IBM Spectrum Protect
8.1.12

使用应用程序编程接口

注：

在使用此信息及其支持的产品前，请阅读第 181 页的『声明』中的信息。

此版本适用于 V8.1.12 的 IBM Spectrum® Protect（产品编号 5725-W98、5725-W99 和 5725-X15）及所有后续发行版和修订版，直到在新版本中另有声明为止。

目录

关于本出版物

本出版物提供的信息有助于执行以下任务：

- 将 IBM Spectrum Protect 应用程序接口调用添加到现存的应用程序
- 使用获取 IBM Spectrum Protect 服务的常规使用程序接口写程序。

除应用程序编程接口 (API) 外，多个操作系统上均包含下列程序：

- 备份－归档客户机程序将文件从工作站或文件服务器备份和归档到存储器，并将文件的备份版本和归档副本恢复和检索到本地文件系统。
- 已授权的管理员、支持人员或最终用户使用 Web 浏览器在远程机器上执行备份、恢复、归档和检索服务的一个 Web 备份－归档客户机。
- 可以从 Web 浏览器或命令行访问的一个管理客户机程序。 管理员控制和监视服务器活动，定义用于备份、归档和空间管理服务的存储管理策略，并设置调度来定期执行这些服务。

本出版物的目标读者

本出版物提供向现有应用程序添加 API 调用的指示信息。您应该熟悉 C 编程语言和 IBM Spectrum Protect 函数。

出版物

IBM Spectrum Protect 产品系列包括 IBM Spectrum Protect Plus、IBM Spectrum Protect for Virtual Environments、IBM Spectrum Protect for Databases 以及若干其他 IBM® 存储管理产品。

要查看 IBM 产品文档，请参阅 IBM Knowledge Center。

本出版物中所使用的约定

本出版物使用以下印刷约定：

示例	描述
autoexec.ncf hsmgui.exe	包含扩展名的一系列小写字母指示程序文件名。
DSMI_DIR	一系列大写字母指示返回码和其他值。
dsmQuerySessInfo	粗体字类型指示在命令行上输入的命令、函数调用的名称、结构的名称、结构中的字段或参数。
timeformat	粗体字斜体类型指示备份/归档客户机选项。粗体类型用于介绍选项或用于示例。
dateformat	斜体类型指示选项、选项的值、新术语、提供的信息的占位符或文本中的特殊强调。
maxcmdretries	等宽字体类型指示程序或信息片段，因为其可能在显示屏幕上显示，例如，命令示例。
加号 (+)	两个键之间的加号指示同时按下两个键。

V8.1.12 中的新增内容

IBM Spectrum Protect V8.1.12 引入新功能和更新。

有关本发行版中的新功能和更新的列表，请参阅 API 更新。

更改左侧的竖线 (|) 指示这是此产品文档中的任何新信息和更改信息。

第 1 章 API 概述

IBM Spectrum Protect 应用程序接口（API）使应用程序客户机能够使用存储管理功能。

API 包括各种函数调用，您可在应用程序中用使用这些函数调用来执行以下操作：

- 开始或结束会话
- 在管理类存储到服务器上之前将其分配到对象
- 将对象备份或归档到服务器
- 从服务器恢复或检索对象
- 从服务器中查询有关存储对象的信息
- 管理文件空间
- 发送保留事件

在您作为应用程序开发者安装 API 时，将收到应用程序的最终用户需要的文件。

- API 共享库。
- 消息文件。
- 样本客户机选项文件。
- 您的应用程序所需的 API 头文件的源代码。
- 样本应用程序的源代码以及用于构建样本应用程序的 Makefile。

对于 64 位应用程序，应使用启用了 64 位支持的编译器选项执行所有编译。 例如，在 AIX® 上构建 API 应用程序时应使用“-q64”，而在 Linux® 上应使用“-m64”。 有关更多信息，请参阅样本 Makefile。

要点: 安装 API 时，请确保所有文件都在相同级别。

有关安装 API 的信息，请参阅安装 IBM Spectrum Protect 备份/归档客户机。

对 UNIX 和 Linux 的引用包括 AIX、HP-UX、Linux、Mac OS X 和 Oracle Solaris 操作系统。

了解配置文件和选项文件

配置文件和选项文件设置会话运行的条件和边界。

您作为管理员或最终用户，可设置选项值以实现以下目的：

- 设置与服务器的连接
- 控制哪些对象发送到服务器以及与其相关联的管理类。

当您在工作站上安装 API 时在一个或两个文件中定义选项。

在 UNIX 和 Linux 操作系统上，选项驻留在两个选项文件中：

- dsm.opt - 客户机选项文件
- dsm.sys - 客户机系统选项文件

在其他操作系统上，客户机选项文件 (dsm.opt) 包含所有选项。

限制: API 不支持以下备份/归档客户机选项：

- `autofsrename`
- `changingretries`
- 域
- `eventlogging`
- 组
- `subdir`

· 用户

· virtualmountpoint

还可以在 **dsmInitEx** 函数调用上指定选项。 使用选项字符串参数或 API 配置文件参数。

相同选项可以派生自多个配置源。 发生这种情况时，具有最高级优先级的源文件优先。 第 2 页的表 1 列出了优先级顺序。

表 *1.* 按降序优先级显示的配置源

优先级	UNIX 和 Linux	Windows	描述
1	dsm.sys 文件 （客户机系统选项）	不适用	此文件包含系统管理员仅为 UNIX 和 Linux 设置的选项。 **提示:** 如果 dsm.sys 文件包含服务器节，请确保 passwordaccess 选项在每个节中指定相同值（prompt 或 generate）。
2	选项字符串 （客户机选项）	选项字符串 （所有选项）	该字符串作为参数传递到 **dsmInitEx** 调用时，其中一个选项会起作用。 该列表可包含客户机选项，如 compressalways、servername（仅 UNIX 和 Linux）或 tcpserveraddr（非 UNIX）。 通过 API 选项字符串，应用程序客户机可以对 API 配置文件和客户机选项文件中的选项值进行更改。 例如，如果要求压缩，那么应用程序可能查询最终用户。 根据用户响应，可以使用该选项构造 API 选项字符串并将其将其传递到对 **dsmInitEx** 的调用。 有关 API 选项字符串格式的信息，请参阅 第 96 页的『dsmInitEx』。 还可以将该参数设置为 NULL。 这表明该会话没有 API 选项字符串。
3	API 配置文件 （客户机选项）	API 配置文件 （所有选项）	在 API 配置文件中设置的值会覆盖在客户机选项文件中设置的值。 使用用户的 IBM Spectrum Protect 会话中合适的值，在 API 配置文件中设置选项。在 API 配置文件名称作为参数传递到 **dsmInitEx** 调用时，该值会起作用。 还可将该参数设置为 NULL。 这表明该会话不存在 API 配置文件。
4	dsm.opt 文件 （客户机选项）	dsm.opt 文件 （所有选项）	在 UNIX 和 Linux 操作系统上，dsm.opt 文件仅包含用户选项。 在其他操作系统上，dsm.opt 文件包含所有选项。 要覆盖这些文件中的选项，请遵循该表中所描述的方法。

相关概念
处理选项

设置 API 环境

API 使用唯一环境变量来查找文件。 可对 API 应用程序使用与备份/归档客户机中所用的文件不同的文件。 应用程序可使用 **dsmSetup** 函数调用来覆盖环境变量设置的值。

提示: 在 Windows 上，缺省安装目录为：`%SystemDrive%\Program Files\Common Files\Tivoli\TSM\api`

第 3 页的表 2 按操作系统列出了 API 环境变量。

表 *2. API* 环境变量

变量	UNIX 和 Linux	Windows
DSMI_CONFIG	客户机选项文件 (dsm.opt) 的标准名称。	客户机选项文件 (dsm.opt) 的标准名称。
DSMI_DIR	指向包含 dsm.sys、en_US 子目录以及任何其他本地语言支持 (NLS) 语言的路径。 en_US 子目录必须包含 dsmclientV3.cat。	指向包含 dscenu.txt 和任何 NLS 消息文件的路径。
DSMI_LOG	指向 dsierror.log 文件的路径。	指向 dsierror.log 文件的路径。 如果设置客户机 **errorlogname** 选项，那么此选项指定的位置将覆盖 DSMI_LOG 所指定的目录。

第 2 章 构建并运行样本 API 应用程序

API 程序包包含演示上下文中的 API 函数调用的样本应用程序。 安装样本应用程序并复审源代码以了解如何使用函数调用。

选择以下某个样本 API 应用程序包：

- 交互式单线程应用程序包 (dapi*)
- 多线程应用程序包 (callmt*)
- 逻辑对象分组测试应用程序 (dsmgrp*)
- 基于事件的保留时间策略样本应用程序 (callevnt)
- 暂时禁用删除样本应用程序 (callhold)
- 数据保留时间保护样本应用程序 (callret)
- IBM Spectrum Protect 数据缓冲区样本程序 (callbuff)

为帮助您入门，请查看按平台构建样本 dapismp 样本应用程序的步骤：

- 对于 UNIX 或 Linux 应用程序，请参阅 第 5 页的『UNIX 或 Linux 样本应用程序源文件』。
- 对于 Windows 应用程序，请参阅第 7 页的『Windows 64 位样本应用程序』。

dapismp 样本应用程序在备份或归档对象时创建自己的数据流。 不会在本地磁盘文件系统读取或写入对象。 对象名不与工作站上的任何文件对应。 您发出的 “seed string” 将生成一个可在恢复或检索对象时进行验证的模式。 编译样本应用程序并运行 **dapismp** 以将其启动后，请遵循屏幕上显示的指示信息。

UNIX 或 Linux 样本应用程序源文件

要构建并运行样本 UNIX 或 Linux 样本应用程序，您需要确保具有特定源文件。构建样本应用程序后，即可编译并运行它。

第 5 页的表 3 中列出的文件包含构建 API 包随附的样本应用程序所需的源文件和其他文件。

表 *3.* 构建 *UNIX* 或 *Linux API* 样本应用程序所需的文件

文件名	描述
README_api_enu	自述文件
dsmrc.h dsmapitd.h dsmapips.h dsmapifp.h release.h	返回码头文件 公共类型定义头文件 特定于操作系统的类型定义头文件 函数原型头文件 版本值头文件
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	命令行驱动的样本应用程序模块
makesmp[64].*xxx*	用于为操作系统构建 dapismp 的 Makefile *xxx* 表示操作系统。
callmt1.c callmt2.c	多线程样本文件

表 3. 构建 *UNIX* 或 *Linux API* 样本应用程序所需的文件 *(续)*

文件名	描述
callmtu1.c callmtu2.c	多线程 Unicode 样本文件
libApiDS.xx libApiDS64.xx 或 libApiTSM64.xx	共享库（后缀与平台相关）。
dsmgrp.c callevnt.c callhold.c callret.c callbuff.c dpsthread.c	分组样本文件 基于事件的保留时间策略样本源代码 暂时禁用删除样本源代码 数据保留时间保护样本源代码

构建 UNIX 或 Linux 样本应用程序

可使用针对您操作系统的编译器来构建 **dapismp** 样本 API 应用程序。

关于此任务

您必须安装以下编译器才能构建 UNIX 或 Linux API 样本应用程序：

· IBM AIX - IBM Visual Age 编译器 V6 或更高版本

· HP-IA64 - aCC 编译器 A.05.50 或更高版本

· Linux - GCC 编译器 V3.3.3 或更高版本

· Mac OS X - GCC 编译器 V4.0 或更高版本

· Oracle Solaris - Oracle Studio C++ 编译器 V11 或更高版本

过程

1. 要构建 API 样本，请运行以下命令：

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

   其中 *xxx* 表示操作系统。
2. 构建样本后，设置环境变量，包括 DSMI_DIR 及选项文件。 有关更多信息，请参阅第 1 页的『了解配置文件和选项文件』。
3. 在您第一次登录时，请以 root 用户身份登录以注册您的密码。

   提示: 将 compressalways 选项设为 no 可能不会重新发送未压缩的对象。 此行为取决于应用程序的功能。

   要在 AIX 上指定共享内存通信方法，IBM Spectrum Protect API 客户机用户必须符合以下某个条件：

   · 必须以 root 用户身份登录。

   · 必须具有与正在运行 IBM Spectrum Protect 服务器的进程相同的 UID。

   有关更多信息，请参阅应用程序文档。
4. 运行 **dapismp** 命令以启动应用程序。
5. 从显示的选项列表中进行选择。 确保首先运行登录操作，然后再运行任何其他操作。

   要求: 在您输入名称时，始终使用正确的路径定界符 (/) 作为文件空间、高级和低级名称的前缀，例如：/myfilespace。 您必须使用此前缀，即使您指定星号 (*) 通配符也是如此。

相关概念

环境变量（UNIX 和 Linux 系统）

Windows 64 位样本应用程序

要构建和运行针对 Microsoft Windows 64 位系统的样本应用程序，您必须安装 IBM Spectrum Protect API 并确保具有特定源文件。

限制:

- 要获得最佳结果，请使用动态装入。 有关示例，请参阅文件 `dynaload.c` 以及样本代码中的实现。
- 样本应用程序的文件位于以下目录中:

  api64\obj
  包含 API 样本程序对象文件。

  api64\samprun
  包含样本程序 **dapismp**。 样本程序包含执行目录。
- DLL tsmapi64.dll 是 64 位 DLL。
- 使用 Microsoft C/C++ Compiler V15 和 makefile `makesmp64.mak` 来编译 API 样本应用程序 **dapismp**。 可能需要调整 makefile 以适合您的环境（具体而言，库或 include 目录）。
- 在您编译应用程序后，通过从 `api64\samprun` 目录发出命令 **dapismp** 来运行样本应用程序。
- 从显示的选项列表中进行选择。 确保首先运行登录操作，然后再运行任何其他操作。
- 在您输入名称时，始终使用正确的路径定界符 (\) 作为文件空间、高级和低级名称的前缀，例如: `\myfilespace`。 您必须使用此前缀，即使您指定星号 (*) 通配符也是如此。

对于 Windows 操作系统，第 7 页的表 4 中列出了要构建样本应用程序而必须具有的源文件。 API 包中包含了样本应用程序。 为了方便起见，还包含了一个预编译的可执行文件 (dapismp.exe)。

表 *4.* 用于构建 *Windows 64* 位 *API* 样本应用程序的文件

文件名	描述
`api.txt`	自述文件
`tsmapi64.dll`	API DLL
`dsmrc.h`	返回码头文件
`dsmapitd.h`	公共类型定义头文件
`dsmapips.h`	特定于操作系统的类型定义头文件
`dsmapifp.h`	函数原型头文件
`dsmapidl.h`	动态装入函数原型头文件
`release.h`	版本值头文件

`dapidata.h`
`dapint64.h`
`dapitype.h`
`dapiutil.h`
	源代码头文件

`tsmapi64.lib`	隐式库

表 4. 用于构建 *Windows 64* 位 *API* 样本应用程序的文件 *(续)*

文件名	描述
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 的源代码文件
makesmpx64.mak (Windows x64) makesmp64.mak (Windows IA64)	用于构建样本应用程序的 Makefile
callmt1.c callmt2.c callmtu164.c callmtu264.c	多线程样本文件
dpsthread.c	样本文件源代码
callevnt.c callhold.c callret.c callbuff.c	基于事件的保留时间策略源代码 暂时禁用删除样本源代码 数据保留时间保护样本源代码 共享缓冲区（非副本）样本源代码。

第 3 章 设计应用程序的注意事项

在设计应用程序时，您必须广泛了解 API 的多个方面。

要了解 API，请查看以下主题：

设计应用程序时，请查看第 9 页的表 5 中的注意事项。 带有 **memset** 字段的开头结构在后续发行版中可能会更改。 `stVersion` 值随每次产品增强而递增。

表 5. 设计应用程序的 *API* 注意事项

设计项	注意事项
设置语言环境	在调用 API 之前，应用程序必须设置语言环境。 要将语言环境设置为缺省值，请向应用程序中添加以下代码： `setlocale(LC_ALL,"");` 要将语言环境设置为其他值，请使用与第二个参数中正确的语言环境相同的调用。 查看文档中有关所使用的各操作系统的特定信息。

表 5. 设计应用程序的 *API* 注意事项 *(续)*

设计项	注意事项
会话控制	将以下准则应用于会话控制： · 为每个 IBM Spectrum Protect 备份/归档客户机和 IBM Spectrum Protect API 客户机产品分配唯一节点名。以下产品是这些客户机的示例： – IBM Spectrum Protect for Mail – 或 IBM Spectrum Protect HSM for Windows · 在整个备份和复原过程中使用一致的所有者名称。 · 使用 passwordaccess 选项来管理对受保护密码文件的访问。 · 确保数据移动的会话在任务完成时结束，以便释放服务器上的设备以供其他会话使用。 · 要准许进行不依赖 LAN 的数据传输，请使用 **dsmSetup** 函数调用并且将 multithread 标志设置为 on。 · 在 AIX 上，当使用多线程应用程序或不依赖 LAN 时（尤其是在具有多个处理器的机器上运行时），在启动应用程序之前，请将环境中的环境变量 AIXTHREAD_SCOPE 设置为 S，以实现更好的性能和更可靠的调度。 例如： `EXPORT AIXTHREAD_SCOPE=S` 通过将 AIXTHREAD_SCOPE 设置为 S，使用缺省属性创建的用户线程将置于系统范围的争用作用域内。 用户线程绑定到内核线程并且由内核进行调度。底层内核线程不与任何其他用户线程进行共享。 有关更多信息，请参阅第 13 页的『使用多线程』。 · 确保在任何时候，会话中仅有一个线程调用任一 API 函数。 使用具有同一会话句柄的多个线程的应用程序必须同步 API 调用。 例如，使用 **mutex** 将 API 调用同步： getTSMMutex() 发出 TSM API 调用 releaseTSMMutex() 仅当线程共享一个句柄时才使用此方法。 您可以对 API 函数使用并行调用，前提是这些调用具有不同会话句柄。
会话控制（继续）	· 实现数据移动的线程化消费者与生产者模型。 API 调用是同步的，在 **dsmGetData 函数**和 **dsmSendData 函数**完成之前，对它们的调用会阻塞。 通过使用消费者与生产者模型，应用程序可能会在网络的等待周期内读取下一个缓冲区。 此外，当存在网络瓶颈或延迟时，对数据读/写和网络的解耦还会提高性能。 通常，以下情况成立： `Data thread <---> shared queue of buffers <---> communication thread (issue calls to the IBM Spectrum Protect API)` · 对多个操作使用同一会话以避免产生开销。 对于处理很多小对象的应用程序，请实现会话池，以便可以在多个小操作之间使用同一会话。 开销与打开和关闭 IBM Spectrum Protect 服务器的会话相关联。 dsmInit/dsmInitEX 调用将进行序列化，因此，即使是在多线程应用程序中，在任何时候也只有一个线程能够登录。 此外，在登录期间，API 会向服务器发送若干一次性查询，以便服务器能够执行所有操作。 这些查询包括策略、选项、文件空间和本地配置。

表 5. 设计应用程序的 *API* 注意事项 *(续)*

设计项	注意事项
操作顺序	IBM Spectrum Protect 服务器在某些操作期间会将文件空间数据库条目锁定。 在您设计 IBM Spectrum Protect API 应用程序时，以下规则适用： · 在整个事务期间，查询会将文件空间锁定。 · 查询锁定可以与其他查询操作共享，以使同一文件空间上的多个查询操作能够并发执行。 · 以下操作用于修改 IBM Spectrum Protect 服务器数据库 (**DB Chg**)：send、get、rename、update 和 delete。 · 完成 **DB Chg** 操作在事务结束时的数据库更改期间需要文件空间锁定。 · 同一文件空间上的多个 **DB Chg** 操作可以并发执行。 序列在事务结束时等待锁定时，可能会有延迟。 · 查询锁定不能与 **DB Chg** 操作共享。 **DB Chg** 操作会延迟在同一文件空间上开始查询的过程，因此请将应用程序设计为将查询与同一文件空间上的 **DB Chg** 操作分离并将查询序列化。
对象命名	在您命名对象时，请考虑以下因素： · 特定对象名是高级和低级对象名。 如果名称中包含唯一标识（如日期戳记），那么备份对象总是处于活动状态。 对象仅在由 **dsmDeleteObj** 函数调用标记为不活动时才会到期。 · 对象的复原方法确定如何对名称进行格式化以轻松进行查询。 如果计划使用部分对象复原 (POR)，那么不能使用压缩。 要禁用压缩，请使用 **dsmSendObj objAttr objCompressed=bTrue** 函数。
对象分组	通过使用文件空间对对象进行逻辑分组。 文件空间是服务器上的容器，用于为对象提供分组类别。 API 在初始登录期间以及在查询期间将查询所有文件空间，因此必须限制文件空间的数量。 合理的假设是，应用程序针对每个节点设置 20 - 100 个文件空间。 API 可以适应更多文件空间，但是每个文件空间会对会话产生开销。 要创建更详细的分离，请在应用程序中使用 `directory` 对象。
对象处理	请勿存储 `objectID` 值以用于将来复原。 不能保证这些值在对象生存期内持久存在。 在复原期间，请特别注意复原顺序。 在查询后，先对该值进行排序，然后再进行复原。 如果要使用多种类型的串行介质，那么在单独会话中访问不同类型的介质。 有关更多信息，请参阅以下主题： 第 54 页的『按恢复顺序选择对象并对其进行排序』
管理类	考虑应用程序必须对与应用程序对象关联的管理类具有多少控制权。 可以定义包含语句，也可以在 **dsmSendObj** 函数调用中指定名称。
对象大小	IBM Spectrum Protect 需要知道每个对象的大小估计。考虑应用程序如何估算对象的大小。 高估对象大小要好于低估对象大小。

