmHandle,
       logInfo               *lopInfoP
);

extern dsInt16_t DSMLINKAGE  dsmLogEventEx(
       dsUint32_t             dsmHandle,
       dsmLogExIn_t          *dsmLogExInP,
       dsmLogExOut_t         *dsmLogExOutP
);

extern dsInt16_t DSMLINKAGE  dsmQueryAccess(
       dsUint32_t             dsmHandle,
       qryRespAccessData     **accessListP,
       dsUint16_t            *numberOfRules) ;

extern void DSMLINKAGE       dsmQueryApiVersion(
       dsmApiVersion         *apiVersionP
);

extern void DSMLINKAGE       dsmQueryApiVersionEx(
       dsmApiVersionEx       *apiVersionP
);

extern dsInt16_t DSMLINKAGE  dsmQueryCliOptions(
       optStruct             *optstructP
);

extern dsInt16_t DSMLINKAGE  dsmQuerySessInfo(
       dsUint32_t             dsmHandle,
       ApiSessInfo           *SessInfoP
);

extern dsInt16_t DSMLINKAGE  dsmQuerySessOptions(
       dsUint32_t             dsmHandle,
       optStruct             *optstructP
);

extern dsInt16_t DSMLINKAGE  dsmRCMsg(
       dsUint32_t             dsmHandle,
       dsInt16_t             dsmRC,
       char                  *msg
);

extern dsInt16_t DSMLINKAGE  dsmRegisterFS(
       dsUint32_t             dsmHandle,
       regFSData             *regFilespaceP
);

extern dsInt16_t DSMLINKAGE   dsmReleaseBuffer(
       releaseBufferIn_t      *dsmReleaseBufferInP,
       releaseBufferOut_t     *dsmReleaseBufferOutP
);

extern dsInt16_t DSMLINKAGE   dsmRenameObj(
```

```
       dsmRenameIn_t          *dsmRenameInP,
       dsmRenameOut_t         *dsmRenameOutP
);

extern dsInt16_t DSMLINKAGE  dsmRequestBuffer(
       requestBufferIn_t     *dsmRequestBufferInP,
       requestBufferOut_t    *dsmRequestBufferOutP
);

extern dsInt16_t DSMLINKAGE   dsmRetentionEvent(
       dsmRetentionEventIn_t  *dsmRetentionEventInP,
       dsmRetentionEventOut_t *dsmRetentionEventOutP
);

extern dsInt16_t DSMLINKAGE   dsmSendBufferData(
       sendBufferDataIn_t     *dsmSendBufferDataInP,
       sendBufferDataOut_t    *dsmSendBufferDataOutP
);

extern dsInt16_t DSMLINKAGE   dsmSendData(
       dsUint32_t             dsmHandle,
       DataBlk                *dataBlkPtr
) ;

extern dsInt16_t DSMLINKAGE   dsmSendObj(
       dsUint32_t             dsmHandle,
       dsmSendType            sendType,
       void                   *sendBuff,
       dsmObjName             *objNameP,
       ObjAttr                *objAttrPtr,
       DataBlk                *dataBlkPtr
);

extern dsInt16_t DSMLINKAGE   dsmSetAccess(
       dsUint32_t             dsmHandle,
       dsmAccessType          accessType,
       dsmObjName             *objNameP,
       char                   *node,
       char                   *owner
);

extern dsInt16_t DSMLINKAGE   dsmSetUp(
       dsBool_t               mtFlag,
       envSetUp               *envSetUpP
);

extern dsInt16_t DSMLINKAGE   dsmTerminate(
       dsUint32_t             dsmHandle
);

extern dsInt16_t DSMLINKAGE   dsmUpdateFS(
       dsUint32_t             dsmHandle,
       char                   *fs,
       dsmFSUpd               *fsUpdP,
       dsUint32_t             fsUpdAct
);

extern dsInt16_t DSMLINKAGE   dsmUpdateObj(
       dsUint32_t             dsmHandle,
       dsmSendType            sendType,
       void                   *sendBuff,
       dsmObjName             *objNameP,
       ObjAttr                *objAttrPtr,
       dsUint32_t             objUpdAct

);

extern dsInt16_t DSMLINKAGE   dsmUpdateObjEx(
       dsmUpdateObjExIn_t     *dsmUpdateObjExInP,
       dsmUpdateObjExOut_t    *dsmUpdateObjExOutP
);