确定大小限制

API 中的某些数据结构或字段具有大小限制。 这些结构通常为名称或不能超过预先确定长度的其他文本字段。

以下字段是具有大小限制的数据结构的示例：

· 应用程序类型

· 归档描述

· 副本组目标

· 副本组名称
· 文件空间信息
· 管理类名称
· 对象所有者名称
· 密码

这些限制定义为头文件 dsmapitd.h 中的常量. 任何存储分配都基于这些常量，而非您输入的数字。 有关更多信息，请参阅第 133 页的『附录 B API 类型定义源文件』。

维护 API 版本控制

所有 API 都具有一些形式的版本控制。您在应用程序中使用的 API 版本必须与用户工作站上安装的 API 库版本兼容。

dsmQueryApiVersionEx 应当是您使用 API 时输入的第一个调用。 该调用用于执行以下任务：

· 确认安装了 API 库并且该库在最终用户的系统上可用
· 返回应用程序访问的 API 库的版本级别

API 设计为向上兼容。 在运行较新版本时，写入到较旧版本或发行版的 API 库的应用程序可正常运作。

确定 API 库的发行版非常重要，因为部分发行版可能具有不同的内存要求和数据结构定义。 向下兼容性不大可能。 有关您的平台的信息，请参阅第 12 页的表 6。

表 6. 平台兼容性信息

平台	描述
Windows	消息文件必须与库 (DLL) 在同一级别。
UNIX 或 Linux	API 库和消息文件必须处于同一级别。

dsmQueryApiVersionEx 函数返回最终用户工作站上安装的 API 库的版本。 然后您可以将返回的值与应用程序客户机正在使用的 API 版本进行比较。

应用程序客户机的 API 版本号作为 dsmapitd.h 中定义的四个一组的常量输入到编译的对象代码中。：

```
DSM_API_VERSION
DSM_API_RELEASE
DSM_API_LEVEL
DSM_API_SUB_LEVEL
```

请参阅第 133 页的『附录 B API 类型定义源文件』。

应用程序客户机的 API 版本应低于或等于用户的系统上安装的 API 库版本。 请留意任何其他条件。 可以随时输入 **dsmQueryApiVersionEx** 调用，无论 API 会话是否已开始。

API 使用的数据结构中也包含版本控制信息。 结构将版本信息作为第一个字段。 随着对结构进行增强，版本号也将增加。 初始化版本字段时，请使用 dsmapitd.h 中定义的 structure Version 值。

第 13 页的图 1 演示了头文件 dsmapitd.h 中结构 **dsmApiVersionEx** 的类型定义。该示例随后定义了一个名为 ***apiLibVer*** 的全局变量。 它还演示了如何在调用 **dsmQueryApiVersionEx** 中使用它来返回最终用户 API 库的版本。 最终，返回值与应用程序客户机的 API 版本号进行比较。

```
typedef struct
{
        dsUint16_t   stVersion;                 /* 结构版本                              */
        dsUint16_t version;       /* API 版本                          */
        dsUint16_t release;       /* API 发行版                        */
        dsUint16_t level;         /* API 级别                          */
        dsUint16_t subLevel;      /* API 子级别                        */
} dsmApiVersionEx;

dsmApiVersionEx apiLibVer;

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

/* check for compatibility problems */
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);
```

图 *1.* 获取 *API* 版本级别的示例

使用多线程

多线程化 API 准许应用程序在同一进程中创建多个 IBM Spectrum Protect 服务器会话。 可以再次输入 API。 任何调用均可从不同的线程中并行运行。

提示: 运行采用多线程化 API 的应用程序时，请使用 **dsmQueryAPIVersionEx** 调用。

要在多线程化方式下运行 API，请在 **dsmSetUp** 调用中将 *mtflag* 值设置为 DSM_MULTITHREAD。 **dsmSetUp** 调用必须是 **dsmQueryAPIVersionEx** 调用后的第一个调用。 此调用必须在任何线程调用 **dsmInitEx** 调用之前返回。 所有线程完成处理时，请调用 **dsmCleanUp**。 主进程不应在所有线程完成处理之前结束。 请参阅样本应用程序 中的 callmt1.c。

限制：API 的缺省设置为单线程方式。 如果应用程序不调用 **dsmSetUp**（其中 *mtflag* 值设置为 DSM_MULTITHREAD），那么 API 对于各进程仅准许一个会话。

一旦 **dsmSetUp** 成功完成，应用程序即可开始多个线程并进行多个 **dsmInitEx** 调用。 各 **dsmInitEx** 调用会返回该会话的句柄。 该会话的此线程上的所有后续调用都必须使用该句柄值。 某些值是进程范围的环境变量（在 **dsmSetUp** 中设置的值）。 各 **dsmInitEx** 调用会再次解析选项。 各线程可以通过在 **dsmInitEx** 调用中指定覆盖文件或选项字符串来使用不同选项进行运行。 不同线程借此可以转至不同服务器或使用不同节点名。

建议： 在 HP 上，将线程堆栈设置为 64K 或更大值。 线程堆栈的缺省值 (32K) 可能不够大

要准许应用程序用户具有不依赖 LAN 的会话，请在应用程序中使用 **dsmSetUp** *mtFlag DSM_MULTITHREAD*。 即使应用程序为单线程化，这仍然有必要。 此标志会激活 IBM Spectrum Protect 不依赖 LAN 的接口必需的线程技术。

信号和信号处理程序

应用程序可处理来自用户或操作系统的信号。 如果用户输入 **CTRL+C** 击键序列，那么应用程序必须捕获该信号并为每个活动线程发送 **dsmTerminate** 调用。 然后，调用 **dsmCleanUp** 以退出。 如果会话未正确关闭，那么服务器上可能发生意外结果。

对于导致应用程序结束的信号，应用程序需要信号处理程序，如 SIGPIPE 和 SIGUSR1。 然后应用程序从 API 接收返回码。 例如，要忽略 SIGPIPE，请在您的应用程序中添加以下指令：`signal(SIGPIPE, SIG_IGN)`。 添加此信息后，将返回正确的返回码，而不是应用程序在遇到损坏的管道时退出。

启动或结束会话

IBM Spectrum Protect 是基于会话的产品，且所有活动都必须在 IBM Spectrum Protect 会话中执行。为了启动会话，应用程序会启动 **dsmInitEx** 调用。 除了 **dsmQueryApiVersionEx**、**dsmQueryCliOptions** 和 **dsmSetUp**，必须首先执行该调用才能执行任何其他 API 调用。

dsmQueryCliOptions 函数只能在 **dsmInitEx** 调用之前进行调用。 此函数返回重要选项的值，如选项文件、压缩设置和通信参数。 **dsmInitEx** 调用将设置与服务器的会话，如在调用中传递或选项文件中定义的参数所示。

客户机节点名称、所有者名称和密码参数传送到 **dsmInitEx** 调用。 所有者名称区分大小写，但是节点名称和密码不区分大小写。 在会话开始前，应用程序客户机节点必须注册到服务器。

每次 API 应用程序客户机启动与服务器的会话时，客户机应用程序类型都会注册到服务器。 始终为应用程序类型值指定一个操作系统缩写，因为该值将输入到服务器上的 platform 字段中。 最大字符串长度为 DSM_MAX_PLATFORM_LENGTH。

dsmInitEx 函数调用使用应用程序客户机的 API 配置文件和选项列表建立 IBM Spectrum Protect。 应用程序客户机可使用 API 配置文件和选项列表设置 IBM Spectrum Protect 的很多选项。 这些值将覆盖安装期间在用户配置文件中设置的值。 用户不能更改管理员定义的选项。如果应用程序客户机没有特定配置文件和选项列表，您可以将这两个参数都设置为 NULL。 有关配置文件的更多信息，请参阅以下主题：

第 1 页的『了解配置文件和选项文件』

dsmInitEx 函数调用通过使用允许扩展验证的参数来建立 IBM Spectrum Protect 会话。

检查 **dsmInitEx** 函数调用和 **dsmInitExOut** 信息返回码。 如果返回码正常 (RC=ok) 并且信息返回码 (infoRC) 为 DSM_RC_REJECT_LASTSESS_CANCELED，那么管理员取消了最后一个会话。要立即结束当前会话，请调用 **dsmTerminate**。

dsmQuerySessOptions 调用返回与 **dsmQueryCliOptions** 调用相同的字段。 只能在某个会话中发送此调用。 值反映在该会话期间，选项文件以及对 **dsmInitEx** 调用的任何覆盖中有效的客户机选项。

开始会话后，应用程序可向 **dsmQuerySessInfo** 发送调用以确定为该会话设置的服务器参数。 诸如策略域和事物限制的项将通过该调用返回到应用程序。

通过 **dsmTerminate** 调用结束会话。 与服务器的任何连接都将关闭，并且将释放与此会话相关联的所有资源。

有关启动和结束会话的示例，请参阅以下主题：

第 16 页的图 2

该示例定义了在对 **dsmInitEx** 和 **dsmTerminate** 的调用中使用的一些全局和局部变量。

dsmInitEx 调用采用指向 dsmHandle 的指针作为参数。**dsmTerminate** 调用采用 dsmHandle 作为参数。

第 16 页的图 3 中的示例显示了 **rcApiOut** 的详细信息。 函数 **rcApiOut** 调用 API 函数 **dsmRCMsg**，后者将返回码转换为消息。

然后，**rcApiOut** 调用将为用户打印消息。 **rcApiOut** 版本包含在 API 样本应用程序中。 **dsmApiVersion** 函数是位于头文件 dsmapitd.h 中的类型定义。

会话安全性

基于 IBM Spectrum Protect 会话的系统具有安全性组件，这些组件允许应用程序以安全方式启动会话。这些安全措施禁止对服务器进行未经授权的访问并可帮助确保系统完整性。

每一个随服务器一起启动的会话都必须完成注册流程，请求密码。 当密码与客户机的节点名耦合时，可保证连接到服务器时具有正确的授权。 应用程序客户机将此密码提供给 API 以开始会话。

有两种密码处理方法：*passwordaccess=prompt* 或 *passwordaccess=generate*。 如果使用 *passwordaccess=prompt* 选项，那么必须在每个 **dsmInitEx** 调用上包含密码值。 或者可以在 **dsmInitEx** 调用上应用节点名和所有者名称。

密码具有与其相关联的到期时间。 如果某个 **dsmInitEx** 调用失败，且密码过期返回码为 (DSM_RC_REJECT_VERIFIER_EXPIRED)，那么应用程序客户机必须使用 **dsmInitEx** 返回的句柄输入 **dsmChangePW** 调用。 这将在成功建立会话前更新密码。 第 17 页的图 4 的示例演示了使用 **dsmChangePW** 更改密码的过程。 登录所有者必须使用 root 用户标识或授权用户标识以更改密码。

第二种方法，*passwordaccess=generate*，对密码值进行加密并将其存储到文件中。 节点名和所有者名称不能在 **dsmInitEx** 调用上提供，但可以使用系统缺省值。 该方法能保护密码文件的安全。 密码到期时，generate 参数会创建一个新密码并自动更新密码文件。

提示:

1. 如果两个不同的物理机器具有相同的 IBM Spectrum Protect 节点名，或者是使用多个服务器节在一个节点上定义了多个路径，那么 *passwordaccess=generate* 可能仅对密码到期后使用的第一个节有效。 第一次客户机服务器连接期间，会提示用户分别对每个服务器节和每个节使用相同密码，并且对于每个节，密码副本将单独存储。 密码到期时，会为连接第一个客户机/服务器连接的节生成密码。 通过其他服务器节进行连接的所有后续尝试均将失败，因为旧密码各自的副本与密码过期后由首个使用的节生成的更新副本之间没有逻辑链路。 在这种情况下，您必须在到期前更新密码，或在到期后，作为对该情况的补救措施而更新密码，如下所示：
   a. 运行 **dsmadmc** 并更新服务器上的密码。
   b. 运行 **dsmc -servername=stanza1** 并使用新密码生成正确条目。
   c. 运行 **dsmc -servername=stanza2** 并使用新密码生成正确条目。
2. 对于 UNIX 或 Linux：仅 root 用户或授权用户可可以在使用 *passwordaccess=prompt* 时更改密码。仅 root 用户或授权用户可以在使用 *passwordaccess=generate* 时启动密码文件。

   限制: 无法识别选项 users 和 groups。

应用程序通过其他方法限制用户访问权限，如设置访问权限过滤器。

如果应用程序使用指向单个 IBM Spectrum Protect 服务器的多个 IP 连接，则应对每个会话使用相同节点名和 IBM Spectrum Protect 客户机密码。执行以下步骤来启用此支持：

1. 在 dsm.sys 文件中定义 IBM Spectrum Protect 服务器节.
2. 对于不使用缺省 IP 地址的连接，为 *TCPserver* 地址指定选项值 和 *TCPport* 在 **dsmInitEx** 调用上。

这些值会覆盖 IP 连接信息，但是会话仍使用相同 dsm.sys 节节点和密码信息。

注: 集群中的节点共享一个密码。

```
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; /* Set the applications compile */
apiApplVer.release = DSM_API_RELEASE;  /* time version.         */
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);   /* Call function to print error message */
   dsmTerminate(dsmHandle);    /* clean up memory blocks */
   return(rc);
}
```

图 *2.* 开始和结束会话的示例

```
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;
}
```

图 *3. Details of rcApiOut*

```
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);
/* If new password entries don't match, try again or exit. */
/* If they do match, call 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;
```

图 *4. An example of changing a password*

控制对密码文件的访问

要在 UNIX 和 Linux 系统上控制对受保护的密码文件的访问，您可以以授权用户身份登录，并将 passwordaccess 选项设置为 generate。

过程

在将 passwordaccess 设置为 generate 时，完成以下步骤：

1. 通过调用 **dsmSetUp**（用于传递 *argv[0]*）编写应用程序。 *argv[0]* 包含调用 API 的应用程序的名称。 允许应用程序由授权用户运行；但是，管理员必须决定授权用户的登录名。
2. 将应用程序可执行文件的有效用户标识位（S 位）设置为 On。 然后，应用程序可执行文件的所有者可以成为授权用户，并且可创建密码文件、更新密码以及运行应用程序。应用程序可执行文件的所有者与运行程序的用户标识必须相同。在以下示例中，*User* 为 user1，应用程序可执行文件的名称为 applA，并且 user1 具有 /home/user1 目录的读/写许可权。applA 可执行文件具有以下许可权：

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

3. 指示应用程序用户使用授权用户名称登录。IBM Spectrum Protect 首先确定登录标识与应用程序可执行文件所有者匹配，然后才允许访问受保护的密码文件。
4. 设置 passworddir 选项（dsm.sys 文件中）以指向此用户具有读/写访问权的目录。
   例如，在 dsm.sys 文件的服务器节中输入以下行：

   ```
   passworddir /home/user1
   ```

5. 创建密码文件并确保授权用户拥有该文件。
6. 以 user1 身份登录并运行 app1A。
7. 调用 **dsmSetUp** 并传递 *argv*。

使用客户机所有者权限创建管理用户

具有客户机所有者权限的管理用户可设置 **dsmInitEx** 函数调用中的参数来开始会话。 此用户可以作为“administrative user”，带有针对被拒绝节点的备份和恢复权限。

过程

要接收客户机所有者权限，请在服务器上完成以下步骤：

1. 定义管理用户：

   ```
   REGister Admin admin_name password
   ```

其中：

- *admin_name* 是管理用户名。
- *password* 是管理密码。

2. 定义权限级别。 具有系统或策略权限的用户也具有客户机所有者权限。

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

其中：

- *admin_name* 是管理用户。
- *classes* 是节点。
- *authority* 具有以下一个级别的权限：
  - `owner`：节点的完全备份和复原权限
  - `node`：单个节点
  - `domain`：节点的组

3. 定义对单个节点的访问权。

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

其中：

- *node_name* 是客户机用户节点
- *password* 是客户机用户节点密码
- *user_id* 是管理用户名

结果

在应用程序使用管理用户时，将使用 `userName` 和 `userNamePswd` 参数调用 **dsmInitEx** 函数。

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

您可以将 `passwordaccess` 选项设置为 `generate` 或 `prompt`。 通过以上任一参数，`userNamePswd` 值都会开始此会话。 此会话启动后，即可针对该节点进行任何备份或复原过程。

对象名和标识

IBM Spectrum Protect 服务器是一种对象存储服务器，其主要功能在于高效存储和检索指定对象。 对象标识对于各对象是唯一的，并且会在对象生存期内随对象保留，但是在使用导出或导入时除外。

为符合此需求，IBM Spectrum Protect 含有两个主要存储区域：数据库和数据存储器。

- 数据库包含所有元数据，如与对象关联的名称或属性。
- 数据存储器包含对象数据。 数据存储器实际上是系统管理员定义的存储层次结构。 根据成本和访问需求，会在联机或脱机介质上对数据进行高效存储和管理。

服务器上存储的各对象均有与其关联的名称。 客户机控制该名称的下列关键组件：

- 文件空间名称
- 高级名称
- 低级名称
- 对象类型

在决定对应用程序的对象进行命名时，可能需要对最终用户的完整对象名使用外部名。 特别是，最终用户可能需要应用程序运行时包含或排除语句中的对象。 这些语句中的对象名的确切语法与平台有关。 在 Windows 操作系统上，包含或排除语句中使用的是与文件空间关联的盘符而不是文件空间名称本身。

创建对象时与执行复原过程时所赋的对象标识值可能不同。 应用程序应保存对象名，然后查询以获取当前对象标识，然后再执行复原。

文件空间名称

文件空间名称是最重要的存储组件之一。 它可以是文件系统、磁盘驱动器的名称或用于将相关数据分组在一起的任何其他高级限定符。

IBM Spectrum Protect 使用文件空间识别数据所在文件系统或磁盘驱动器。 通过此方法，可以在文件空间中的所有实体上执行操作，如查询指定文件空间中的所有对象。 由于文件空间是 IBM Spectrum Protect 命名约定的一个如此重要的组件，因此使用特殊调用来注册、更新、查询和删除文件空间。

服务器还具有管理命令，用于查询 IBM Spectrum Protect 存储器中任何节点上的文件空间，并在必要时将其删除。 应用程序客户机存储的所有数据都必须具有与该应用程序客户机关联的文件空间名称。 请仔细选择名称，以在系统中将类似数据分组在一起。

为避免可能的干扰，应用程序客户机应选择与备份/归档客户机所用不同的文件空间名称。 如有必要，应用程序客户机应发布其文件空间名称，以便最终用户能够识别包含/排除语句的对象。

注: 在 Windows 平台上，盘符与文件空间相关联。 注册或更新文件空间时，必须提供盘符。 由于包含/排除列表会引用盘符，因此必须记录各盘符及其关联的文件空间。 在样本程序 dapismp 中，盘符缺省设置为“G”。

请参阅第 5 页的『第 2 章 构建并运行样本 API 应用程序』，以获取有关样本程序的更多信息。

高级名称和低级名称

对象名的其他两个组件为高级名称限定符和低级名称限定符。 高级名称限定符是对象所属的目录路径，低级名称限定符是该目录路径中对象的实际名称。

当并置文件空间名称、高级名称和低级名称时，这些名称必须在客户机运行所在的操作系统上组成语法正确的名称。 该名称不必作为系统上的对象存在，也不必与本地文件系统上实际数据类似。 不过，名称必须符合标准命名规则，才能由 **dsmBindMC** 调用进行正确处理。 请参阅第 34 页的『了解备份和归档对象』，以了解与策略管理相关的命名注意事项。

对象类型

对象类型将对象识别为文件或目录。 文件是指同时包含属性和二进制数据的对象，而目录是指仅包含属性的对象。

第 19 页的表 7 按平台显示了应用程序客户机对对象名的编码方式。

表 7. 应用程序对象名称示例（按平台）

平台	对象名的客户机代码
UNIX 或 Linux	/myfs/highlev/lowlev
Windows	"myvol\\highlev\\lowlev" **注:** 在 Windows 平台上，双反斜杠将转换为单个反斜杠，因为反斜杠是转义字符。 在 UNIX 或 Linux 平台上，文件空间名称以斜杠开始，但在 Windows 平台上不以斜杠开始。

以会话所有者身份访问对象

每个对象都具有与其关联的所有者名称。 用于确定所访问的对象的规则取决于启动会话时使用的所有者名称。 使用此会话所有者值可以控制对对象的访问权。

会话所有者是在 **dsmInitEx** 调用期间在 *clientOwnerNameP* 参数中设置的。 如果启动 **dsmInitEx** 所有者名称为 *NULL* 的会话并且使用 *passwordaccess=prompt*，那么会通过会话权限（root 或授权用户）对该会话所有者进行处理。 如果使用 root 用户标识或授权用户标识登录并且使用 *passwordaccess=generate*，那么此情况也适用。在以该方式启动的会话期间，您可以对此节点拥有的任何对象执行任何操作，而不考虑该对象的实际所有者。

如果使用特定所有者名称启动会话，那么此会话只能对与该对象所有者名称关联的对象执行操作。 到系统中的所有备份或归档操作都必须与该所有者名称关联。 所执行的任何查询仅会返回与此所有者名称关联的值。对象所有者值是在 **dsmSendObj** 调用期间在 **ObjAttr** 结构的 **Owner** 字段中设置的。 所有者名称区分大小写。 第 20 页的表 8 概括了用户有权访问对象的条件。

表 *8.* 对对象的用户访问权摘要

会话所有者	对象所有者	用户访问权
NULL（root 和系统所有者）	“ ”（空字符串）	是
空	特定名称	是
特定名称	“ ”（空字符串）	否
特定名称	相同名称	是
特定名称	不同名称	否

跨节点和所有者访问对象

三种函数调用支持在同一平台上跨节点、跨所有者进行访问：**dsmSetAccess**、**dsmDeleteAccess** 和 **dsmQueryAccess**。 这些函数以及在 **dsmInitEx** 上传递的 *-fromnode* 和 *-fromowner* 字符串选项准许通过 API 进行完整的跨节点查询、复原和检索过程。

例如，节点 A 上的用户 A 使用 **dsmSetAccess** 函数调用将对 /db 文件空间下其备份的访问权提供给来自节点 B 的用户 B。访问规则显示如下：

ID	类型	节点	用户	路径
1	备份	节点 B	用户 B	/db/*/*

当用户 B 在节点 B 上登录时，**dsmInitEx** 的选项字符串为：

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

这些选项针对此会话进行设置。 任何查询均会访问节点 A 的文件空间和文件。不准许使用备份和归档。 仅准许在用户 B 具有访问权的文件空间中进行查询、复原和检索过程。 在设置了 *-fromnode* 或 *-fromowner* 选项的情况下登录时，如果应用程序尝试使用 **dsmBeginTxn** 执行任何操作（例如备份或更新），那么 **dsmBeginTxn** 将失败，返回码为 DSM_RC_ABORT_NODE_NOT_AUTHORIZED。 请参阅单独函数调用和 第 96 页的『dsmInitEx』，以获取更多信息。

提示: 在 UNIX 和 Linux 上，可以在 **dsmInitEx** 函数调用中传递的选项字符串中指定 *–fromowner=root*。如果已执行 set access，那么这准许非 root 用户对 root 拥有的文件进行访问。

在带有相应函数的 **dsmInitEx** 选项字符串上使用 *asnodename* 选项可以备份、归档、复原、检索、查询或删除 IBM Spectrum Protect 服务器上目标节点名下的数据。 请参阅第 66 页的『使用客户机节点代理支持备份多个节点』，以获取有关启用此选项的信息。

管理文件空间

由于文件空间对于系统的运行很重要，因此会使用一组单独的调用来注册、更新和删除文件空间标识。 必须首先向 IBM Spectrum Protect 注册系统上的文件空间，然后才能存储与该文件空间关联的任何对象。

使用 **dsmRegisterFS** 调用可以完成此任务。 有关对象名和标识的更多信息，请参阅第 18 页的『对象名和标识』。

文件空间标识是三部分名称层次结构中的顶级限定符。 通过在文件空间中将相关数据分组在一起，可以更轻松地管理该数据。 例如，应用程序客户机或 IBM Spectrum Protect 服务器管理员可以删除文件空间及该文件空间中的所有对象。

文件空间还准许应用程序客户机向服务器提供有关文件空间的信息，管理员之后可以查询该信息。在 **qryRespFSData** 结构中进行查询时会返回此信息，其中包含以下文件系统信息：

类型	定义
fstype	文件空间类型。 此字段是应用程序客户机设置的字符串。
fsAttr[platform].fsInfo	用于特定于客户机的数据的客户机信息字段。
容量	文件空间中的总空间量。
占用率	当前在文件空间中占用的空间量。
backStartDate	最新备份启动时的时间戳记（通过发送 **dsmUpdateFS** 调用进行设置）。
backCompleteDate	最新备份完成时的时间戳记（通过发送 **dsmUpdateFS** 调用进行设置）。

使用 capacity 和 occupancy 取决于应用程序客户机。 一些应用程序可能不需要有关文件空间大小的信息，在此情况下，这些字段可以缺省为 0。有关查询文件空间的更多信息，请参阅第 26 页的『查询 IBM Spectrum Protect 系统』。

向系统注册文件空间后，可以随时备份或归档对象。 要在备份或归档操作之后更新文件空间的 occupancy 和 capacity 字段，请调用 **dsmUpdateFS**。 此调用确保文件系统的 occupancy 和 capacity 值是最新的。 您也可以更新 **fsinfo**、**backupstart** 和 **backupcomplete** 字段。

如果要监视最后备份日期，请在启动备份之前进行 **dsmUpdateFS** 调用。 将更新操作设置为 DSM_FSUPD_BACKSTARTDATE。 这会强制服务器将文件空间的 **backStartDate** 字段设置为当前时间。 完成该文件空间的备份后，在更新操作设置为 DSM_FSUPD_BACKCOMPLETEDATE 的情况下进行 **dsmUpdateFS** 调用。 此调用会在备份结束时创建时间戳记。

如果不再需要文件空间，那么可以使用 **dsmDeleteFS** 命令将其删除。 在 UNIX 或 Linux 操作系统上，仅 root 用户或授权用户可删除文件空间。

第 22 页的图 5 中的示例说明了如何对 UNIX 或 Linux 使用三个文件空间调用。 有关如何使用 Windows 的三个文件空间调用的示例，请参阅系统上安装的样本程序代码。

```
/*  Register the file space if it has not already been done.  */

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;  /* already done */
if (rc)
{
   printf("Filespace registration failed: ");
   rcApiOut(dsmHandle, rc);
   free(bkup_buff);
   return (RC_SESSION_FAILED);
}
```

图 *5.* 使用文件空间的示例（第一部分）

```
/* Update the file space. */

dsmFSUpd     updFilespace;         /* for update FS */

updFilespace.stVersion = dsmFSUpdVersion;
updFilespace.fsType = 0;             /* no change */
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);
```

图 *6.* 使用文件空间的示例（第二部分）

```
/*  Delete the file space.  */

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

图 *7.* 使用文件空间的示例（第三部分）

将对象与管理类相关联

IBM Spectrum Protect 的主功能是使用策略（管理类）来定义如何在 IBM Spectrum Protect 存储器中存储和管理对象。 在对象已备份或归档时，此对象与管理类相关联。

此管理类确定：

· 在备份情况下保留的对象版本数

· 保存归档副本的时间

· 对象在服务器上的存储器层次结构中的插入位置

管理类由备份副本组和归档副本组两者组成。 副本组是指用于为所备份或归档的对象定义管理策略的属性集。 如果正在执行备份操作，那么应用备份副本组中的属性。 如果执行的是归档操作，那么归档副本组中的属性适用。

特定管理类中的备份或归档副本组可以为空或 NULL。 如果对象绑定到 NULL 备份副本组，那么无法备份该对象。 如果对象绑定到 NULL 归档副本组，那么无法归档该对象。

由于策略的使用是 IBM Spectrum Protect 中非常重要的组成部分，因此 API 要求通过使用 **dsmBindMC** 调用首先向已发送到服务器的所有对象指派管理类。 通过 IBM Spectrum Protect 软件，您可以使用包含/排除列表来影响管理类绑定。 **dsmBindMC** 调用使用当前包含/排除列表来执行管理类绑定。

包含语句可以将特定管理类与备份对象或归档对象相关联。 排除语句可以防止对象进行备份，但不防止其进行归档。

API 要求在备份或归档对象之前调用 **dsmBindMC**。 **dsmBindMC** 调用会返回 mcBindKey 结构，其中包含有关与对象关联的管理类或副本组的信息。 请检查副本组目标，然后再继续执行发送操作。 在单个事务中发送多个对象时，这些对象必须具有同一副本组目标。 **dsmBindMC** 函数调用会返回以下信息：

表 *9. dsmBindMC* 调用时返回的信息

信息	描述
管理类	绑定到对象的管理类的名称。 应用程序客户机可以发送 **dsmBeginQuery** 调用以确定此管理类的所有属性。
备份副本组（Backup Copy Group）	通知您是否存在此管理类的备份副本组。 如果正在执行备份操作并且备份副本组不存在，那么无法将此对象发送到存储器。如果已尝试使用 **dsmSendObj** 调用来发送此对象，那么会收到错误代码。
备份副本目标	此字段用于识别数据发送到的存储池。如果执行的是多对象备份事务，那么该事务中的所有副本目标都必须相同。 如果某个对象与事务中的先前对象具有不同的副本目标，请结束当前事务并开始新事务，然后才能发送此对象。 如果尝试将对象发送到同一事务中的不同副本目标，那么会收到错误代码。
归档副本组（Archive Copy Group）	通知您是否存在此管理类的归档副本组。 如果正在执行归档操作并且归档副本组不存在，那么无法将此对象发送到存储器。如果已尝试使用 **dsmSendObj** 调用来发送此对象，那么会收到错误代码。
归档副本目标	此字段用于识别数据发送到的存储池。如果执行的是多对象归档事务，那么该事务中的所有副本目标都必须相同。 如果某个对象与事务中的先前对象具有不同的副本目标，请结束当前事务并开始新事务，然后再发送此对象。 如果尝试将对象发送到同一事务中的不同副本目标，那么会收到错误代码。

如果使用与原始管理类不同的管理类通过同一对象名进行了后续备份，那么可以将对象的备份副本重新绑定到其他管理类。 例如，如果备份 ObjectA 并将其绑定到 Mgmtclass1，之后备份 ObjectA 并将其绑定到 Mgmtclass2，那么最新备份会将任何不活动副本重新绑定到 Mgmtclass2。 Mgmtclass2 中定义的参数此时会控制所有副本。 不过，如果目标不同，那么数据不移动。

您也可以将 **dsmUpdateObj** 或 **dsmUpdateObjEx** 调用与 DSM_BACKUPD_MC 操作配合使用来将备份副本重新绑定到其他管理类。

相关参考
重复数据删除选项

查询管理类

应用程序可以查询管理类，以确定对于给定节点适用的管理类，以及确定管理类中的属性。

只能通过使用 **dsmBindMC** 调用来将对象绑定到管理类。 您可能希望应用程序查询管理类属性并向最终用户显示这些属性。 请参阅第 26 页的『查询 IBM Spectrum Protect 系统』以获取更多信息。

在 第 24 页的图 8 内的示例中，当调用 **dsmBindMC** 时，使用了开关语句来区分备份操作和归档操作。 从此调用返回的信息存储在 **MCBindKey** 结构中。

```
dsUint16_t     send_type;
dsUint32_t     dsmHandle;
dsmObjName   objName;     /* structure containing the object name  */
mcBindKey    MCBindKey;   /* management class information          */
char         *dest;       /* save destination value                */

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;
}
```

图 *8.* 将管理类与对象相关联的示例

免到期/删除和释放

您可以保留特定归档对象的删除和到期以响应需要保留特定数据的暂挂或持续进行的操作。如果启动操作可能需要访问数据，那么，在操作结束前该数据必须为可用的，且对数据的访问不再需要作为该过程的一部分。 确定不再需要暂挂（已释放）后，正常删除和到期计时会按照原始保留期恢复。

开始之前

通过发出 **dsmRetentionEvent** 调用来验证服务器是否已获得许可：

1. 针对想要保留并获取标识的一个对象的查询。
2. 发出 **dsmBeginTxn**、带有 Hold 的 **dsmRetentionEvent** 及 **dsmEndTxn**。
3. 如果服务器未获得许可，那么将收到异常中止表决，原因码为 DSM_RC_ABORT_LICENSE_VIOLATION。

限制:

1. 在单个事务中无法发出一个以上的 **dsmRetentionEvent** 调用。
2. 无法对已保留的对象发出保留命令。

过程

1. 要保留对象，请完成以下步骤：

a) 查询服务器中希望保留的所有对象。 获取每个对象的对象标识。

b) 发出 **dsmBeginTxn** 调用，然后对对象列表发出 **dsmRetentionEvent** 调用，接着是 **dsmEventType**:eventHoldObj 调用。 如果对象数超出 maxObjPerTxn 的值，请使用多个事务。

c) 使用 qryRespArchiveData **dsmGetNextQObj** 上的响应 函数调用来确认对象已保留。检查 qryRespArchiveData 中 objHeld 的值。

2. 要释放对象保留，请完成以下步骤：

a) 查询服务器中希望将其从保留状态释放出来的所有对象。 获取每个对象的对象标识。

b) 发出 **dsmBeginTxn** 调用，然后对对象列表发出 **dsmRetentionEvent** 调用，接着是 **dsmEventType**:eventReleaseObj 调用。 如果对象数超出 maxObjPerTxn 的值，请使用多个事务。

c) 在 dsmGetNextQObj 函数调用上使用 **qryRespArchiveData** 响应以确认是否已释放对象保留。检查 qryRespArchiveData 中 **objHeld** 的值。

归档数据保留保护

未经授权的代理程序无法修改 IBM Spectrum Protect 控制下的数据，例如，个人或程序。此保护扩展为在保留期到期前可阻止任何代理程序删除数据，例如，归档对象。

关于此任务

保护归档保留帮助确保没有个人或程序故意或意外删除 IBM Spectrum Protect 控制下的数据。发送到归档保留保护服务器的归档对象将受保护免于意外删除，并且具有强制实施的保留期。归档保留保护具有以下限制：

- 保留保护服务器上仅允许归档操作。
- 未通过 **dsmBindMc** 函数调用中的值或包含/排除语句明确绑定到管理类的对象与缺省管理类的名称明确绑定。 例如，如果节点策略中的缺省管理类为 MC1，那么对象明确绑定到 MC1，而不是 DEFAULT。在查询响应中，该对象将显示为与 MC1 绑定。
- 启用归档数据保留保护后，在保留期到期前删除对象的任何尝试都将在结束事务上返回代码 DSM_RC_ABORT_DELETE_NOT_ALLOWED。

请参阅 IBM Spectrum Protect 服务器文档以获取有关为归档对象设置保留保护的指示信息。

过程

要设置归档数据保留保护，请完成以下步骤：

1. 在无先前数据的新服务器安装上，运行 **SET ARCHIVERETENTIONPROTECTION ON** 命令。
2. 在 **dsmInit** 或 **dsmInitEx** 函数调用的 API 选项字符串上，输入以下指示信息：

```
-ENABLEARCHIVERETENTIONPROTECTION=yes
```

设置 enablearchiveretentionprotection 选项： 非 UNIX 系统上的 dsm.opt 文件或 UNIX 系统上的 dsm.sys 文件中的选项：

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

有关此选项的更多信息，请参阅第 26 页的『enablearchiveretentionprotection 选项』。

3. 向服务器发出查询以确认 IBM Spectrum Protect 服务器是否启用归档保留保护。检查 dsmQuerySessInfo 结构中 archiveRetentionProtection 字段的值。

enablearchiveretentionprotection 选项

enablearchiveretentionprotection 选项指定是否在专用于数据保留保护的 IBM Spectrum Protect 服务器上启用归档对象的数据保留保护。服务器管理员必须在尚无存储对象（备份、归档或空间管理的对象）的新服务器上激活数据保留时间保护。如果 API 应用程序尝试在服务器上存储备份版本或空间管理的对象，那么将发出错误消息。

第 9 页的『第 3 章 设计应用程序的注意事项』 状态注释：“请勿存储 objectID 值以用于将来复原。不能保证这些值在对象生存期内持久存在。”由于归档管理器服务器不支持导出或导入，因此，归档管理器应用程序可能非常空闲。 归档管理器应用程序可以保存和使用 objectID 来在对象复原期间提高性能。

如果服务器发出 **SET ARCHIVERETENTIONPROTECTION ON** 命令，那么无法使用 **delete filespace** 命令从服务器删除归档对象，直至归档副本组的策略参数得到满足为止。请参阅相应的服务器文档以获取有关如何设置管理类的信息。

基于事件的保留保护

在基于事件的保留策略中，归档对象的保留时间由业务事件启动，例如，关闭银行帐户。基于事件的保留将 IBM Spectrum Protect 数据保留策略与数据的业务需求密切结合。在事件发生时，应用程序将此对象的 **eventRetentionActivate** 事件发送到服务器以启动保留。

过程

要使用基于事件的保留策略，请完成以下步骤：

1. 在服务器上，使用类型为 EVENT 的归档 **copygroup** 创建管理类。有关更多信息，请参阅 IBM Spectrum Protect 服务器文档。
2. 查询管理类以确认此类是基于事件。如果管理类是基于事件，那么 **archDetailCG** 结构中的 **retainInit** 字段为 ARCH_RETINIT_EVENT。
3. 使用包含、**archmc** 或明确通过 **dsmSendObj** 函数调用中的 **ObjAttr** 结构的 **mcNameP** 属性将对象与基于事件的管理类绑定。
4. 想要启动对象保留时，查询服务器以查找受影响的所有对象。 检查以查看它们是否处于 PENDING 状态，并获取对象标识。在暂挂状态下，**qryRespArchiveData** 结构中的 **retentionInitiated** 字段指示 DSM_ARCH_RETINIT_PENDING。
5. 发出 **dsmBeginTxn** 调用，然后对对象列表发出 **dsmRetentionEvent** 调用，接着进行 **dsmEventType**:eventRetentionActivate 调用。 如果对象数超出 maxObjPerTxn 的值，请使用多个事务。

   限制: 每个事务只能发出一个 **dsmRetentionEvent** 调用。
6. 查询对象以确认保留是否已激活。 如果保留已启动，那么 **qryRespArchiveData** 结构中的 **retentionInitiated** 字段具有值 I。

查询 IBM Spectrum Protect 系统

API 有多个应用程序可以使用的查询，如管理类查询。

过程

使用 **dsmBeginQuery** 调用的所有查询遵循这些步骤：

1. 使用相应的查询类型发送 **dsmBeginQuery** 调用：
   - 备份
   - 归档
   - 活动备份对象
   - 文件空间
   - 管理类

dsmBeginQuery 调用可向 API 通知从服务器返回的数据格式。可将相应的字段放在通过 **dsmGetNextQObj** 调用所传递的数据结构中。 开始查询调用还允许应用程序客户机通过正确指定开始查询调用上的参数来设置查询范围。

限制: 在 UNIX 或 Linux 系统上，仅 root 用户可查询活动的已备份对象。 此查询类型被称为“快速路径”。

2. 输入 **dsmGetNextQObj** 调用以获取该查询中的每个记录。 此调用可传递足以保留从该查询返回的数据的缓冲区。 针对返回的数据，每个查询类型都有一个相应的数据结构。 例如，备份查询类型具有关联的 **qryRespBackupData** 结构，在发送 **dsmGetNextQObj** 调用时填充此结构。
3. **dsmGetNextQObj** 调用通常返回以下代码的其中之一：
   - DSM_RC_MORE_DATA：再次发送 **dsmGetNextQObj** 调用。
   - DSM_RC_FINISHED：无更多数据。发送 **dsmEndQuery** 调用。
4. 发送 **dsmEndQuery** 调用。 在检索所有查询数据或无需更多查询数据时，输入 **dsmEndQuery** 调用以结束查询过程。API 清空查询流中的任何剩余数据，并释放用于查询的任何资源。

结果

第 27 页的图 9 显示查询操作的状态图。

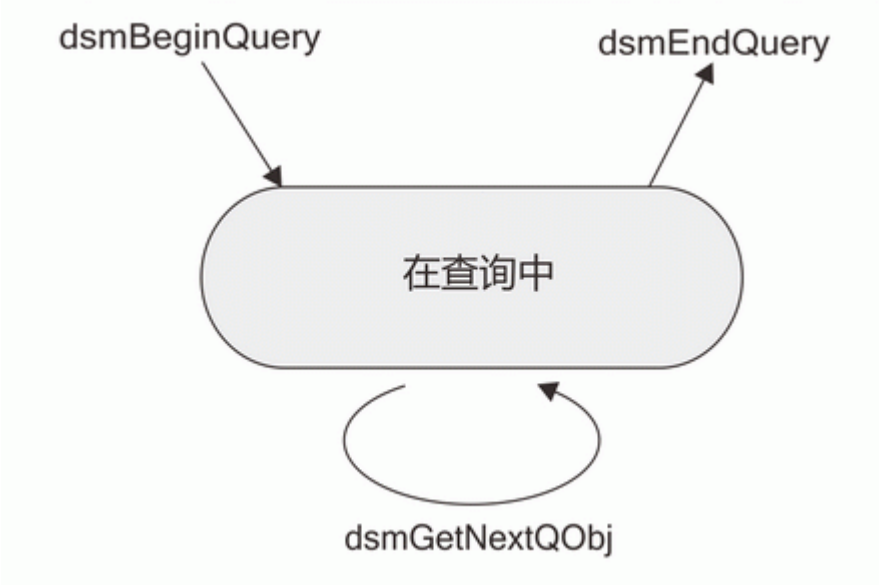


图 9. 常规查询的状态图。

第 28 页的图 10 显示查询操作的流程图。

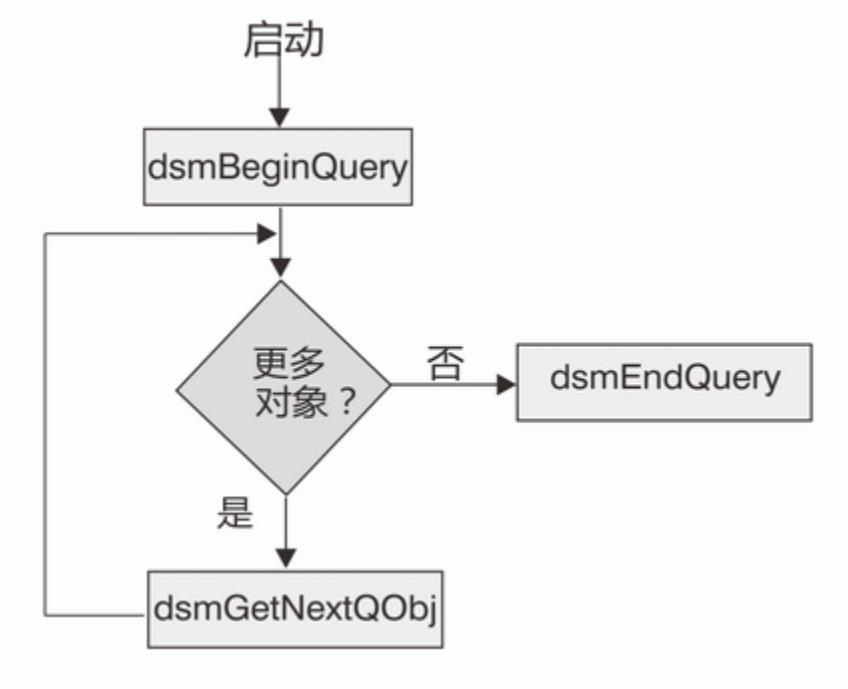


图 *10.* 常规查询的流程图

查询系统的示例

本示例中，管理类查询打印出特定管理类的备份和归档副本组中的所有字段的值。

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

/* Fill in the qMCData structure with the query criteria we want */
qMCData.stVersion = qryMCDataVersion;   /* structure version     */
qMCData.mcName    = mc;                 /* management class name */
qMCData.mcDetail  = bTrue;              /* want full details?    */

/* Set parameters of the data block used to get or send data     */
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:
            .
            .  /* 备份和归档副本组的其他字段                */
            .
         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);
}
```