#endif /* ifdef DYNALOAD */

#if defined(__cplusplus)
   }
#endif

#endif /* _H_DSMAPIFP */
```

tsmapifp.h

本節包含 API 的函數定義。 這是 tsmapifp.h 標頭檔的副本。

註: DSMLINKAGE 在 每一個作業系統上的定義並不同。 請參閱您的作業系統專用的 tsmapips.h 檔案中的定義。

```
/*************************************************************************
*       IBM Spectrum Protect            *
* API Client Component                                                   *
*                                                                        *
* IBM Confidential                                                       *
* (IBM Confidential-Restricted when combined with the Aggregated OCO   *
* source modules for this program)                                       *
*                                                                        *
* OCO Source Materials                                                   *
*                                                                        *
* 5648-020 (C) Copyright IBM Corporation 1993, 2016                     *
*************************************************************************/

/*************************************************************************/
/* 標頭檔名稱：tsmapifp.h                                    */
/*                                                                   */
/* 敘述名稱：IBM Spectrum Protect API 函數原型                       */
/*************************************************************************/
#ifndef _H_TSMAPIFP
#define _H_TSMAPIFP

#if defined(__cplusplus)
extern "C" {
#endif

#ifdef DYNALOAD_DSMAPI

/* 將會動態載入函數 */
#include "dsmapidl.h"

#else

/* 將從程式庫隱含地載入函數 */

/*=======================================================================*/
/*P U B L I C   F U N C T I O N S                                         */
/*=======================================================================*/


typedef void tsmQueryBuff;

extern dsInt16_t DSMLINKAGE  tsmBeginGetData(
       dsUint32_t            tsmHandle,
       dsBool_t              mountWait,
       tsmGetType            getType,
       dsmGetList            *dsmGetObjListP
);

extern dsInt16_t DSMLINKAGE  tsmBeginQuery(
       dsUint32_t            tsmHandle,
       tsmQueryType          queryType,
       tsmQueryBuff          *queryBuffer
);

extern dsInt16_t DSMLINKAGE  tsmBeginTxn(
       dsUint32_t            tsmHandle
);

extern dsInt16_t DSMLINKAGE  tsmBindMC(
       dsUint32_t            tsmHandle,
       tsmObjName            *objNameP,
       tsmSendType           sendType,
       tsmMcBindKey          *mcBindKeyP
);

extern dsInt16_t DSMLINKAGE  tsmChangePW(
       dsUint32_t            tsmHandle,
       dsChar_t              *oldPW,
       dsChar_t              *newPW
);

extern dsInt16_t DSMLINKAGE tsmCleanUp(
```

```
       dsBool_t              mtFlag
);


extern dsInt16_t DSMLINKAGE  tsmDeleteAccess(
       dsUint32_t            tsmHandle,
       dsUint32_t            ruleNum
);

extern dsInt16_t DSMLINKAGE  tsmDeleteObj(
       dsUint32_t            tsmHandle,
       tsmDelType            delType,
       tsmDelInfo            delInfo
);

extern dsInt16_t DSMLINKAGE  tsmDeleteFS(
       dsUint32_t            tsmHandle,
       dsChar_t              *fsName,
       dsUint8_t             repository
);

extern dsInt16_t DSMLINKAGE  tsmEndGetData(
       dsUint32_t            tsmHandle
);

extern dsInt16_t DSMLINKAGE  tsmEndGetDataEx(
       tsmEndGetDataExIn_t      *tsmEndGetDataExInP,
       tsmEndGetDataExOut_t     *tsmEndGetDataExOutP
);

extern dsInt16_t DSMLINKAGE  tsmEndGetObj(
       dsUint32_t            tsmHandle
);

extern dsInt16_t DSMLINKAGE  tsmEndQuery(
       dsUint32_t            tsmHandle
);

extern dsInt16_t DSMLINKAGE  tsmEndSendObj(
       dsUint32_t            tsmHandle
);

extern dsInt16_t DSMLINKAGE  tsmEndSendObjEx(
       tsmEndSendObjExIn_t      *tsmEndSendObjExInP,
       tsmEndSendObjExOut_t     *tsmEndSendObjExOutP
);