图 *11.* 执行系统查询的示例

服务器效率

从 IBM Spectrum Protect 服务器检索或向其发送对象时，请使用下列准则。

· 从 IBM Spectrum Protect 服务器中检索对象时，请遵循下列准则：

  – 按照 IBM Spectrum Protect 服务器提供的复原顺序检索数据。 复原顺序对于磁带设备尤其重要，因为检索无序数据会导致倒带和安装。

- 即使数据存储在磁盘设备上，有序检索也仍然可以节省时间。
- 在单个 IBM Spectrum Protect 服务器会话中尽可能执行更多工作。
- 不要启动和停止多个会话。

· 向 IBM Spectrum Protect 服务器中发送对象时，请遵循下列准则：

- 在单个事务中发送多个对象。
- 避免每个事务发送一个对象（尤其在将数据直接发送到磁带设备的情况下）。 磁带设备事务的一部分用于确保将磁带 RAM 缓冲区中的数据写入到介质。

相关概念

第 54 页的『按恢复顺序选择对象并对其进行排序』

执行备份或归档查询后，应用程序客户机必须确定要恢复或检索的对象（如果有）。

相关信息

第 14 页的『启动或结束会话』

IBM Spectrum Protect 是基于会话的产品，且所有活动都必须在 IBM Spectrum Protect 会话中执行。为了启动会话，应用程序会启动 **dsmInitEx** 调用。 除了 **dsmQueryApiVersionEx**、**dsmQueryCliOptions** 和 **dsmSetUp**，必须首先执行该调用才能执行任何其他 API 调用。

向服务器发送数据

API 允许应用程序客户机向 IBM Spectrum Protect 服务器存储器发送数据或已指定对象及其相关联数据。

提示: 还可以对数据进行备份或归档。 执行事务中所有发送操作

事务模型

在备份或归档操作期间发送到 IBM Spectrum Protect 存储器的所有数据均在一个事务内完成。 事务模型提供高级别的数据完整性，但是它会施加一些限制，应用程序客户机必须将这些限制考虑在内。

通过调用 **dsmBeginTxn** 开始事务，或通过调用 **dsmEndTxn** 结束事务。 单个事务是一种原子操作。 事务边界内发送的数据将在事务结束时提交到系统，或在事务提前结束的情况下回滚。

事务可能由单个对象发送或多个对象发送组成。 要通过降低系统开销来提高系统性能，请在多个对象事务中发送较小对象。 应用程序客户机将确定单个事务还是多个事务比较适合。

将多个对象事务中的所有对象发送到同一副本目标。 如果需要将对象发送到先前对象以外的其他目标，那么结束当前事务并开始新事务。 在新事务中，可向新副本目标发送对象。

注: 不会检查不包含任何位数据 (*sizeEstimate=0*) 的对象的副本目标一致性。

IBM Spectrum Protect 限制可以在多个对象事务中发送的对象数。 要查找此限制，请调用 **dsmQuerySessInfo** 并检查 **maxObjPerTxn** 字段。 此字段显示服务器上设置的 *TXNGroupmax* 选项的值。

应用程序客户机必须跟踪在某个事务期间发送的对象，以在事务过早结束的情况下执行重试处理或错误处理。 服务器或客户机可随时停止事务。 应用程序客户机必须准备好处理未开始的突发事务结束。

文件聚集

IBM Spectrum Protect 服务器使用一项称为文件聚集的功能。 通过文件聚集，在单个事务中发送的所有对象会存储在一起，从而节省空间并改进性能。 您仍然可以单独查询和复原对象。

要使用此功能，事务中的所有对象都应具有同一文件空间名称。 如果文件空间名称在事务中发生更改，那么服务器会关闭现有聚集对象并开始新聚集对象。

不依赖 LAN 的数据传输 (LAN-free data transfer)

如果用于多线程的 **dsmSetUp** 选项设置为 ON，那么 API 可以利用不依赖 LAN 的数据传输。 API 会在 **Query Mgmt Class** 响应结构 **archDetailCG** 或 **backupDetailCG** 字段 **bLanFreeDest** 中返回不依赖 LAN 的目标的存在性。

可以在存储代理程序支持的平台上使用不依赖 LAN 的操作。 不包括 Macintosh 平台。

下列输出结构中会提供不依赖 LAN 的信息。 **dsmEndGetData** 的输出结构 (**dsmEndGetDataExOut_t**) 包含字段 **totalLFBytesRecv**。 此为收到的不依赖 LAN 的总字节数。 **dsmEndSendObjEx** 的输出结构 (**dsmEndSendObjExOut_t**) 包含字段 **totalLFBytesSent**。 此为已发送的不依赖 LAN 的总字节数。

相关信息

不依赖 LAN 的数据移动：存储代理程序概述

并发写操作

您可以配置 IBM Spectrum Protect 服务器存储池以在备份或归档期间同时写入到主存储池和一个或多个副本存储池。 使用此配置来创建对象的多个副本。

如果同时写操作失败，**dsmEndTxn** 函数的返回码可能是 DSM_RC_ABORT_STGPOOL_COPY_CONT_NO，表明对某个副本存储池的写入失败，并且 IBM Spectrum Protect 存储池选项 COPYCONTINUE 已设置为 NO。 应用程序将终止，该问题必须由 IBM Spectrum Protect 服务器管理员解决。

有关设置同时写操作的更多信息，请参阅 IBM Spectrum Protect 服务器文档。

增强 API 性能

可使用 tcpbuffsize 和 tcpnodelay 客户机选项和 **DataBlk** API 参数来增强 API 性能。

text="可增强性能的备份/归档选项以及 API 参数" 描述为增强 API 性能可执行的操作。

表 *10.* 可增强性能的备份/归档选项以及 *API* 参数

备份/归档客户机选项	**描述**
tcpbuffsize	指定 TCP 缓冲区的大小。 缺省值为 31 KB。 要增强性能，请将该值设置为 32 KB。
tcpnodelay	指定是否向服务器发送小的缓冲区（而非将其保留）。 要增强性能，请针对所有平台将此选项设置为 *yes*。 此选项仅对 Windows 和 AIX 有效。
API 参数	**描述**
DataBlk	此参数与 **dsmSendData** 函数调用结合使用，以确定应用程序缓冲区大小。 为实现最佳结果，请将该参数设置为 *tcpbuffsize* 值的关联值，即由 tcpbuffsize 减去 4 个字节而指定的值。 例如，当 tcpbuffsize 值设置为 32 KB 时，请将此参数的值设置为 28。

每个 **dsmSendData** 调用是同步的，并且只有在传输到 **dataBlkPtr** 中 API 的数据清空到网络之后才会返回。 API 会向放置在网络上的每个事务缓冲区添加 4 字节开销。

例如，如果事务缓冲区大小为 32 KB，而应用程序 **DataBlk** 缓冲区大小为 31 KB，那么每个应用程序 **DataBlk** 缓冲区将适应通信缓冲区，并可以立即清空。 但是，如果应用程序 **DataBlk** 缓冲区恰好为 32 KB，由于 API 要为每个事务缓冲区添加 4 个字节，因此需要两种清空：一个为 32 KB，一个为 4 字节。 此外，如果您将 tcpnodelay 选项设置为 no，那么清空这 4 个字节可能会耗时长达 200 毫秒。

设置 API 以将性能数据发送到客户机性能监视器

客户机性能监视器是 Tivoli® Storage Manager Administration Center 的组件，用于显示 API 收集到的性能数据。客户机性能监视器可为客户机备份、归档和复原操作记录和显示性能数据。

启用性能监视器时，您可以显示 API 使用性能监视器收集到的性能数据；性能收集器位于 Tivoli Storage Manager Administration Center 中。从 V7.1 开始，Administration Center 组件不再包含在 Tivoli Storage Manager 或 IBM Spectrum Protect 分发版中。如果您具有随先前服务器发行版一起安装的 Administration Center，那么可继续使用其来显示性能数据。如果尚未安装 Administration Center，那么可以从 ftp://public.dhe.ibm.com/storage/tivoli-storage-management/maintenance/admincenter/v6r3/ 下载先前发布的版本。有关使用性能监视器的信息，请参阅 Tivoli Storage Manager V6.3 服务器文档。

配置客户机性能监视器选项

通过在客户机选项文件中指定参数，启用 IBM Spectrum Protect 客户机以使用性能监视器。针对想要监视的每个客户机指定这些选项。

开始之前

在 UNIX 和 Linux 计算机上监视性能时，使用以下命令将打开文件描述符限制设置为至少 1024：

```
ulimit -n 1024
```

过程

要配置客户机性能监视器选项，请完成以下步骤：

1. 打开监视的每个客户机的客户机选项文件。根据配置，客户机选项位于以下一个文件中：
   - dsm.opt
   - dsm.sys
2. 将以下选项添加到客户机选项文件：

   ```
   PERFMONTCPSERVERADDRESS
   PERFMONTCPPORT
   PERFMONCOMMTIMEOUT
   ```

PERFMONTCPSERVERADDRESS

PERFMONTCPSERVERADDRESS 选项用于指定安装客户机性能监视器的系统的主机名和 IP 地址。

支持的客户机

该选项是独立的平台，支持所有客户机。

选项文件

在客户机选项文件（dsm.opt 或 dsm.sys）中设置该选项。

语法

参数

server
已安装客户机性能监视器的系统的服务器主机名或 IP 地址（与运行 Administration Center 的服务器是同一服务器）。

示例

选项文件：

```
PERFMONTCPSERVERADDRESS 131.222.10.5
```

命令行：

无法使用命令行设置该选项。

PERFMONTCPPORT

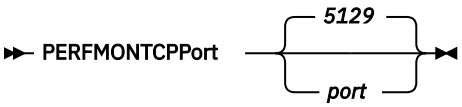
用于客户机性能监视器侦听客户机性能数据的端口号。

支持的客户机

该选项是独立的平台，支持所有客户机。

选项文件
在客户机选项文件（dsm.opt 或 dsm.sys）中设置该选项。

语法



参数

端口
为获取客户机性能数据而监视的端口。 端口 5129 是缺省端口。

示例

选项文件：
```
PERFMONTCPPPORT 5000
```

命令行：
无法使用命令行设置该选项。

PERFMONCOMMTIMEOUT

指定会话结束后 dsmTerminate 调用等待性能数据到达的最长时间（以秒为单位）。

支持的客户机
该选项是独立的平台，支持所有客户机。

选项文件
在客户机选项文件（dsm.opt 或 dsm.sys）中设置该选项。

语法

参数

seconds
会话结束前，等待剩余性能数据到达的时间。

示例

选项文件：
```
PERFMONCOMMTIMEOUT 60
```

命令行：
无法使用命令行设置该选项。

向服务器发送对象

应用程序客户机可使用 API 备份和归档函数将数据或指定对象及其关联数据发送到 IBM Spectrum Protect 存储。 系统的备份和归档组件允许对发送到存储的数据使用不同管理过程。

大小估算属性是对发送到服务器的数据对象总大小的估算。 如果应用程序不知道确切的对象大小，请将 *sizeEstimate* 设置为更高的估算值。 如果估算值小于实际大小，IBM Spectrum Protect 服务器将使用额外资源来管理额外空间分配。

提示:

- 在进行此大小估算时，请尽可能地精确。 服务器使用此属性在其存储资源中进行高效的空间分配和对象放置。
- 如果估算值小于实际大小，具有高速缓存的服务器不会分配额外空间并且停止发送。

如果 *sizeEstimate* 过大，可能会遇到问题。 服务器可能不具有足够空间用于估算的大小，但具有空间用于实际大小；或服务器可能使用较慢的设备。

可备份或归档大小大于 2 GB 的对象。 对象可以是压缩的，或解压缩的。

要开始发送操作，请调用 **dsmSendObj**。 如果一次要发送更多的数据，可重复调用 **dsmSendData** 以传输信息的其余部分。 调用 **dsmEndSendObj** 以完成发送操作。

了解备份和归档对象

IBM Spectrum Protect 系统的备份组件支持存储在服务器上的多个版本的指定对象。

备份到服务器的任何对象，只要与已从该客户机存储到服务器上的某个对象同名，均受到版本控制。 对象在服务器上可被视为活动状态或非活动状态。 服务器上尚未停用的对象的最新副本处于活动状态。 具有同一名称的任何其他对象（无论其为更旧版本还是已停用副本）均被视为处于不活动状态。 管理类构造会定义不同的管理条件。 这些构造会指派给服务器上的活动和不活动对象。

第 34 页的表 11 列出了适用于活动和不活动状态的副本组字段:

表 *11.* 备份副本组字段

字段	**描述**
VEREXISTS	现行版本存在时的非现行版本数。
VERDELETED	现行版本不存在时的非现行版本数。
RETEXTRA	非现行版本的保留天数。
RETONLY	现行版本不存在时上一个非现行版本的保留天数。

如果备份版本各自具有唯一名称（如在名称中使用时间戳记），那么版本控制不会自动发生：每个对象均处于活动状态。 活动对象永远不会到期，因此应用程序会负责通过 **dsmDeleteObj** 调用来停用这些活动对象。 在此情况下，应用程序需要尽快使已停用对象到期。 用户会定义 VERDELETED=0 且 RETONLY=0 的备份副本组。

IBM Spectrum Protect 系统的归档组件准许将对象存储在具有保留期或截止期控制而不是版本控制的服务器上。 存储的各对象均唯一，即使其名称可能与已归档的对象相同也如此。 归档对象具有与元数据关联的描述字段，此字段在查询期间可用于识别特定对象。

将为 IBM Spectrum Protect 服务器上的每个对象指定唯一对象标识。在对象生存期内（具体而言，即导出或导入后）不保证原始值的持久性。 因此，应用程序不应查询并保存原始对象标识来用于以后进行复原。 应用程序而是应保存对象名和插入日期。 可以在复原期间使用此信息来查询对象和验证插入日期。 然后，可以使用当前对象标识来复原对象。

压缩

给定节点上的配置选项以及 **dsmSendObj** objCompressed 选项确定在发送期间 IBM Spectrum Protect 是否会压缩对象。 此外，绝不会压缩 sizeEstimate 小于 DSM_MIN_COMPRESS_SIZE 的对象。

如果已压缩该对象 (objCompressed=bTrue)，那么不会再次对其进行压缩。 如果未进行压缩，那么 IBM Spectrum Protect 会根据管理员在 API 配置源中设置的压缩选项的值来决定是否压缩对象。

管理员可以使用 register node 命令（compression=yes、no 或 client-determined）更改服务器上的压缩阈值。如果是 client-determined，那么压缩行为由配置源中的压缩选项值来确定。

某些类型的数据（如已压缩的数据）在通过压缩算法进行处理时可能实际会增大。 在发生此情况时，会生成返回码 DSM_RC_COMPRESS_GREW。 如果意识到可能会发生此情况，但是仍要继续进行发送操作，请指示最终用户在其选项文件中指定下列选项：

```
COMPRESSAlways Yes
```

在启用压缩的情况下，如果您在 **dsmSendData** 函数期间收到 DSM_RC_COMPRESS_GREW 返回码，那么您可能希望从头开始，在不压缩的情况下再次发送对象。 要强制实施此操作，请将 **dsmSendObj** ObjAttr.objCompressed 设置为 bTrue。

有关 **dsmSendObj** 期间的实际压缩行为的信息会由 **dsmEndSendObjEx** 调用返回。 objCompressed 指定是否已完成压缩。 totalBytesSent 为应用程序发送的字节数。 totalCompressedSize 为压缩后的字节数。 **dsmEndSendObjEx** 调用还具有 totalLFBytesSent 字段，其中包含不依赖 LAN 发送的总字节数。

注意: 如果应用程序计划使用部分对象恢复或检索，在发送该数据时您无法压缩它。 要强制实施此操作，请将 **dsmSendObj** ObjAttr.objCompressed 设置为 bTrue。

压缩类型

客户机使用的压缩类型由压缩与备份或归档处理期间使用的客户机端重复数据删除的组合确定。

qryRespArchiveData 和 **qryRespBackupData** 结构的新字段中的 API 报告客户机所使用的压缩算法：

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

报告以下压缩类型：

LZ4
客户机在以下任何情况下使用的更快更高效的压缩方法：

- 将客户机端已进行重复数据删除的对象发送到 IBM Spectrum Protect 服务器上的容器存储池。服务器必须为 V7.1.5 或更高版本。
- 对象不经历客户机端重复数据删除。
- 对象仅经历传统服务器端重复数据删除。

LZW
在将客户机重复数据删除的对象发送到服务器上的传统（非容器）存储池时客户机使用的一种传统压缩类型。

空白字段
客户机未压缩对象。 由于 compression 选项设置为 *no*，或者在备份或归档处理期间未指定该选项，因此无法压缩对象。 虽然对象未由客户机压缩，但其可能由服务器压缩。

压缩类型非可配置。它在备份或归档处理时由备份/归档客户机确定。

示例

以下示例显示来自 64 位样本应用程序 **dapismp** 的备份和归档查询输出中的 Compresssion Type 字段：

```
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
```

缓冲区副本消除

缓冲区副本消除功能可以除去应用程序和 IBM Spectrum Protect 服务器之间的数据缓冲区副本，从而提高处理器利用率。为最大限度实现效果，请在不依赖 LAN 的环境中使用此方法。

数据移动的缓冲区由 IBM Spectrum Protect 分配，并且会将指针传递回应用程序。 应用程序将数据放置在所提供的缓冲区中，并且该缓冲区会使用共享内存通过通信层传递到存储代理程序。 然后，数据将移动至磁带设备，从而消除了数据副本。 此功能可以用于备份或归档操作。