extern dsInt16_t DSMLINKAGE  tsmEndTxn(
       dsUint32_t            tsmHandle,
       dsUint8_t             vote,
       dsUint16_t            *reason
);

extern dsInt16_t DSMLINKAGE  tsmEndTxnEx(
       tsmEndTxnExIn_t      *tsmEndTxnExInP,
       tsmEndTxnExOut_t     *tsmEndTxnExOutP
);

extern dsInt16_t DSMLINKAGE  tsmGetData(
       dsUint32_t            tsmHandle,
       DataBlk*dataBlkPtr
);

extern dsInt16_t DSMLINKAGE  tsmGetBufferData(
       getBufferDataIn_t     *tsmGetBufferDataInP,
       getBufferDataOut_t    *tsmGetBufferDataOutP
);

extern dsInt16_t DSMLINKAGE  tsmGetNextQObj(
       dsUint32_t            tsmHandle,
       DataBlk*dataBlkPtr
) ;

extern dsInt16_t DSMLINKAGE  tsmGetObj(
       dsUint32_t            tsmHandle,
       ObjID                 *objIdP,
       DataBlk                *dataBlkPtr
);

extern dsInt16_t DSMLINKAGE  tsmGroupHandler(
       tsmGroupHandlerIn_t   *tsmGroupHandlerInP,
       tsmGroupHandlerOut_t  *tsmGroupHandlerOutP
```

```
);

extern dsInt16_t DSMLINKAGE  tsmInitEx(
       dsUint32_t            *tsmHandleP,
       tsmInitExIn_t         *tsmInitExInP,
       tsmInitExOut_t        *tsmInitExOutP
);


extern dsInt16_t DSMLINKAGE  tsmLogEventEx(
       dsUint32_t            tsmHandle,
       tsmLogExIn_t          *tsmLogExInP,
       tsmLogExOut_t         *tsmLogExOutP
);

extern dsInt16_t DSMLINKAGE  tsmQueryAccess(
       dsUint32_t            tsmHandle,
       tsmQryRespAccessData  **accessListP,
       dsUint16_t            *numberOfRules) ;


extern void DSMLINKAGE       tsmQueryApiVersionEx(
       tsmApiVersionEx       *apiVersionP
);

extern dsInt16_t DSMLINKAGE  tsmQueryCliOptions(
       tsmOptStruct          *optstructP
);

extern dsInt16_t DSMLINKAGE  tsmQuerySessInfo(
       dsUint32_t            tsmHandle,
       tsmApiSessInfo        *SessInfoP
);

extern dsInt16_t DSMLINKAGE  tsmQuerySessOptions(
       dsUint32_t            tsmHandle,
       tsmOptStruct          *optstructP
);

extern dsInt16_t DSMLINKAGE  tsmRCMsg(
       dsUint32_t            tsmHandle,
       dsInt16_t             tsmRC,
       dsChar_t              *msg
);

extern dsInt16_t DSMLINKAGE  tsmRegisterFS(
       dsUint32_t            tsmHandle,
       tsmRegFSData          *regFilespaceP
);

extern dsInt16_t DSMLINKAGE  tsmReleaseBuffer(
       releaseBufferIn_t     *tsmReleaseBufferInP,
       releaseBufferOut_t    *tsmReleaseBufferOutP
);

extern dsInt16_t DSMLINKAGE  tsmRenameObj(
       tsmRenameIn_t         *tsmRenameInP,
       tsmRenameOut_t        *tsmRenameOutP
);

extern dsInt16_t DSMLINKAGE  tsmRequestBuffer(
       requestBufferIn_t     *tsmRequestBufferInP,
       requestBufferOut_t    *tsmRequestBufferOutP
);

extern dsInt16_t DSMLINKAGE   tsmRetentionEvent(
       tsmRetentionEventIn_t  *tsmRetentionEventInP,
       tsmRetentionEventOut_t *tsmRetentionEventOutP
);

extern dsInt16_t DSMLINKAGE   tsmSendBufferData(
       sendBufferDataIn_t     *tsmSendBufferDataInP,
       sendBufferDataOut_t    *tsmSendBufferDataOutP
);

extern dsInt16_t DSMLINKAGE  tsmSendData(
       dsUint32_t            tsmHandle,
       DataBlk                *dataBlkPtr
);

extern dsInt16_t DSMLINKAGE  tsmSendObj(
       dsUint32_t            tsmHandle,
```

```
       tsmSendType           sendType,
       void                  *sendBuff,
       tsmObjName            *objNameP,
       tsmObjAttr            *objAttrPtr,
       DataBlk                *dataBlkPtr
);

extern dsInt16_t DSMLINKAGE  tsmSetAccess(
       dsUint32_t            tsmHandle,
       tsmAccessType         accessType,
       tsmObjName            *objNameP,
       dsChar_t              *node,
       dsChar_t              *owner
);

extern dsInt16_t DSMLINKAGE tsmSetUp(
       dsBool_t             mtFlag,
       tsmEnvSetUp          *envSetUpP
);

extern dsInt16_t DSMLINKAGE  tsmTerminate(
       dsUint32_t            tsmHandle
);

extern dsInt16_t DSMLINKAGE  tsmUpdateFS(
       dsUint32_t            tsmHandle,
       dsChar_t              *fs,
       tsmFSUpd              *fsUpdP,
       dsUint32_t            fsUpdAct
);

extern dsInt16_t DSMLINKAGE  tsmUpdateObj(
       dsUint32_t            tsmHandle,
       tsmSendType           sendType,
       void                  *sendBuff,
       tsmObjName            *objNameP,
       tsmObjAttr            *objAttrPtr,
       dsUint32_t            objUpdAct

);

extern dsInt16_t DSMLINKAGE tsmUpdateObjEx(
       tsmUpdateObjExIn_t         *tsmUpdateObjExInP,
       tsmUpdateObjExOut_t        *tsmUpdateObjExOutP
);