注意: 使用此方法时，请特别注意正确的缓冲区处理和缓冲区大小。 缓冲区在组件之间进行共享，因编程错误而产生的任何内存覆盖将会导致严重错误。

对备份/归档的总体调用顺序如下：

```
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 函数可以多次调用，次数最多为 numTsmBuffers 选项所指定的值。 应用程序可以包含两个线程：生产者线程（用于使用数据填充缓冲区）和消费者线程（用于通过 **dsmSendBufferData** 调用将这些缓冲区发送到 IBM Spectrum Protect）。 在发出 **dsmRequestBuffer** 调用时，如果达到 **numTsmBuffers**，那么该 **dsmRequestBuffer** 调用会阻塞，直至释放了缓冲区为止。 通过调用 **dsmSendBufferData**（发送和释放缓冲区）或调用 **dsmReleaseBuffer**，可以释放缓冲区。 有关更多信息，请参阅 API 样本目录中的 `callbuff.c`。

如果在任何时候发送失败，那么应用程序必须释放被占用的所有缓冲区并终止会话。 例如：

```
  If failure
    for each data buffer held by application
```

```
      call dsmReleaseBuffer
  dsmTerminate
```

如果应用程序调用了 **dsmTerminate** 之后仍有缓冲区被占用，那么 API 不会退出。 将返回以下代码：DSM_RC_CANNOT_EXIT_MUST_RELEASE_BUFFER。 如果应用程序无法释放该缓冲区，那么应用程序必须退出进程以强制清除。

缓冲区副本消除和恢复/检索

IBM Spectrum Protect 服务器根据对复原和检索的磁带存取优化，控制要放置在缓冲区中的数据量。 对于应用程序而言，此方法不及正常的数据获取方法有用。 在原型设计期间，检查缓冲区副本消除方法的性能，并且仅当您看到性能明显提升时才使用此方法。

IBM Spectrum Protect 服务器返回的单个缓冲区中的最大数据量为（256KB - 报头开销）。因此，只有处理小型缓冲区写入的应用程序才能从该数据检索机制中获益。 应用程序必须特别注意缓冲区中的字节数，具体取决于对象大小、网络和其他边界条件。 在某些情况下，使用缓冲区副本消除实际上可能比正常恢复的性能更差。 API 通常将数据进行高速缓存，并将固定长度返回到应用程序。 然后，应用程序可以控制返回磁盘的数据写入数。

如果使用缓冲区副本消除，请为小于首选写缓冲区大小的缓冲区创建一种数据高速缓存机制。 例如，如果应用程序将 64K 数据块写入磁盘，那么应用程序必须执行以下操作：

1. 调用 **dsmGetBufferData**。
2. 写出 64K 的块。
3. 在最后一个块上，将其余部分复制到 **tempBuff**，发出另一个 **dsmGetBufferData** 调用，并使用数据的其余部分填充 **tempBuff**。
4. 继续写入 64K 的块：

```
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
```

在此示例中，有 6 个磁盘写入是直接写入的，有 1 个进行了高速缓存。

对复原和检索的调用的总体顺序如下所示：

dsmInitEx（UseTsmBuffers = True numTsmBuffers = 应用程序要分配的缓冲区数量）。

```
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
```

对于每个 **dsmGetBufferData** 调用，实现一个 **dsmReleaseBuffer** 调用。 **dsmGetBufferData** 和相对应的 **dsmReleaseBuffer** 无需连续。 一个应用程序可能会首先发出多个 **dsmGetBufferData** 调用以获取若干缓冲区，稍后发出相应的 **dsmReleaseBuffer** 调用。 有关使用此函数的样本代码，请参阅 API 样本目录中的 callbuff.c。

限制: 由于 API 提供缓冲区，并且目标是最大限度降低处理器利用率，因此不允许对缓冲区中的数据进行更多处理。应用程序无法将加密和压缩用于缓冲区副本消除，因为这两种操作都需要数据处理和复制。

同时实现常规数据移动途径和缓冲区副本消除，以支持用户根据需要在两种途径之间切换。 如果用户必须压缩或加密数据，那么使用现有机制。 如果存在处理器约束，那么使用新机制。这两种机制是互补的，不会完全取代对方。

API 加密

两种方法可用于加密数据：应用程序管理的加密和 IBM Spectrum Protect 客户机加密。

仅选择并使用其中一种方法来加密数据。这两种方法互斥，并且如果使用两种方法加密数据，那么将无法复原或检索部分数据。例如，假定应用程序使用应用程序管理的加密来加密对象 A，然后使用 IBM Spectrum Protect 客户机加密来加密对象 B。在复原操作期间，如果应用程序设置选项以使用 IBM Spectrum Protect 客户机加密并尝试复原两个对象，那么只能复原对象 B；无法复原对象 A，因为其由应用程序进行加密，而非客户机。

无论使用何种加密方法，IBM Spectrum Protect 都必须启用密码认证。缺省情况下，服务器使用 SET AUTHENTICATION ON。

API 使用 AES 128 位或 AES 256 位加密。AES 256 位数据加密提供了比 AES 128 位数据加密更高级别的数据加密。无法使用较早版本的客户机复原使用 AES 256 位加密备份的文件。 可以在压缩或不压缩的情况下启用加密。 如果使用加密，那么无法使用部分对象复原和检索以及缓冲区副本消除功能。

应用程序管理的加密

通过应用程序管理的加密，应用程序可以向 API 提供密钥密码（通过使用密钥 DSM_ENCRYPT_USER）并且负责管理密钥密码。

注意: 如果未保存加密密钥，并且您忘记了密钥，那么数据将不可恢复。

应用程序在 **dsmInitEx** 调用中提供密钥密码，并且必须在复原时提供正确的密钥密码。

注意: 如果密钥密码丢失，那么将无法复原数据。

必须将相同密钥密码用于同一对象的备份和复原（或归档和检索）操作。此方法不具有 IBM Spectrum Protect 服务器级别依赖性。要设置此方法，应用程序需要遵循以下步骤：

1. 在 **dsmInitEx** 调用中将 bEncryptKeyEnabled 变量设置为 bTrue，并且将 encryptionPasswordP 变量设置为指向含有加密密钥密码的字符串。
2. 将对象的 include.encrypt 设置为 encrypt。 例如，要加密所有数据，请设置：

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

   以及

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

   要加密对象 /FS1/DB2/FULL，请设置：

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

3. 在传递到 **dsmInitEx** 调用中的 API 的选项字符串中设置 ENCRYPTKEY=PROMPT|SAVE 选项（在 Windows 上）。也可以在 dsm.opt (Windows) 或 dsm.sys（UNIX 或 Linux）中设置此选项。

缺省情况下，encryptkey 选项设置为 prompt。 此设置确保不会自动存储密钥。如果指定了 encryptkey save，那么会通过 IBM Spectrum Protect 将密钥存储在本地机器上，但是这样一来，仅一个密钥可对含有同一节点名的所有 IBM Spectrum Protect 操作均有效。

发送对象后，**dsmEndSendObjEx** 会指定是否已将对象加密以及所使用的方法。*encryptionType* 字段中可能的值：

- DSM_ENCRYPT_NO
- DSM_ENCRYPT_USER
- DSM_ENCRYPT_CLIENTENCRKEY

下表列出了 API 加密类型、先决条件和可用函数。

表 *12. API* 加密类型、先决条件和可用函数

类型	先决条件	可用函数
ENCRYPTIONTYPE	无	在传递到 **dsmInitEx** 调用的 API 的选项字符串中设置 ENCRYPTIONTYPE（在 Windows 上）。缺省情况下，ENCRYPTIONTYPE=AES128。
EncryptKey=save	无	API 和备份/归档
EncryptKey=prompt	无	API 和备份/归档
EncryptKey=generate	无	API 和备份/归档
EnableClientEncryptKey	无	仅 API

注: 建议服务器将 authentication 设置为 ON。如果 authentication 设置为 OFF，那么不会加密密钥，但是仍然会加密数据。 但是，不建议此设置。

第 39 页的表 13 显示授权用户和非授权用户如何能够根据为 passwordaccess 选项指定的值，在备份或复原操作期间加密或解密数据。要执行以下授权用户和非授权用户操作，TSM.KDB、TSM.sth 和 TSM.IDX 文件必须存在。授权用户将 encryptkey 选项设置为 **save**，并且将 passwordaccess 选项设置为 **generate**。

表 *13.* 在 *UNIX* 或 *Linux* 上使用应用程序管理的密钥加密或解密数据

操作	passwordaccess 选项	encryptkey 选项	结果
已授权用户备份	generate	save	数据已加密。
	generate	prompt	如果 encryptionPasswordP 包含加密密码，那么数据已加密。
	prompt	save	如果 encryptionPasswordP 包含加密密码，那么数据已加密。
	prompt	prompt	如果 encryptionPasswordP 包含加密密码，那么数据已加密。
已授权用户恢复	generate	save	数据已加密。
	generate	prompt	如果 encryptionPasswordP 包含加密密码，那么数据已加密。
	prompt	save	如果 encryptionPasswordP 包含加密密码，那么数据已加密。
	prompt	prompt	如果 encryptionPasswordP 包含加密密码，那么数据已加密。
非授权用户备份	generate	save	数据已加密。
	generate	prompt	如果 encryptionPasswordP 包含加密密码，那么数据已加密。
	prompt	save	如果 encryptionPasswordP 包含加密密码，那么数据已加密。
	prompt	prompt	如果 encryptionPasswordP 包含加密密码，那么数据已加密。
非授权用户恢复	generate	save	数据已加密。
	generate	prompt	如果 encryptionPasswordP 包含加密密码，那么数据已加密。
	prompt	save	如果 encryptionPasswordP 包含加密密码，那么数据已加密。
	prompt	prompt	如果 encryptionPasswordP 包含加密密码，那么数据已加密。

IBM Spectrum Protect 客户机加密

IBM Spectrum Protect 客户机加密使用 DSM_ENCRYPT_CLIENTENCRKEY 值管理的密钥来保护数据。客户机加密对于使用 API 的应用程序透明，例外是部分对象复原操作和接收操作不可用于已加密或已压缩的对象。

对于 IBM Spectrum Protect 客户机加密和应用程序管理的加密，加密密码指示用于生成实际加密密钥的字符串值。加密密码选项值的长度为 1-63 个字符，但是从其生成的密钥始终为 8 字节（对于 56 DES）、16 字节（对于 128 AES）和 32 字节（对于 256 AES）。

注意: 如果加密密钥不可用，那么无法复原或检索数据。使用 ENABLECLIENTENCRYPTKEY 进行加密时，加密密钥存储在服务器数据库中。对于使用此方法的对象，服务器数据库必须存在并且具有对象的正确值才能进行正确复原。请确保经常备份服务器数据库以防止数据丢失。

这是较为简单的实施方法，通过此方法，每个会话生成一个随机加密密钥，并且该密钥会与服务器数据库中的对象一起存储在 IBM Spectrum Protect 服务器上。在复原期间，存储的密钥用于解密。 使用此方法，IBM Spectrum Protect 负责密钥管理，应用程序完全不必处理密钥。因为密钥存储在服务器数据库中，您必须具有有效的 IBM Spectrum Protect 数据库才能对加密对象执行复原此操作。当密钥在 API 和服务器之间传输时，也进行加密。密钥传输是安全的，并且当密钥存储在 IBM Spectrum Protect 服务器数据库中时，密钥进行加密。仅当节点的数据在服务器之间导出时，才会将密钥以未加密方式与导出数据流放置在一起。

要启用 IBM Spectrum Protect 客户机加密，请完成以下步骤：

1. 在传递到 dsmInitEx 调用上的 API 的选项字符串中指定 -ENABLECLIENTENCRYPTKEY=YES，或者在系统选项文件 dsm.opt (Windows) 或 dsm.sys（UNIX 或 Linux）中设置选项。
2. 将对象的 include.encrypt 设置为 encrypt。 例如，要加密所有数据，请设置：

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

   以及

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

   要加密对象 /FS1/DB2/FULL，请设置：

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

数据去重

重复数据删除是一种通过消除冗余数据来减少存储需求的方法。

概述

IBM Spectrum Protect 上提供两种类型的重复数据删除：客户机端重复数据删除和服务器端重复数据删除。

客户机端重复数据删除是在备份/归档客户机上使用的重复数据删除技术，用于备份和归档处理期间，在将数据传输到 IBM Spectrum Protect 服务器之前除去冗余数据。使用客户机端重复数据删除可以减少通过局域网发送的数据量。

服务器端重复数据删除是通过服务器执行的重复数据删除方法。 IBM Spectrum Protect 管理员可以指定重复数据删除位置（客户机或服务器）以与 **REGISTER NODE** 或 **UPDATE NODE** 服务器命令上的 **DEDUP** 参数一起使用。

增强功能

有了客户机端重复数据删除，您可以：

- 将客户机上的特定文件从客户机端重复数据删除中排除。
- 启用减少客户机和服务器之间网络流量的数据去重高速缓存。 该高速缓存包含在先前增量备份操作中发送到服务器的扩展数据块。 客户机不会在服务器中查询扩展数据块是否存在，而是在服务器的高速缓存中查询。

指定客户机高速缓存的大小和位置。 如果在服务器和本地高速缓存之间检测到不一致，会移除并重新填充本地高速缓存。

注: 对于使用 IBM Spectrum Protect API 的应用程序，不能使用数据去重高速缓存，因为与 IBM Spectrum Protect 服务器的高速缓存不同步可能造成备份失败。 如果配置了多个并发备份/归档客户机会话，那么必须为每个会话配置单独的高速缓存。

- 同时启用客户机端重复数据删除和压缩来减少服务器存储的数据量。 每个扩展数据块在发送到服务器之前都经过压缩。 优点是节省存储器，缺点是压缩客户机数据需要处理能力。 通常，如果您在客户机系统上压缩数据并去重，使用的处理能力大约是只进行数据去重的两倍。

  服务器可以使用经过去重的压缩数据。 此外，早于 V6.2 的备份/归档客户机可以复原经过去重的压缩数据。

客户机端重复数据删除使用以下过程:

- 客户机创建扩展数据块。 扩展数据块是与其他文件扩展数据块相比较的文件部分，用于识别重复内容。
- 客户机和服务器协作以识别重复扩展数据块。 客户机将非重复扩展数据块发送到服务器。
- 后续客户机重复数据删除操作创建新的扩展数据块。 部分或全部这些扩展数据块可能与先前重复数据删除操作中创建并已发送到服务器的扩展数据块相匹配。 不会再次将匹配的扩展数据块发送到服务器。

优点

客户机端重复数据删除提供多种优点:

- 它可以减少通过局域网 (LAN) 发送的数据量。
- 识别重复数据所需的处理能力负载从服务器转到了客户机节点。 服务器端数据去重总是对支持重复数据删除的存储池启用。 但是，支持重复数据删除的存储池中的文件和由客户机去重的文件不需要附加处理。
- 移除服务器上重复数据的处理能力需求被省去了，从而可在服务器上立即实现空间节省。

客户机端重复数据删除也有潜在的缺点。 服务器没有客户机文件的完整副本，除非您将包含客户机扩展数据块的主存储池备份到非去重副本存储池。 （扩展数据块是在数据去重过程中创建的文件部分。） 在存储池备份到非去重存储池的过程中，客户机扩展数据块会被重新组合为邻接文件。

缺省情况下，必须先将为数据去重设置的主顺序存取存储池备份到非去重副本存储池，才能回收这些存储池和移除重复数据。 缺省设置确保了服务器始终在主存储池或副本存储池中拥有完整文件的副本。

要点: 为进一步减少数据，可以将客户机端重复数据删除和压缩一起启用。 每个扩展数据块在发送到服务器之前都经过压缩。 压缩可以节省空间，但会增加客户机工作站上的处理时间。

下列选项与重复数据删除相关:

- 重复数据删除
- Dedupcachepath
- Dedupcachesize
- Enablededupcache
- Exclude.dedup
- Include.dedup

API 客户机端重复数据删除

客户机端重复数据删除由备份/归档客户机上的 API 使用，以在将数据传输到 IBM Spectrum Protect 服务器之前，除去备份和归档处理期间的冗余数据。

客户机端重复数据删除由 API 使用，用于在备份和归档处理期间，在将数据传输到 IBM Spectrum Protect 服务器之前除去冗余数据。使用客户机端重复数据删除可以减少通过局域网发送的数据量。 使用客户机端重复数据删除还可减少 IBM Spectrum Protect 服务器存储空间。

在已启用客户机来进行客户机端重复数据删除，并且执行备份或归档操作时，会将数据作为扩展数据块发送到服务器。 下次执行备份或归档操作后，客户机和服务器即可识别已备份或归档的扩展数据块，并且仅将唯一的扩展数据块发送到服务器。

对于客户机端重复数据删除，服务器和 API 必须为版本 6.2 或更高版本。

使用客户机端重复数据删除来备份文件或归档文件之前，系统必须符合下列需求：

- 客户机必须已启用 deduplication 选项。
- 服务器必须通过 **REGISTER NODE** 或 **UPDATE NODE** 命令上的 **DEDUP=CLIENTORSERVER** 参数来使客户机支持客户机端重复数据删除。
- 数据的存储池目标必须是启用了重复数据删除的存储池。 启用了重复数据删除的存储池仅为文件设备类型。
- 确保将文件绑定到正确的管理类。
- 可以从客户机端重复数据删除处理中排除文件。 缺省情况下，所有文件都包括在内。
- 文件必须大于 2 KB。
- 通过在服务器上设置 CLIENTDEDUPTXNLIMIT 选项，服务器可以限制重复数据删除的最大事务大小。 请参阅服务器文档以获取有关此选项的信息。

如果不符合这些要求中的任一要求，那么会正常处理数据，而不进行任何客户机端重复数据删除。

下列是一些重复数据删除限制：

- 不依赖 LAN 的数据移动和客户机端重复数据删除是互斥的。 如果同时启用不依赖 LAN 的数据移动和客户机端重复数据删除，那么会完成不依赖 LAN 的数据移动操作而忽略客户机端重复数据删除。
- 加密和客户机端重复数据删除是互斥的。 如果同时启用加密和客户机端重复数据删除，那么会完成加密操作而忽略客户机端重复数据删除。 加密文件和适合进行客户机端重复数据删除的文件可以在同一操作中进行处理，但是在不同事务中执行。

  需求：

  1. 在任何事务中，都必须包含所有文件以进行重复数据删除或排除这些文件。 如果事务包含混合文件，那么事务将失败，并且 API 会返回返回码 DSM_RC_NEEDTO_ENDTXN。
  2. 将存储设备加密与客户机端重复数据删除一起使用。 由于 SSL 与客户机端重复数据删除组合使用，因此无需进行客户机加密。
- 下列功能不适用于客户机端重复数据删除：
  - IBM Spectrum Protect for Space Management (HSM) 客户机
  - API 共享缓冲区
  - NAS
  - 子文件备份
- 缓冲区副本消除无法用于压缩、加密和重复数据删除之类的数据变换。
- 如果使用客户端重复数据删除，那么发送数据到该服务器期间，API 将检测并放弃服务器上标记为到期的文件扩展数据块备份（带有 RC=254）。 如果要重试该操作，您需要包括在调用程序中编程的文件扩展数据块。
- 服务器上的同时写入操作优先于客户机端重复数据删除。 如果启用了同时写入操作，那么不会进行客户机端重复数据删除。

  限制： 在启用了客户机端重复数据删除时，API 无法从服务器已耗尽目标池上的存储空间这一状态中恢复，即使已定义有下一个池也如此。 此时会返回停止原因码 DSM_RS_ABORT_DESTINATION_POOL_CHANGED，并且操作失败。 从此情况中恢复有两种方法：

  1. 请求管理员向原始文件池中添加更多临时卷。
  2. 在禁用重复数据删除的情况下重试操作。

要节省更多带宽，可以启用本地高速缓存来进行重复数据删除。 本地高速缓存可以避免将查询发送到 IBM Spectrum Protect 服务器。 ENABLEDEDUPCACHE 的缺省值为 NO，以便高速缓存不会与服务器不同步。 如果高速缓存与服务器不同步，那么应用程序将重新发送所有数据。 如果应用程序可以对失败的事务进行重试，并且要使用本地高速缓存，请在 dsm.opt (Windows) 或 dsm.sys (UNIX) 文件中将 ENABLEDEDUPCACHE 选项设置为 YES。

复原结束时，如果通过 API 已复原所有数据，并且客户机已对对象进行重复数据删除，那么会计算端到端摘要并将其与备份时计算的值相比较。 如果这些值不匹配，那么会返回错误 DSM_RC_DIGEST_VALIDATION_ERROR。 如果应用程序收到此错误，那么表明数据损坏。 此错误也可能是网络上的瞬时错误导致的，因此请重试复原或检索。

以下是显示重复数据删除信息的查询会话命令示例：

```
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:
```

以下是显示重复数据删除信息的查询管理类命令示例：