#endif /* ifdef DYNALOAD */

#if defined(__cplusplus)
   }
#endif

#endif /* _H_TSMAPIFP */
```

附錄 D IBM Spectrum Protect 系列產品的協助工具特性

協助工具特性可協助殘障使用者（如行動不方便或是視力受損）順利使用資訊技術內容。

概觀

IBM Spectrum Protect 系列產品包含下列主要協助工具特性：

· 純鍵盤作業

· 使用螢幕閱讀器的作業

IBM Spectrum Protect 系列產品使用最新的 W3C 標準 WAI-ARIA 1.0 (www.w3.org/TR/wai-aria/)，以確保遵守 US Section 508 (www.access-board.gov/guidelines-and-standards/communications-and-it/about-the-section-508-standards/section-508-standards) 和 Web Content Accessibility Guidelines (WCAG) 2.0 (www.w3.org/TR/WCAG20/)。 若要利用協助工具特性，請使用最新版的螢幕閱讀器以及該產品支援的最新 Web 瀏覽器。

IBM Knowledge Center 中的產品說明文件可支援協助工具。 IBM Knowledge Center 的協助工具特性在 IBM Knowledge Center 說明的「協助工具」小節 (www.ibm.com/support/knowledgecenter/about/releasenotes.html?view=kc#accessibility) 中進行了說明。

鍵盤導覽

此產品使用標準導覽鍵。

介面資訊

使用者介面沒有每秒閃動 2 - 55 次的內容。

Web 使用者介面依賴階式樣式表來正確呈現內容並提供可用的體驗。 該應用程式會為弱視使用者提供相當的方式來使用系統顯示設定，包括高對比模式。 您可以透過使用裝置或 Web 瀏覽器設定來控制字型大小。

Web 使用者介面包括 WAI-ARIA 導覽界標，可用來快速導覽至應用程式中的功能區。

供應商軟體

IBM Spectrum Protect 系列產品包含 IBM 授權合約未涵蓋的某些供應商軟體。 IBM 對這些產品的協助工具特性不發表意見。 如需供應商協助工具的資訊，請與供應商聯絡。

相關的協助工具資訊

除了標準的 IBM 服務台及支援網站以外，IBM 還提供 TTY 電話服務，以供聽障或聽力不好的客戶取得銷售及支援服務：

TTY 服務
800-IBM-3383 (800-426-3383)
（北美地區）

如需 IBM 對協助工具所做承諾的相關資訊，請參閱 IBM 協助工具 (www.ibm.com/able)。

注意事項

本資訊係針對 IBM 在美國所提供之產品與服務所開發。 IBM 可能會以其他語言提供此資料。 不過，您可能需要擁有該語言的產品或產品版本副本，才能存取它。

在其他國家，IBM 不見得有提供本文件所提及之各項產品、服務或功能。 請洽詢當地的 IBM 業務代表，以取得當地目前提供的產品和服務之相關資訊。 本文件在提及 IBM 的產品、程式或服務時，不表示或暗示只能使用 IBM 的產品、程式或服務。 只要未侵犯 IBM 之智慧財產權，任何功能相當之產品、程式或服務皆可取代 IBM 之產品、程式或服務。 不過，任何非 IBM 的產品、程式或服務，使用者必須自行負責作業的評估和驗證責任。

本文件所說明之主題內容，IBM 可能擁有其專利或專利申請案。 提供本文件不代表提供這些專利的授權。 您可以書面提出授權查詢，來函請寄到：

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

如果是有關雙位元組 (DBCS) 資訊的授權查詢，請洽詢所在國的 IBM 智慧財產部門，或書面提出授權查詢，來函請寄到：

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION 只依「現狀」提供本出版品，不提供任何明示或默示之保證，其中包括且不限於不侵權、可商用性或特定目的之適用性的隱含保證。 有些轄區在特定交易上，不允許排除明示或暗示的保證，因此，這項聲明不一定適合您。

本資訊中可能會有技術上或排版印刷上的訛誤。 因此，IBM 會定期修訂；並將修訂後的內容納入新版中。 IBM 隨時會改進及/或變更本出版品所提及的產品及/或程式，不另行通知。

本資訊中任何對非 IBM 網站之敘述僅供參考，IBM 對這些網站不提供保證。 這些網站上的內容並非本 IBM 產品內容的一部分，用戶使用這些網站時應自行承擔風險。

IBM 得以各種 IBM 認為適當的方式使用或散布 貴客戶提供的任何資訊，而無需對 貴客戶負責。

想要擁有本程式相關資訊以完成下列目的之本程式被授權人：(i) 在個別建立的程式和其他程式（包括本程式）之間交換資訊，以及 (ii) 交互使用已交換的資訊，應該聯絡：

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

上述資料之取得有其特殊要件，在某些情況下必須付費方得使用。

IBM 基於 IBM 客戶合約、IBM 國際程式授權合約或雙方之任何同等合約的條款，提供本文件所提及的授權程式與其所有適用的授權資料。

本文件中所討論的效能資料為特定作業條件下之衍生。 實際結果可能不同。

本文件所提及之非 IBM 產品資訊，取自產品的供應商，或其公佈聲明或其他公開管道。 IBM 並未測試過這些產品，也無法確認這些非 IBM 產品的執行效能、相容性或任何對產品的其他主張是否完全無誤。 有關非 IBM 產品的功能問題應直接洽詢該產品的供應商。

本資訊含有日常商業運作所用之資料和報告範例。 為了盡可能地加以完整說明，範例中含有個人、公司、品牌及產品的名稱。 所有這些名稱全為虛構，任何與實際商場企業使用的名稱及地址類似之處，純屬巧合。

著作權：

本資訊含有原始語言之範例應用程式，用以說明各作業平台中之程式設計技術。 貴客戶可以為了研發、使用、銷售或散布符合範例應用程式所適用的作業平台之應用程式介面的應用程式，以任何形式複製、修改及散布這些範例程式，不必向 IBM 付費。 這些範例並未在所有情況下完整測試。 故 IBM 不保證或默示保證這些樣本程式之可靠性、服務性或功能。 這些程式範例以「現狀」提供，且無任何保證。 IBM 對因使用這些程式範例而產生的任何損害概不負責。

這些範例程式或任何衍生著作的每一份拷貝或其中任何部分，都必須具有下列著作權聲明：©（您的公司）（年份）。 本程式之若干部分係衍生自 IBM Corp. 的範例程式。 © Copyright IBM Corp. _輸入年份_.

商標

IBM、IBM 標誌和 ibm.com® 是 International Business Machines Corp. 在全球許多適用範圍登錄的商標或註冊商標。 其他產品和服務名稱可能是 IBM 或其他公司的商標。 IBM 商標的最新清單可在 Web 的 "Copyright and trademark information" 中找到，網址為 www.ibm.com/legal/copytrade.shtml。

Adobe 是 Adobe Systems Incorporated 在美國及/或其他國家或地區的註冊商標。

Linear Tape-Open、LTO 和 Ultrium 是 HP、IBM Corp. 和 Quantum 在美國及其他國家或地區的商標。

Intel 及 Itanium 是 Intel Corporation 或其子公司在美國及其他國家的商標或註冊商標。

註冊商標 Linux 的使用與 Linus Torvalds 的獨家被授權人、世界範圍擁有者 Linux Foundation 的轉授權相關。

Microsoft、Windows 及 Windows NT 是 Microsoft Corporation 在美國及/或其他國家或地區的商標。

Java™ 和所有以 Java 為基礎的商標和標誌是 Oracle 及/或其分支機構的商標或註冊商標。

Red Hat®、OpenShift®、Ansible® 及 Ceph® 是 Red Hat, Inc. 或其子公司在美國及其他國家或地區的商標或註冊商標。

UNIX 是 The Open Group 在美國及其他國家的註冊商標。

VMware、VMware vCenter Server 及 VMware vSphere 是 VMware, Inc. 或其子公司在美國及/或其他轄區的註冊商標或商標。

產品說明文件條款

這些出版品的使用許可權，係遵循下列條款而授與。

適用性
這些條款是 IBM 網站的全部使用條款的增補項目。

個人使用
貴客戶可以為了非商務性的私人用途而複製這些出版品，但必須保留所有專利注意事項。 如果沒有 IBM 的明文同意， 貴客戶不能散布、顯示或衍生這些出版品或其中的任何部分。

商業使用
貴客戶可以在企業內複製、散布和顯示這些出版品，但必須保留所有專利注意事項。 如果沒有 IBM 的明文同意，貴客戶不能在您的企業外衍生這些出版品，或複製、散布或顯示這些出版品或其中的任何部分。

權利
除非本許可聲明允許，否則本出版品或任何資訊、資料、軟體或其他智慧財產權所附帶的其他聲明、授權及權限，無論是明示或暗示，皆不具效力。

如 IBM 認為出版品的使用途徑損及 IBM 的利益，或經 IBM 判斷為未適當遵守上述指示時，IBM 保留撤銷本項授權的權利。

除非完全符合所有適當的法律和規章，其中包括所有美國輸出法律和規章，否則，貴客戶不能下載、輸出或再輸出本項資訊。

IBM 不提供這些出版品內容的任何保證。 這些出版品只依「現狀」提供，不含任何明示或暗示的保證，其中包括且不限於可商用性、不侵權和特定目的之適用性的暗示保證。

隱私權條款考量

IBM 軟體產品 - 包括軟體即服務解決方案（軟體供應項目）- 可能會使用 Cookie 或其他技術來收集產品使用資訊，以協助提升一般使用者體驗，自訂與一般使用者的互動或用於其他用途。 在許多情況下，「軟體供應項目」並不會收集個人識別資訊。 部分「軟體供應項目」可協助您收集個人識別資訊。 如果此「軟體供應項目」使用 Cookie 來收集個人識別資訊，下方將規定關於此供應項目使用 Cookie 的特定資訊。

本軟體供應項目不使用 Cookie 或其他技術來收集個人識別資訊。

如果針對此「軟體供應項目」部署的配置可讓貴客戶透過 Cookie 及其他技術來收集一般使用者的個人識別資訊，則貴客戶應該探查有關這類資料收集的任何適用法律的專屬法律建議（包括通知及同意的任何需求）。

如需針對這些用途使用各種技術（包括 Cookie）的相關資訊，請參閱 IBM 的「隱私權條款」(http://www.ibm.com/privacy)、IBM 的「線上隱私權聲明」(http://www.ibm.com/privacy/details)中標題為「Cookie、Web Beacon 和其他技術」的章節以及「IBM 軟體產品和軟體即服務 (Software-as-a-Service) 隱私權聲明」(http://www.ibm.com/software/info/product-privacy)。

名詞解釋

提供了含有 IBM Spectrum Protect 產品系列術語與定義的名詞解釋。

請參閱 IBM Spectrum Protect 名詞解釋。

程式號碼： 5725-W98
5725-W99
5725-X15