```
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
```

相关参考

重复数据删除选项

从重复数据删除中排除文件

您可以选择从重复数据删除中排除备份文件或归档文件。

要从重复数据删除处理中排除文件，请执行下列步骤：

1. 为要排除的对象设置 `exclude.dedup` 选项。

例如，要排除 UNIX 系统的所有 dedup 数据，请设置：

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

2. 要排除 Windows 系统的所有 dedup 数据，请设置：

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

要点: 如果向重复数据删除池发送了对象，那么在服务器上会进行重复数据删除，即使从客户机端重复数据删除中排除了该对象也如此。

包含文件以进行重复数据删除

您可以选择包含备份文件或归档文件来进行重复数据删除。

要优化需要包含的文件的列表，可以将 `include.dedup` 选项与 `exclude.dedup` 选项组合使用。

缺省情况下，会包含所有合适对象来进行重复数据删除。

下列是一些 UNIX 和 Linux 示例：

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

下列是一些 Windows 示例：

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

服务器端数据去重

服务器端重复数据删除是通过服务器执行的重复数据删除。

IBM Spectrum Protect 管理员可以指定重复数据删除位置（客户机或服务器）以与 **REGISTER NODE** 或 **UPDATE NODE** 服务器命令上的 **DEDUP** 参数一起使用。

在启用了重复数据删除的存储池（文件池）中，仅保留一个扩展数据块实例。 同一扩展数据块的其他实例会替换为已保留实例的指针。

有关服务器端重复数据删除的更多信息，请参阅 IBM Spectrum Protect 服务器文档。

应用程序故障转移

在 IBM Spectrum Protect 服务器由于停运而变为不可用时，使用 API 的应用程序可自动故障转移至辅助服务器以进行数据恢复。

正常生产过程期间，客户机和 API 连接至的 IBM Spectrum Protect 服务器称为主服务器。在设置主服务器以进行节点复制时，该服务器也称为源复制服务器。可以将源复制服务器上的客户机节点数据复制到目标复制服务器。此服务器也称为辅助服务器，是主服务器发生故障时客户机自动故障转移到的服务器。

客户机和 API 必须针对自动客户机故障转移进行配置，并且必须连接到复制客户机节点数据的 V7.1（或更高版本）服务器。API 的配置与备份/归档客户机的配置相同。

正常操作期间，在登录过程中会自动将辅助服务器的连接信息从主服务器发送到客户机。辅助服务器信息将自动保存到客户机选项文件。

每次客户机应用程序登录到 IBM Spectrum Protect 服务器时，都会尝试连接主服务器。如果主服务器不可用，那么应用程序会使用客户机选项文件中的辅助服务器信息自动故障转移到辅助服务器。在故障转移方式下，应用程序可查询辅助服务器并复原或检索复制的数据。

必须至少将应用程序备份到主服务器一次。仅在已将来自于客户机节点的数据从主服务器复制到辅助服务器时，API 才能故障转移到辅助服务器以恢复数据。

相关概念
自动客户机故障转移配置和使用

故障转移状态信息

API 提供状态信息，应用程序可使用此信息来确定故障转移状态和辅助服务器上复制的客户机数据的状态。

复制状态指示是否将最近一次备份复制到辅助服务器。 如果 API 上最新备份操作的时间戳记与辅助服务器上备份的时间戳记相匹配，那么复制状态为最新。如果两个时间戳记不匹配，那么复制状态非最新，并且复制的数据可能过期。

qryRespFSData 结构中 **dsmGetNextQObj** 函数调用的 **query filespace** 响应上返回以下复制状态信息：

表 *14. API* 报告的复制状态信息

状态信息	类型	定义
上次复制的开始时间	**lastReplStartDate**	最近一次启动复制的时间。
上次复制的结束时间	**lastReplCmpltDate**	上次完成复制的时间，即使存在故障。
上次备份存储日期（服务器）	**lastBackOpDateFromServer**	上次在服务器上进行保存的存储时间戳记。
上次备份存储日期（本地）	**lastBackOpDateFromLocal**	上次在客户机上进行保存的存储时间戳记。

dsmInitExOut_t 结构中的 **bIsFailOverMode** 字段报告故障转移状态。

请参阅第 133 页的『附录 B API 类型定义源文件』以了解 API 的结构和类型定义。

DSM_RC_SIGNON_FAILOVER_MODE 返回码指示客户机和 API 已故障转移到辅助服务器，并且正在以故障转移方式运行。

故障转移期间的登录示例

以下样本输出是故障转移期间登录服务器的示例：

```
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
************************************************************
```

query session 命令示例

以下样本输出是显示辅助（复制）服务器信息的 **query session** 命令的示例：

```
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
```

query filespace 命令示例

以下样本输出是显示辅助服务器上文件空间复制状态的 **query filespace** 命令的示例：

```
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
```

相关参考

第 89 页的『dsmGetNextQObj』

dsmGetNextQObj 函数调用从先前的 **dsmBeginQuery** 调用中获取下一个查询响应，并将响应放置在调用者缓冲区中。

备份和归档的流程图示例

API 针对直接明了的逻辑流以及应用程序客户机各种状态之间的清晰转换而设计。 这种清晰的状态转换可在开发周期中捕获逻辑缺陷和程序错误，从而极大增强了系统的质量和可靠性。

例如，除非已启动事务并且先前已经为要备份的对象调用了 **dsmBindMC**，否则将无法调用 **dsmSendObj**。

第 47 页的图 12 显示在某个事务内执行备份或归档操作的状态图。 从 “In Send Object” 指向 **dsmEndTxn** 的箭头表示您可以在调用 **dsmSendObj** 或 **dsmSendData** 后启动 **dsmEndTxn** 调用。 如果在发送对象期间发生错误条件并且希望停止操作，那么可能希望执行此操作。 在这种情况下，必须使用 DSM_VOTE_ABORT 的表决。 但是在正常条件下，在结束事务之前调用 **dsmEndSendObj**。

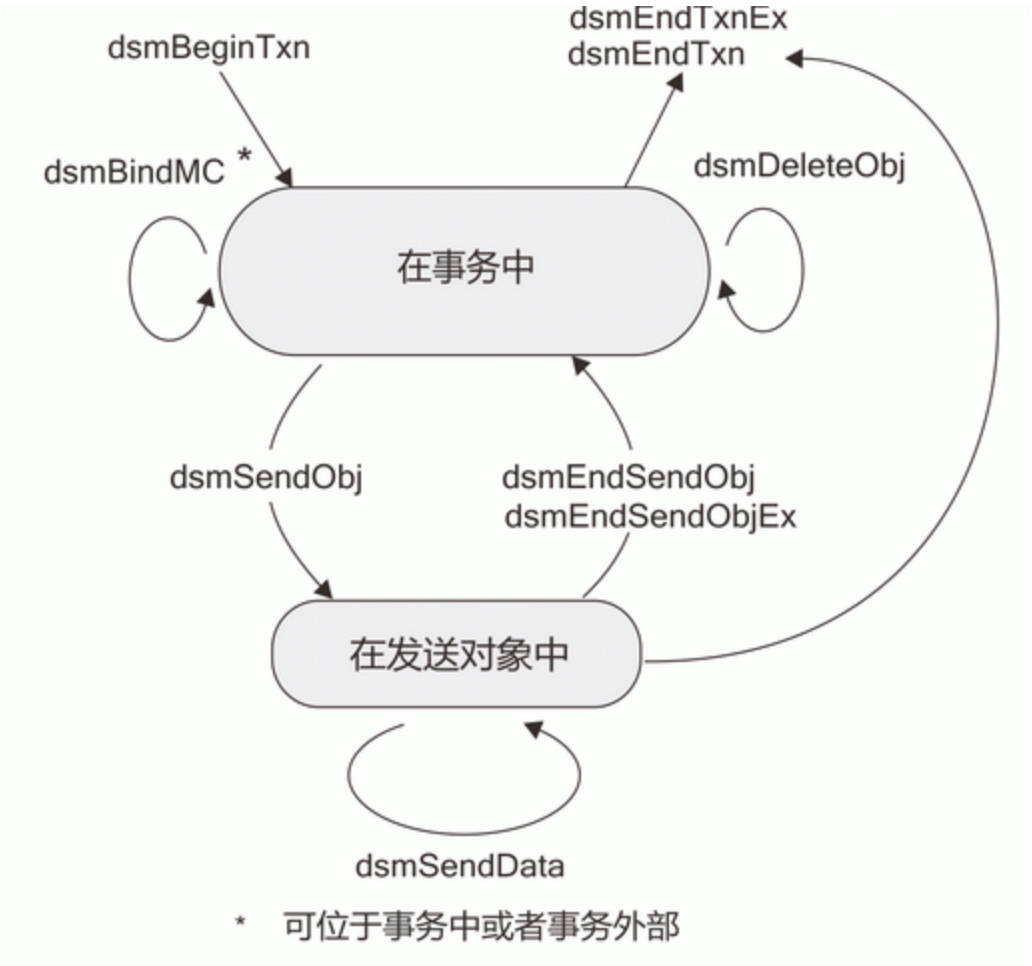


图 *12.* 备份和归档操作的状态图

第 48 页的图 13 显示有关在某个事务中执行备份或归档操作的流程图。

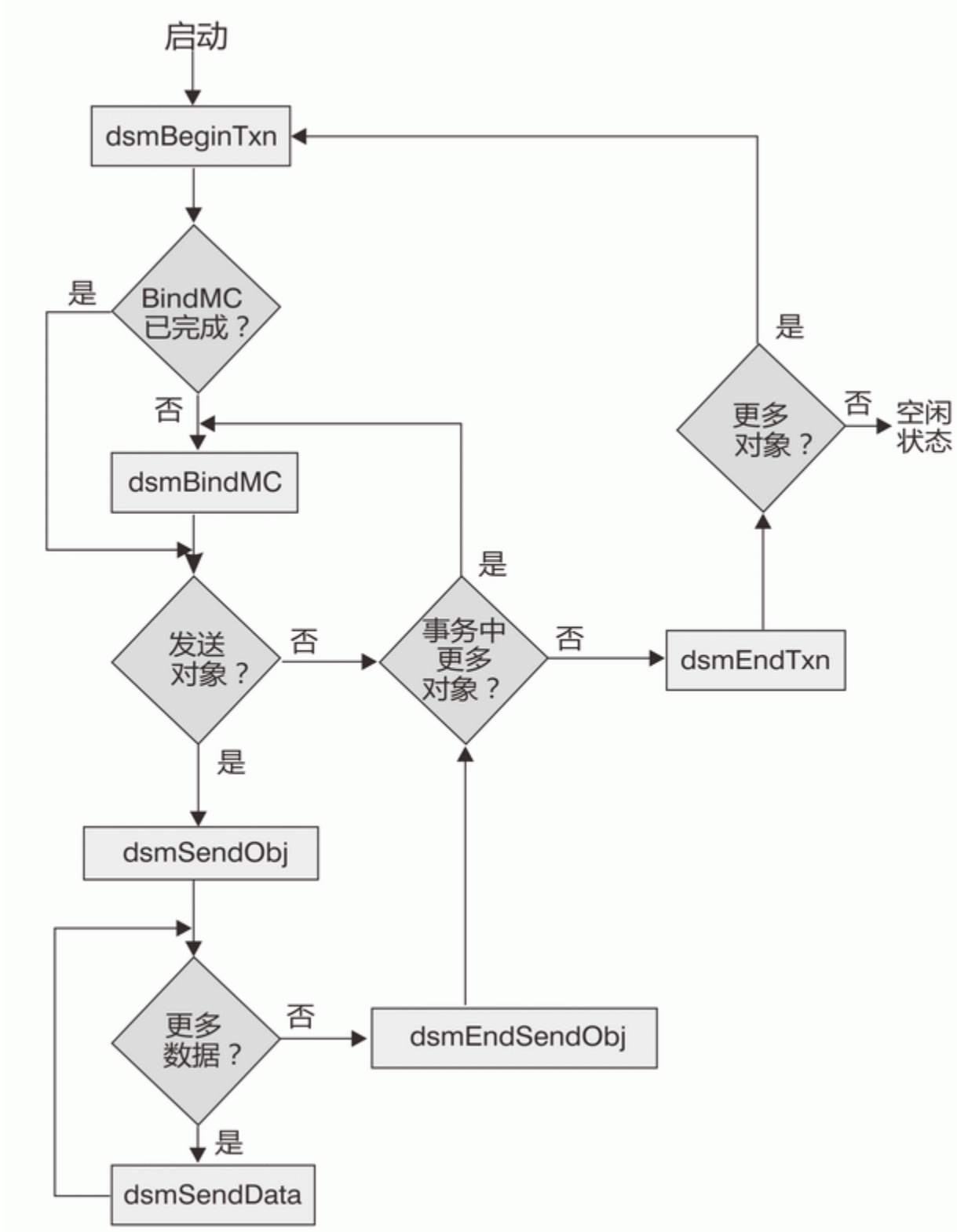


图 *13.* 备份和归档操作的流程图

这两个图中的主要功能是某个事务中以下 API 调用之间的循环：

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

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

dsmBindMC 调用是唯一的，因为您可以从事务边界的外部或内部将其启动。 如果需要，也可以从其他事务中将其启动。 **dsmBindMC** 调用的唯一要求是在备份或归档对象之前进行该调用。 如果要备份或归档的对象未与管理类相关联，那么从 **dsmSendObj** 返回错误代码。 在这种情况下，通过调用 **dsmEndTxn** 结束事务（流程图中未显示此错误条件）。

流程图显示了应用程序如何使用多个对象事务。 它显示了可以放置决策点的位置，以确定是发送的对象适应事务，还是开始新事务。

向 IBM Spectrum Protect 存储器发送数据的 API 函数代码示例

此示例演示向 IBM Spectrum Protect 存储器发送数据的 API 函数的使用。 **dsmSendObj** 调用显示在 switch 语句中，因此根据正在执行的是备份还是归档操作，可以调用的参数也将有所不同。

dsmSendData 调用从循环的内部调用，该循环不断发送数据，直至设置了某个允许程序执行退出循环的标志。 整个发送操作在事务内执行。

dsmSendObj 调用上的第三个参数是包含归档描述的缓冲区。 由于备份对象不具有描述，因此在备份对象时该参数为 NULL。

第 24 页的图 8 显示的示例展示了 **dsmBindMC** 函数调用的使用。

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

/*  Call dsmBindMC if not done previously */
objAttr.sizeEstimate.hi = 0;       /* estimate of        */
objAttr.sizeEstimate.lo = 32000;   /*        object size */
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;
   }
   /* Adjust the dataBlk buffer for the next piece to send */
}
rc = dsmEndSendObj(dsmHandle);
if (rc)
{
   printf("*** dsmEndSendObj failed: ");
   rcApiOut(dsmHandle, rc);
}
txn_reason = 0;
rc = dsmEndTxn(dsmHandle,           /* API session handle       */
            DSM_VOTE_COMMIT,      /* Commit transaction       */
            &txn_reason);         /* Reason if txn aborted    */
if (rc || txn_reason)
{
   printf("*** dsmEndTxn failed: rc = ");
   rcApiOut(dsmHandle, rc);
   printf("   reason =
}
```

图 *14.* 向服务器发送数据的示例

文件分组

IBM Spectrum Protect API 具有将多个单独对象关联在一起的逻辑文件分组协议。可以将这些组作为服务器上的逻辑组进行引用和管理。 逻辑组要求所有组成员和组长属于服务器上的同一节点和文件空间。

各逻辑组均有组长。 如果删除了组长，那么也会删除组。 您无法删除属于某个组的成员。 组中所有成员是否到期取决于组长。 例如，如果某个成员标记为到期，那么除非组长到期，否则该成员不会到期。 但是，如果成员未标记为到期，而组长到期，那么所有成员都一起到期。

文件组仅包含备份数据，不能包含归档数据。 如果应用程序需要，那么归档对象可以使用 **Archive Description** 字段来帮助进行某一类型的分组。

dsmGroupHandler 调用可将操作分组。 必须从事务中调用 **dsmGroupHandler** 函数。 大多数组错误情况均在 **dsmEndTxn** 或 **dsmEndTxnEx** 调用中捕获。

dsmEndTxnEx 中的 out 结构包括新字段 **groupLeaderObjId**。 此字段包含组长的对象标识（如果在该事务中打开了组）。 可以创建跨多个事务的组。 执行 close 之前，不会将组落实或保存在服务器上。
dsmGroupHandler 是可以接受五个不同操作的接口。 这些操作包括：

- DSM_GROUP_ACTION_OPEN
- DSM_GROUP_ACTION_CLOSE
- DSM_GROUP_ACTION_ADD
- DSM_GROUP_ACTION_ASSIGNTO
- DSM_GROUP_ACTION_REMOVE

第 51 页的表 15 列出了 **dsmGroupHandler** 函数调用操作：

表 *15. dsmGroupHandler* 函数

操作	**描述**
OPEN	OPEN 操作会创建一个组。 所发送的下一个对象会成为组长。 组长不能具有任何内容。 第一个对象后的所有对象都会成为添加到组中的成员。 要创建组，请打开组并传入唯一字符串以标识该组。 通过此唯一标识，可以打开多个同名组。 打开组后，所发送的下一个对象即是组长。 发送的所有其他对象为组成员。
CLOSE	CLOSE 操作将落实并保存打开的组。 要关闭组，请传入 open 操作中使用的对象名和唯一字符串。 应用程序必须检查是否存在打开的组，并根据需要关闭或删除组。 组在关闭之前不会落实或保存。 CLOSE 操作在以下情况下会失败： · 您所尝试关闭的组与现有打开的组同名。 · 当前已关闭的组与要关闭的同名新组之间存在管理类不兼容的情况。 在此情况下，请完成以下步骤： 1. 查询先前已关闭的组。 2. 如果现有已关闭的组的管理类不同于与当前打开的组关联的管理类，请发出类型为 DSM_BACKUPD_MC 的 **dsmUpdateObject**。 此命令会将现有组更新为新管理类。 3. 发出 CLOSE 操作。
ADD	ADD 操作将向组中附加对象。 在 ADD 操作后发送的所有对象都会指派到组。
ASSIGNTO	ASSIGNTO 操作准许客户机将存在于服务器上的对象指派到已声明的对等组。 此事务将建立 PEER 组关系。 ASSIGNTO 操作类似于 ADD 操作，但存在以下例外情况： · ADD 操作适用于正在进行的事务中的对象。 · ASSIGNTO 操作适用于服务器上的对象。
REMOVE	REMOVE 操作将除去组中的成员或成员列表。 不能除去组中的组长。 必须除去组成员，然后才能删除该成员。

将以下查询类型用于组支持：

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

qtBackupGroups 查询的是已关闭的组，而 **qtOpenGroups** 查询的是打开的组。 新类型的查询缓冲区包含 **groupLeaderObjId** 和 **objType** 字段。 查询根据这两个字段的值以不同方式执行。 下表包含一些查询可能性：

表 16. 查询示例

groupLeaderObjId.hi	groupLeaderObjId.lo	objType	结果
0	0	空	返回所有组长的列表
grpLdrObjId.hi	grpLdrObjId.lo	0	返回指派给指定组长 (**grpLdrObjId**) 的所有组成员的列表。
grpLdrObjId.hi	grpLdrObjId.lo	objType	通过针对分配到指定组长 (**grpLdrObjId**) 且与对象类型 (**objType**) 匹配的每个组成员使用 **BackQryRespEnhanced3**，返回一个列表。

来自 **dsmGetNextQObj** 的响应结构 (**qryRespBackupData**) 包括两个组支持字段：

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

这些字段是布尔标志。 以下示例显示了如何创建组，向组中添加成员，以及关闭组以将组落实到 IBM Spectrum Protect 服务器上。

```
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
```

图 15. 用于创建组的伪码示例

有关代码示例，请参阅样本组程序 `dsmgrp.c` （包含在 API `sampsrc` 目录中）。

从服务器接收数据

应用程序客户机可使用复原和检索功能从 IBM Spectrum Protect 存储器接收数据或指定对象及其关联数据。恢复功能可访问先前已备份的对象，而检索功能可访问先前已归档的对象。

限制: 通过 API 调用，API 只能恢复和检索已备份或归档的对象。

恢复和检索功能都以查询操作开始。 根据最初备份还是归档数据，查询将返回不同信息。 例如，对备份对象的查询将返回有关某个对象是否活动的信息，而对归档对象的查询则返回诸如对象描述之类的信息。 两种查询均返回用于唯一标识服务器上的对象的对象标识。

部分对象恢复或检索

应用程序客户机可仅检索对象的一部分。 这称为部分对象恢复或部分对象检索。

注意: 部分恢复或检索压缩或加密对象会导致不可预测的结果。

注: 如果将应用程序编码为使用部分对象恢复或检索，那么无法在发送数据时对其进行压缩。 要强制实行此操作，请将 *ObjAttr.objCompressed* 设置为 *bTrue*。

要执行部分对象恢复或检索，请将以下两个数据字段与每个对象 **GetList** 条目相关联:

偏移量
始从中返回数据的对象中的字节偏移量。

长度
要返回的对象字节数。

使用 DSM_MAX_PARTIAL_GET_OBJ 确定可以为某一特定 **dsmBeginGetData** 列表执行部分对象恢复或检索的最大对象数。

dsmBeginGetData 调用中使用的以下数据字段确定将要恢复或检索的对象部分:

- 如果偏移量和长度均为零，那么从 IBM Spectrum Protect 存储器中恢复或检索整个对象。
- 如果偏移量大于零，但长度为零，那么将恢复或检索从偏移量到结束的对象。
- 如果长度大于零，那么仅恢复或检索从偏移量到指定长度的对象部分。

恢复或检索数据

查询并通过 IBM Spectrum Protect 服务器建立会话后，可以运行恢复或检索数据的过程。

过程

要复原或检索数据，请完成以下步骤:

1. 为备份或归档数据查询 IBM Spectrum Protect 服务器。
2. 确定要从服务器恢复或检索的对象。
3. 在 **Restore Order** 字段中对对象进行排序。
4. 对所有想要访问的对象列表发送 **dsmBeginGetData** 调用。
5. 发送 **dsmGetObj** 调用以获取系统的每个对象。 每个对象可能需要多个 **dsmGetData** 调用才能获取所有相关联的对象数据。 在获取一个对象的所有数据后，发送 **dsmEndGetObj** 调用。
6. 在收到所有对象的所有数据或结束接收操作后，发送 **dsmEndGetData** 调用。

查询服务器

在您开始任何复原或检索操作之前，请首先查询 IBM Spectrum Protect 服务器以确定可以从存储器接收的对象。

要发送查询，应用程序必须输入 **dsmBeginQuery** 调用的参数列表和结构。 此结构必须包括查询所检查的文件空间以及高级别和低级别名称字段的模式匹配条目。 如果使用空用户名初始化会话，那么不需要指定 owner 字段。 但是，如果会话是使用显式所有者名称初始化的，将仅返回与该所有者名称相关联的对象。

时间点 **BackupQuery** 查询提供在某一特定时间的系统快照。 通过指定有效日期，可查询截至该时间已备份的所有文件。 即使对象具有更晚日期的活动备份，时间点仍会覆盖对象状态，以便返回先前的不活动副本。 有关更多信息，请参阅以下示例：pitDate。

查询将返回与对象一起存储的所有信息，以及下表中的信息。

表 *17.* 查询服务器返回信息

字段	描述
copyId	copyIdHi 和 copyIdLo 值提供一个 8 字节数字，用于在 IBM Spectrum Protect 存储器中唯一标识此节点的此对象。 使用此标识可从存储器中请求特定的对象以进行恢复或检索处理。
restoreOrderExt	restoreOrderExt 值提供以尽可能高效的方式从 IBM Spectrum Protect 存储器中接收对象的机制。 按此值将待复原对象排序，以确保磁带仅安装一次并且从前到后读取。

必须保留部分或所有查询信息以用于以后的处理。 保留 copyId 和 restoreOrderExt 字段，因为实际恢复操作中需要这两个字段。 您还必须保留打开数据文件或识别目标所需的任何其他信息。

调用 **dsmEndQuery** 以完成查询操作。

按恢复顺序选择对象并对其进行排序

执行备份或归档查询后，应用程序客户机必须确定要恢复或检索的对象（如果有）。

然后按升序（从低到高）将对象进行排序。 这种排序对于恢复操作的性能非常重要。 将 **restoreOrderExt** 字段上的对象进行排序可确保以最高效地顺序从服务器读取数据。

首先恢复磁盘上的所有数据，然后恢复要求卷安装的介质类（如磁带）上的数据。 **restoreOrderExt** 字段还确保按照处理开始于磁带的前端并向末尾进行的顺序读取磁带上的数据。

在 **restoreOrderExt** 字段上正确排序意味着不会发生重复磁带装入和不必要的倒带。

restoreOrderExt.top 字段中的非零值对应于 IBM Spectrum Protect 服务器上某个唯一串行存取设备。 由于一个串行存取设备一次只能由一个会话/安装点使用，因此应用程序应确保，如果使用多个会话，那么并发恢复不能具有相同 **restoreOrderExt**.top 值。 否则，第一个会话可以访问对象，但其他会话将等待直至第一个会话终止并且设备变得可用。

以下示例显示如何使用 **Restore Order** 字段将对象排序。

图 *16.* 通过 *restore order* 字段将对象排序

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

}  SortOrder;               /* struct used for sorting */

=====================================================================
/* the code for sorting starts from here */
dsmQueryType      queryType;
qryBackupData     queryBuffer;
DataBlk           qDataBlkArea;
qryRespBackupData qbDataArea;
dsInt16_t   rc;
dsBool_t  done = bFalse;
int i = 0;
int qry_item;
SortOrder  sortorder[100]; /* sorting can be done up to 100 items
                              only right now.  Set appropriate
                              array size to fit your needs */
/*-------------------------------------------------------------------+
| NOTE: Make sure that proper initializations have been done to
|  queryType,
|       queryBuffer, qDataBlkAre, and 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))
      {
         /*****************************************/
         /* transferring restoreOrderExt and objId */
         /*****************************************/
         sortorder[i].restoreOrderExt = qbDataArea.restoreOrderExt;
         sortorder[i].objId = qbDataArea.objId;

      } /*  if ((rc == DSM_RC_MORE_DATA) || (rc == DSM_RC_FINISHED)) */
      else
      {
```

```
            done = bTrue;
            /****************************/
            /* take appropriate action. */
            /****************************/
        }

        i++;
        qry_item++;

    } /* while (!done) */
    rc = dsmEndQuery(dsmHandle);
   /*check rc  */
    /*****************************************************/
    /* sorting the array using qsort.  After the call,   */
    /* sortorder will be sorted by restoreOrderExt field */
    /*****************************************************/

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

 /*-------------------------------------------------------------------+
| NOTE: Make sure to extract sorted object ids and store them in
|       any data structure you want.
-------------------------------------------------------------------*/

/*-------------------------------------------------------------------+
| int SortRestoreOrder(SortOrder *a, SortOrder *b)
|
| This function compares restoreOrder fields from two structures.
| 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);
}
```

启动 dsmBeginGetData 调用

选择要接收的对象并对其进行排序后，将其提交到 IBM Spectrum Protect 以进行复原或检索操作。**dsmBeginGetData** 调用开始恢复或检索操作。 对象按照请求的顺序返回到应用程序客户机。

为以下调用中的这两个参数完成信息：

mountWait
此参数告知服务器以下内容：应用程序客户机是等待脱机介质安装以获取对象的数据，还是应在恢复或检索操作期间跳过对象。

dsmGetObjListP
此参数是包含 **objId** 字段（即要恢复或检索的所有对象标识的列表）的数据结构。 每个 **objId** 均与一个 **partialObjData** 结构相关联，该结构描述了将检索整个 **objId** 还是仅检索对象的特定部分。

每个 **objId** 的长度均为八个字节，因此单个恢复或检索请求可包含数千个对象。 单个调用中请求的对象数限制为 DSM_MAX_GET_OBJ 或 DSM_MAX_PARTIAL_GET_OBJ。

接收每个要恢复或检索的对象

发送 **dsmBeginGetData** 调用后，您可以执行过程以接收服务器发出的每个对象。

关于此任务

DSM_RC_MORE_DATA 返回码表示返回了缓冲区，应再次调用 **dsmGetData**。 有关实际返回的字节数，请检查 **DataBlk.numBytes**。

获取一个对象的所有数据时，必须发送 **dsmEndGetObj** 调用。 如果将接收更多对象，请再次发送 **dsmGetObj**。

如果想要停止此过程，例如，要丢弃所有对象的尚未接收的复原流中的任何剩余数据，请发送 **dsmEndGetData** 调用。 此调用将数据从服务器清空到客户机。但是，使用此方法可能需要花费一点时间才能完成。 如果想要结束复原操作，请使用 **dsmTerminate** 以关闭会话。

过程

1. 发送 **dsmGetObj** 调用来识别从数据流请求的对象，并获取与该对象相关联的第一个数据块。
2. 如有必要，请发送更多 **dsmGetData** 调用来获取剩余的对象数据。

复原和检索的流程图示例

状态图和流程图可用于演示如何执行复原或检索操作。

从 “In Get Object” 指向 **dsmEndGetData** 的箭头表示您可以在调用 **dsmGetObj** 或 **dsmGetData** 后发送 **dsmEndGetData** 调用。 如果从 IBM Spectrum Protect 存储器获取对象时发生错误条件并且希望停止操作，那么可能需要执行此操作。 但是在某些情况下，请首先调用 **dsmEndGetObj**。

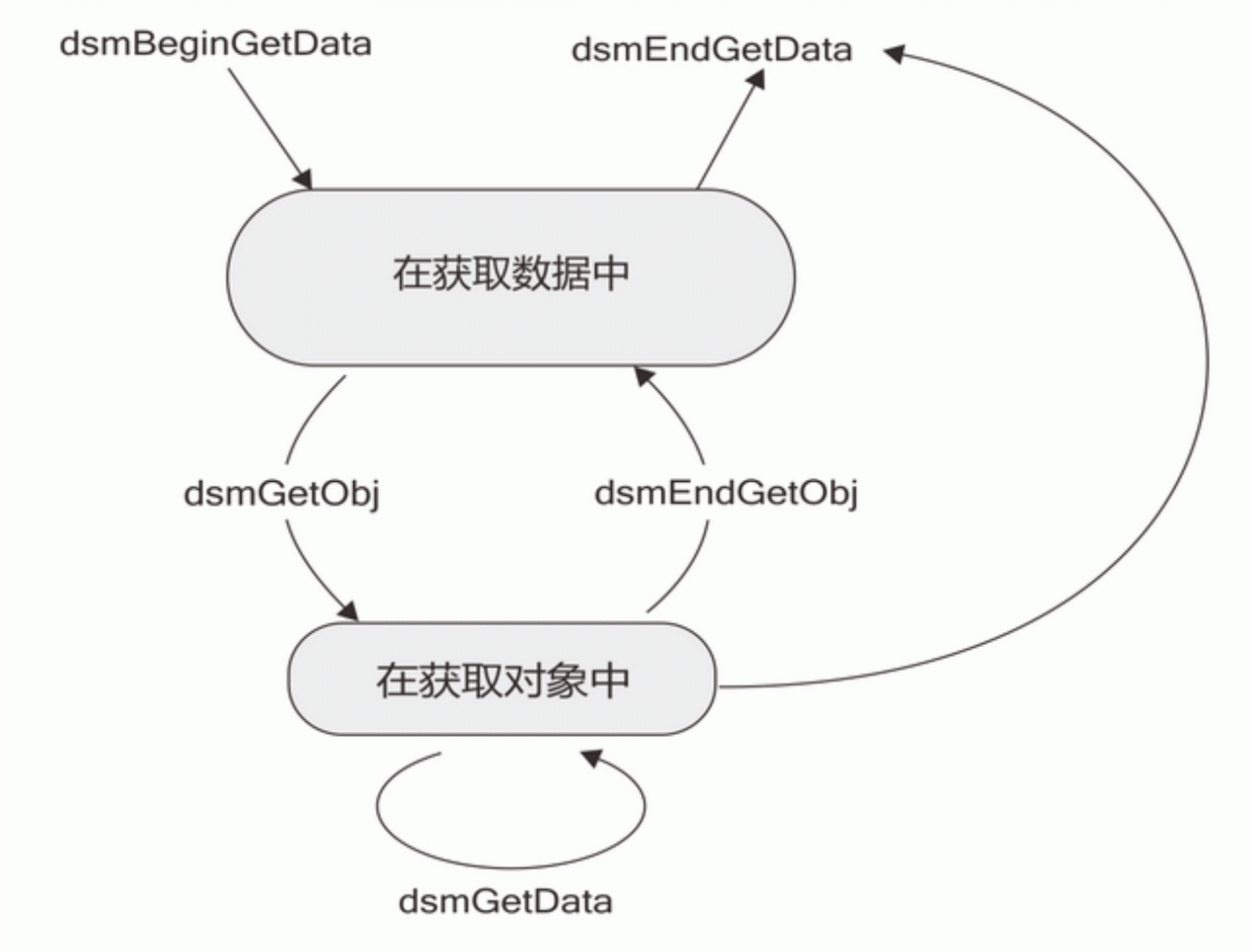


图 *17.* 复原和检索操作的状态图

第 58 页的图 18 显示执行恢复或检索操作的流程图。

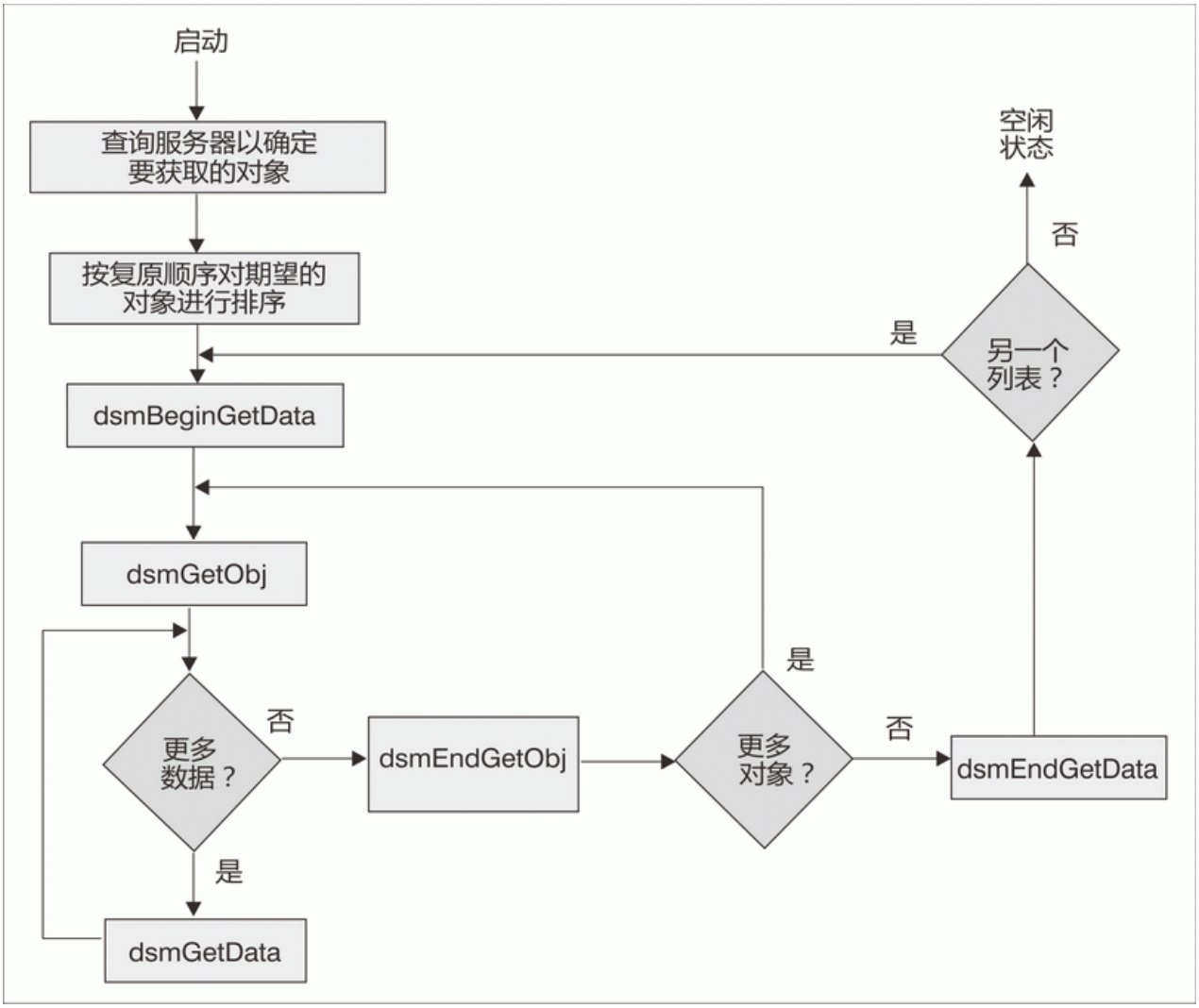


图 *18.* 复原和检索操作的流程图

从服务器接收数据的代码示例

此示例演示使用 API 函数从 IBM Spectrum Protect 存储器中检索数据。

dsmBeginGetData 函数调用显示在 switch 语句中，因此根据正在执行的是恢复还是检索操作，可以调用的参数也将有所不同。 **dsmGetData** 函数调用从循环的内部调用，该循环不断从服务器获取数据，直至设置了某个允许程序执行退出循环的标志。

图 *19.* 从服务器接收数据的示例

```
/* Call dsmBeginQuery and create a linked list of objects to restore. */
/* Process this list to create the proper list for the GetData calls. */
/* Set up the getList structure to point to this list.                */
/* This example is set up to perform a partial object retrieve.  To   */
/* retrieve only complete objects, set up:                            */
/*       getList.stVersion = dsmGetListVersion;                       */
/*       getList.partialObjData = NULL;                               */
dsmGetList getList;
getList.stVersion = dsmGetListPORVersion;   /* structure version      */
getList.numObjId  = items;               /* number of items in list   */
getList.objId     = (ObjID *)rest_ibuff;
                                    /* list of object IDs to restore  */
getList.partialObjData = (PartialObjData *) part_ibuff;
                                      /* list of partial object data  */
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;
}
/* Get each object from the list and verify whether it is on the  */
/* server.  If so, initialize structures with object attributes for */
/* data validation checks.  When done, call 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);
/* check rc from dsmEndGetObj */
/* check rc from dsmEndGetData */
rc = dsmEndGetData(dsmHandle);
return 0;
```

更新和删除服务器上的对象

您的 API 应用程序可使用 **dsmUpdateObj** 或 **dsmUpdateObjEx** 函数调用来更新已归档或备份的对象。 在会话状态中仅能使用一种调用，每次更新一个对象。 使用 **dsmUpdateObjEx** 可更新多个包含相同名称的归档对象中的任意对象。

要选择归档对象，请将 **dsmSendType** 函数调用设置为 **stArchive**。

- 通过 **dsmUpdateObj**，仅更新具有指定名称的最新归档对象。
- 通过 **dsmUpdateObjEx**，可通过指定正确的对象标识来更新归档对象。

对于归档对象，应用程序可更新以下字段：

- 描述
- 对象信息
- 所有者

要选择备份对象，请将 **dsmSendType** 设置为 **stBackup**。 对于备份对象，仅更新活动副本。

对于备份对象，应用程序可更新以下字段：

- 管理类
- 对象信息
- 所有者

从服务器删除对象

API 应用程序可以进行调用来删除已归档的对象或关闭已备份的对象。 删除归档对象取决于管理员在注册节点时授予的节点权限。管理员可以指定节点是否能够删除归档对象。

使用 **dsmDeleteObj** 函数调用可以删除归档对象并关闭备份对象。 使用此 **delType** 将从服务器除去备份对象。 这会基于 **objID** 来删除服务器数据库中的对象。 仅对象的所有者才能删除此对象。 您可以删除对象的任何版本（现行或非现行）。 服务器将协调这些版本。 如果删除现行版本的对象，那么第一个非现行版本将变为现行版本。 如果删除非现行版本的对象，那么所有更旧的版本都将前移。 必须注册具有 **backDel** 许可权的节点。

系统执行其下一个对象到期周期时，会将存储器中的归档对象标记为要删除。 一旦从服务器删除归档对象，便无法对其进行检索。

在服务器上停用备份对象时，此对象会从活动状态转换为不活动状态。 这些状态与基于所指派的管理类的不同保留策略相关联。

与 **dsmSendObj** 调用类似，**dsmDeleteObj** 调用也在事务边界内发送。 第 47 页的图 12 中的状态图显示如何将 **dsmDeleteObj** 调用置于 **dsmBeginTxn** 调用之后且在 **dsmEndTxn** 调用之前。

记录事件

API 应用程序可以将事件消息记录到中央位置。 应用程序可以将日志记录定向到 IBM Spectrum Protect 服务器和/或本地机器。 **dsmLogEventEx** 函数调用在会话内执行。 要查看服务器上记录的消息，请通过管理客户机使用 **query actlog** 查询命令。

如果应用程序将许多客户机消息写入到 dsmLogType 为 logLocal 或 logBoth 的客户机日志，请使用 IBM Spectrum Protect 客户机选项 errorlogretention 来修剪客户机错误日志文件。

有关 IBM Spectrum Protect 日志的更多信息，请参阅 IBM Spectrum Protect 服务器文档。

IBM Spectrum Protect API 的状态图摘要

复审通过 IBM Spectrum Protect API 创建您自己的应用程序的所有注意事项后，请复审整个应用程序的这一状态图摘要。

第 61 页的图 20 包含 API 的状态图。 除先前未显示的其他若干个调用外，它包含先前显示的所有状态图。

此图中的要点包括：

- 随时调用 **dsmQueryApiVersionEx**。 它不具有任何与其关联的状态。 请参阅第 13 页的图 1，以获取示例。
- 仅在 **dsmInitEx** 调用之前调用 **dsmQueryCliOptions**。
- 使用 **dsmRegisterFS**、**dsmUpdateFS** 和 **dsmDeleteFS** 管理文件空间。 这些调用从空闲会话状态中进行。 使用 **dsmBeginQuery** 调用来查询文件空间。 有关文件空间调用的更多信息，请参阅第 21 页的『管理文件空间』。
- 从空闲会话状态中或从发送对象事务状态中发送 **dsmBindMC** 调用。 请参阅第 24 页的图 8 中的示例。
- 从空闲会话状态中发送 **dsmChangePW** 调用。

  注: 如果 **dsmInitEx** 调用返回的内容带有密码过期返回码，那么必须在启动有效会话前进行 **dsmChangePW** 调用。 请参阅第 17 页的图 4，以获取使用 **dsmChangePW** 的示例。
- 如果调用返回的内容带有错误，那么状态保持不变。 例如，如果 **dsmGetObj** 返回的内容带有错误，那么状态保持为 In Get Data，并且 **dsmEndGetObj** 调用会发生调用顺序错误。

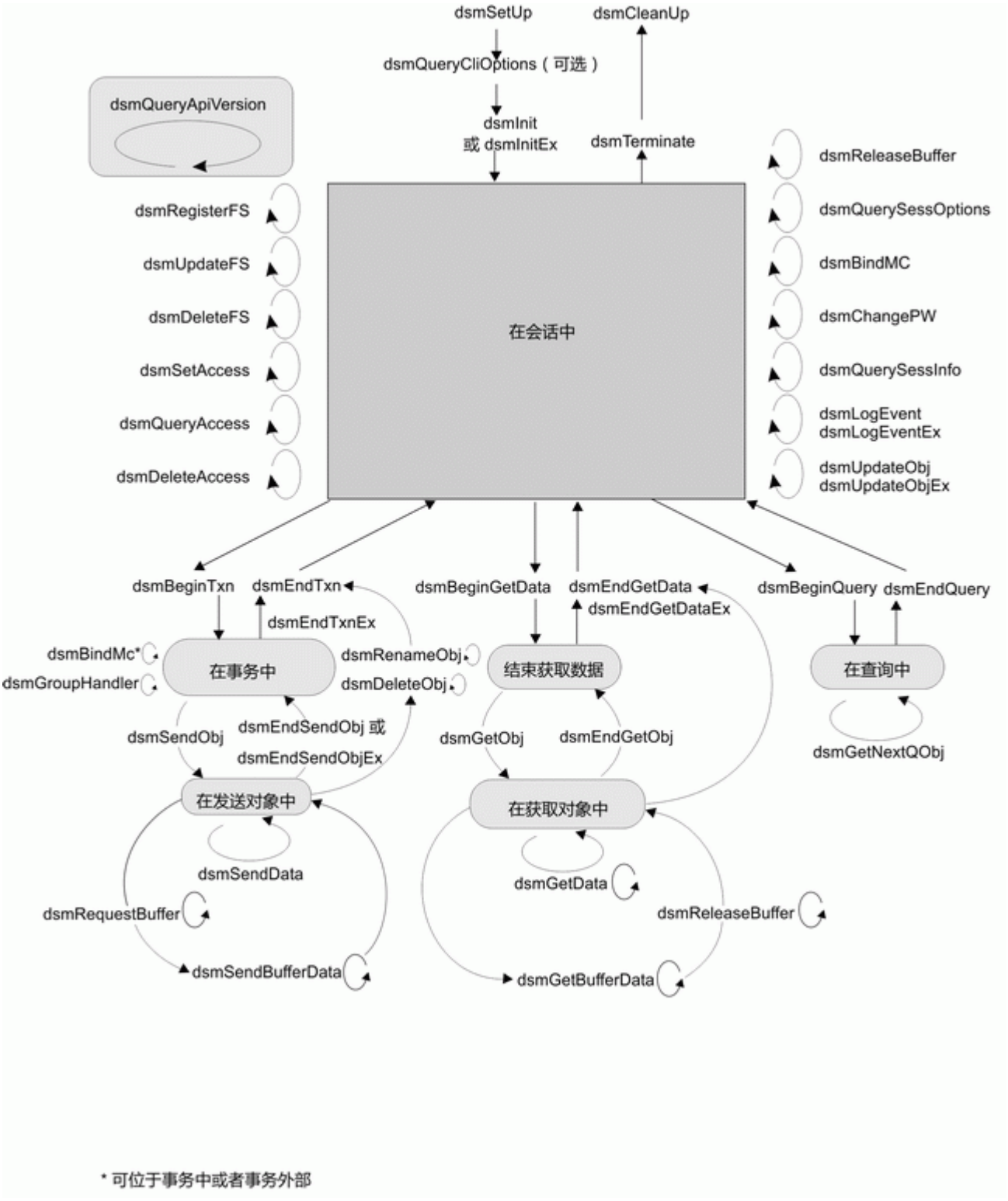


图 *20. API* 摘要状态图

第 4 章 了解互操作性

API 具有两种类型的互操作性：备份/归档客户机和 API 应用程序之间的互操作性和不同操作系统之间的互操作性。

备份/归档客户机互操作性

备份/归档命令行可以访问 API 对象以提供有限互操作性。 只能从备份/归档命令行客户机查看和访问 API，不能从图形界面进行查看和访问。 备份/归档命令行客户机只能复原文件内容而不能复原其他任何内容，因此应仅将其用于抢救类型的操作。

提供下列命令行操作：

- Delete archive
- Delete filespace
- 查询
- 恢复
- 检索
- Set access

路径信息是备份/归档客户机对象的实际目录。 相反，API 对象路径信息可能与现有目录没有任何关系：路径可能纯属人为设定。 互操作性不会更改这些对象类型的此方面。 要成功使用此特性，请遵循限制和约定。

注:

1. 备份/归档客户机和保留保护服务器上存储的 API 对象之间不存在互操作性。
2. 不能使用备份/归档客户机 GUI 来访问已使用 API 客户机存储的文件。 只能使用命令行来访问这些文件。

命名 API 对象

为 API 对象名建立一致的命名约定。 命名约定必须适应文件空间名称、高级限定符和低级限定符。 文件空间名称和高级限定符可以是指实际目录名称。 每个对象名可以包含适用于低级限定符的多个目录名称。

为了方便起见，请使用未以目录信息为前缀的对象名作为低级限定符。 有关更多信息，请参阅第 18 页的『对象名和标识』。

文件空间名称在从 API 或备份/归档命令行进行引用时必须是完全限定的。 例如，在 UNIX 或 Linux 操作系统上，您注册以下文件空间：

- /a
- /a/b

在您引用 /a 时，将显示仅与文件空间 /a 相关的对象。 要查看与 /a/b 相关的对象，您必须指定 /a/b 作为文件空间名称。

注册这两个文件空间后，如果将对象 b 备份到文件空间 /a 中，那么对 /a/b 的查询仍然显示仅与文件空间 /a/b 相关的对象。

当尝试通过 API 查询或删除文件空间时，此限制在文件空间引用中存在例外情况。 在这两种情况下，如果您使用通配符，那么文件空间名称不必是完全限定的。 例如，/a* 同时指代 /a 和 /a/b。

提示: 如果互操作性对您很重要，那么避免重叠的文件空间名称。

在 Windows 系统上，从备份/归档命令行界面访问 API 对象时，请将这些对象的文件空间名称括在花括号 { } 中。注册或引用文件空间名称时，Windows 操作系统会自动将这些名称置为大写字母形式。 但是，对于对象名规范的其余部分，不会发生此自动运行。如果需要完全互操作性，请在备份 API 对象时，在应用程

序中将高级限定符和低级限定符置为大写字母形式。 如果应用程序未在将对象发送到服务器之前大写高级限定符（目录名称）和低级限定符（文件名），那么您将无法直接通过备份/归档客户机按名称访问对象。

例如，如果对象在服务器上存储为 {"FileSpacename"}\TEST\MYDIRNAME\file.txt，那么您无法直接复原或查询 file.txt 对象，因为应用程序未在将文件复制到服务器之前大写文件名。处理这些对象的唯一方式是使用通配符。例如，要查询 \TEST\MYDIRNAME\file.txt，备份/归档客户机用户必须在将对象发送到服务器之前对于对象名称的所有部分使用通配符。必须使用以下命令来查询此 file.txt 文件：

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

如果还以小写文本形式保存任何其他限定符，那么也必须使用通配符来查询这些限定符。例如，要查询存储为 {"FileSpaceName"}\TEST\mydirname\file.txt 的对象，使用以下命令：

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

以下示例说明了这些概念。 在 Windows 以及 UNIX 或 Linux 环境中，无需指定完整的高级或低级限定符。但是，如果您不指定完整限定符，那么必须使用通配符。

平台	示例
Windows	要查询文件空间 MYFS 中的所有已备份文件，请输入以下字符串： `dsmc q ba "{MYFS}\*\*"` 对于每个高级和低级限定符，必须至少使用一个星号 (*)。
UNIX 或 Linux	要查询文件空间 /A 中的所有已备份文件，请输入以下字符串： `dsmc q ba "/A/*/*"` 对于每个高级和低级限定符，必须至少使用一个星号 (*)。

可以用于 API 的备份/归档客户机命令

您可以在应用程序中使用备份/归档客户机命令的子集。 例如，可以查看和管理其他用户在同一节点或不同节点上拥有的对象。

要查看和管理其他用户在同一节点或不同节点上拥有的对象，请执行下列步骤：

1. 使用 **set access** 命令提供访问权。
2. 指定所有者和节点。 从备份/归档命令行使用 *fromowner* 和 *fromnode* 选项指定所有者和节点。 例如：

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

第 64 页的表 18 描述了可用于 API 对象的命令。

表 *18.* 可以用于 *API* 对象的备份/归档客户机命令

命令	描述
delete archive	可以删除当前用户拥有的归档文件。 set access 命令设置对此命令没有任何影响。
Delete Filespace	**delete filespace** 命令会影响 API 对象。

表 18. 可以用于 *API* 对象的备份/归档客户机命令 (续)

命令	描述
查询	从备份/归档命令行可以查询备份和归档 API 对象及其他用户拥有或存在于其他节点上的对象。请参阅第 63 页的『命名 API 对象』，以获取有关查询 API 对象的信息。 使用现有 *–fromowner* 选项可以查询具有 set access 许可权的其他用户拥有的对象。 使用现有 *–fromnode* 选项可以查询授予 set access 许可权的其他节点上存在的对象。有关更多信息，请参阅第 96 页的『dsmInitEx』。
复原 **检索**	**注:** 请仅针对异常情况使用这些命令。如果加密密钥已知或保存在密码文件中，那么可以复原或检索使用由应用程序管理的密钥加密的 API 对象。使用备份/归档客户机无法复原或检索使用透明加密进行加密的 API 对象。 这些命令会将数据作为使用缺省文件属性创建的位文件来返回。可以复原或检索其他用户拥有或来自其他节点的 API 对象。 set access 命令可确定哪些对象合格。
设置访问权	**set access** 命令允许用户管理其他用户拥有或来自其他节点的 API 对象。

示例

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

```
          大小         备份日期                  管理类              A/I 文件
          ----         -----------               ----------          --- ----
            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         备份日期                  管理类              A/I 文件
          ----         -----------               ----------          --- ----
            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
```

操作系统互操作性

IBM Spectrum Protect API 支持跨平台互操作性。UNIX 或 Linux 系统上的应用程序可在从 Windows 系统备份的文件空间和对象上执行操作。类似，Windows 系统可在从 UNIX 或 Linux 系统备份的文件空间和对象上执行操作。

关于此任务

缺省情况下，来自于一个 UNIX 系统的对象名称与来自于其他 UNIX 系统的对象名称相兼容。缺省情况下，来自于 Windows 系统的对象名称不兼容来自于 UNIX 系统的对象名称。多个参数控制 IBM Spectrum Protect 文件空间中的对象命名。如果相应地设置应用程序，那么在 Windows 系统和 UNIX 系统上运行的应用程序都可使用对象的名称。使用相同参数来备份和复原对象。

限制: 使用 Unicode 的 Windows 应用程序创建的文件空间不兼容在 UNIX 系统上运行的应用程序。

过程

要以交互方式进行归档，请完成以下设置任务:

1. 建立一致的命名约定。 选择一个字符用于 `dir` 定界符，例如，正斜杠 (/) 或反斜杠 (\)。将目录定界符放置在文件空间名、高级限定符和低级限定符之前。
2. 在调用 **dsmInitEx** 时，将 **dirDelimiter** 字段的值设置为选中的目录定界符，并将 **bCrossPlatform** 设置为 **bTrue**。
3. 在使用 IBM Spectrum Protect 界面时，将 **useUnicode** 标志设置为 **bFalse**。 Unicode 文件名不兼容非 Unicode 文件名。

使用客户机节点代理支持备份多个节点

可以将共享存储器的多个节点的备份组合为 IBM Spectrum Protect 服务器上的公共目标节点名。 如果运行备份的系统可能随时间推移而变化（例如，带有某一集群），此方法将很有用。 您还可以使用 asnodename 选项从不同于运行了备份的系统的其他系统复原数据。

关于此任务

使用 **dsmInitEx** 选项字符串上的 asnodename 选项来备份、归档、复原和检索、查询或删除 IBM Spectrum Protect 服务器上的目标节点名下的数据。 还可以在 dsm.opt 或 dsm.sys 文件中指定 asnodename 选项。

限制: 不要将目标节点用作传统节点，特别是在将文件备份到服务器之前要加密这些文件的情况下。

过程

要启用此选项，请完成以下步骤：

1. 将 API 客户机安装在共享数据环境中的所有节点上。
2. 如果尚未注册，请向 IBM Spectrum Protect 服务器注册每个节点。 注册公共“目标”节点名以供共享数据环境中使用的每个代理节点共享。
3. 向服务器注册共享数据环境中的每个代理节点。代理节点名用于认证。 使用 asnodename 选项时，不会使用代理节点名存储数据。
4. 要求管理员使用 **grant proxynode** 命令向共享环境中的所有节点授予代理权限以访问 IBM Spectrum Protect 服务器上的目标节点名。
5. 使用 **query proxynode** 管理客户机命令以显示有权代表其他节点执行客户机操作的客户机节点。 该权限通过 **grant proxynode** 命令授予。 或者，使用查询类型为 **qtProxyNodeAuth** 的 **dsmQuery** 命令来查看此节点可代理的节点。
6. 如果应用程序要使用数据的用户加密（非 TSMENCRKEY），请确保所有节点都使用同一加密密钥。 必须将相同的加密密钥用于共享节点环境中备份的所有文件。

相关任务

使用客户机节点代理支持备份数据（UNIX 和 Linux 系统）

使用客户机节点代理支持备份数据（Windows 系统）

第 5 章 使用具有 Unicode 的 API

IBM Spectrum Protect API 支持 UnicodeUCS2 的固定长度的双字节代码页，这种代码页具有所有已知代码页（例如日语、中文或德语）的代码点。 它支持多达 65535 个唯一的代码点。

限制: 此功能仅在 Windows 上可用。

通过 Unicode，应用程序可从同一机器中备份和恢复使用任何字符集的文件名。 例如，在一台英语环境的机器上，可备份和恢复使用任何其他语言代码页的文件名。

在何种情况下使用 Unicode

可通过编写 Unicode 应用程序以及利用 IBM Spectrum Protect Unicode 接口来简化支持多语言的应用程序。

如果以下任何条件成立，那么使用 IBM Spectrum Protect Unicode 接口：

- 应用程序已针对 Unicode 进行了编译，并且在调用 IBM Spectrum Protect API 之前已经转换为多字节字符集 (mbcs)。
- 正在编写新应用程序并且希望使应用程序支持 Unicode。
- 应用程序使用的某个字符串是从使用 Unicode 的操作系统或其他应用程序传递到该应用程序的。

如果不需要使用 Unicode，那么无需再次编译应用程序。

API 继续支持 DSM 接口。 API SDK 包含 callmtu1.c 和 callmtu2.c 样本程序，这两个样本程序演示如何使用 Unicode API。 使用 **makemtu** 可编译这些程序。

设置 Unicode

要设置和使用 Unicode，必须执行特定步骤，以便 API 在服务器上注册 Unicode 文件空间，并且该文件空间中的所有文件名成为 Unicode 字符串。

限制: 不能在同一文件空间中存储 Unicode 和非 Unicode 文件名。

1. 使用 `-DUNICODE` 标志编译代码。
2. 应用程序中的所有字符串必须为 wchar 字符串。
3. 对于对 API 的调用，遵循 tsmapitd.h 文件中结构以及 tsmapifp.h 文件中的函数定义。
4. 在 **tsmInitEx** 函数调用上将 *useUnicode* 标志设置为 *bTrue*。 任何新文件空间均注册为 Unicode 文件空间。

向先前注册的非 Unicode 文件空间发送数据时，API 继续将文件名作为非 Unicode 发送。 将服务器上的旧文件空间重命名为 fsname_old 并为新数据启动新的 Unicode 文件空间。 API 从旧文件空间恢复非 Unicode 数据。 使用查询文件空间上返回的 **tsmQryRespFSData** 结构中的 **bIsUnicode** 字段确定文件空间是否为 Unicode。

每个 **dsmXXX** 函数调用均具有一个匹配的 **tsmXXX** 函数调用。 两者之间的区别在于所使用的结构。 在使用 UNICODE 标志编译所有 **tsmXXX** 函数调用结构时，它们均具有 dsChar_t 类型（对于字符串值）。dsChar_r 映射到 wchar。 这两个接口之间没有任何区别。

限制: 可使用其中任一接口。 请勿混用 **dsmXXX** 函数调用和 **tsmXXX** 函数调用接口。请确保使用 IBM Spectrum Protect 结构和 IBM Spectrum Protect 版本定义。

部分常量继续在 dsmapitd.h 文件中定义，因此在编译时您需要 dsmapitd.h 和 tsmapitd.h 这两个文件。

可在其他操作系统上使用 IBM Spectrum Protect 接口，如 UNIX 或 Linux，但在这些操作系统上 dsChar_t 类型映射到 char，因为仅 Windows 操作系统上支持 Unicode。 可以仅编写应用程序的一个变体，并使用 IBM Spectrum Protect 接口在多个操作系统上进行编译。如果正在编写新应用程序，请使用 IBM Spectrum Protect 接口。

如果正在升级现有应用程序：

1. 将 **dsmXXX** 函数调用结构和调用转换为 IBM Spectrum Protect 接口。
2. 迁移现有文件空间。
3. 在将 *useUnicode* 标志设置为 *true* 的情况下备份新文件空间。

注: 在使用支持 Unicode 的客户机访问节点后，无法使用较旧版本的 API 或使用其他操作系统中的 API 来连接到同一节点。 如果您的应用程序使用交叉平台功能，请勿使用 Unicode 标志。 在 Unicode 和非 Unicode 操作系统之间没有交叉平台支持。

启用 *useUnicode* 标志时，所有字符串结构均视为 Unicode 字符串。 在服务器上，仅以下字段为真正的 Unicode：

· 文件空间名称
· 高级别
· 低级别
· 归档描述

所有其余字段在被发送到服务器之前将被转换为本地代码页。 诸如节点名之类的字段为 wchar 字符串。 它们必须在当前语言环境中有效。 例如，在日语环境的机器上，可使用中文名称备份文件，但是节点名必须是日语中的有效字符串。 选项文件仍在当前代码页中。 如果需要创建 Unicode 包含/排除 列表，请使用带有文件名的 *inclexcl* 选项，并使用其中的 Unicode 模式创建一个 Unicode 文件。

相关参考
inclexcl 选项

第 6 章 API 函数调用

第 69 页的表 19 提供 API 函数调用的字母顺序列表、简短描述以及可获取有关函数调用的更详细信息的位置，其中包括：

Element	描述
用途	描述函数调用。
语法	包含函数调用的实际 C 代码。 此代码从 UNIX 或 Linux 版本的 dsmapifp.h 头文件复制而来. 请参阅第 171 页的『附录 C API 函数定义源文件』。 此文件在其他操作系统上略有不同。 其他操作系统的应用程序程序员应检查其头文件 dsmapifp.h 的版本，以了解 API 定义的确切语法。
参数	描述函数调用中的各参数，根据其使用方式将其识别为输入 (I) 或输出 (O)。 部分参数同时指定为输入和输出 (I/O)。 此部分中引用的数据类型在 dsmapitd.h 头文件中进行定义。 请参阅第 133 页的『附录 B API 类型定义源文件』。
返回码	包含特定于函数调用的返回码列表。 未列出可能出现在任何调用中的常规系统错误（如通信错误、服务器问题或用户错误）。 返回码在 dsmrc.h 头文件中进行定义。 请参阅第 123 页的『附录 A API 返回码源文件：dsmrc.h』。

表 *19. API* 函数调用

函数调用和位置	描述
第 71 页的『dsmBeginGetData』	启动对存储器中对象列表的复原或检索操作。
第 72 页的『dsmBeginQuery』	启动对 IBM Spectrum Protect 的查询请求以获取信息。
第 76 页的『dsmBeginTxn』	启动一个或多个开始完整操作的事务。 所有操作要么都成功，要么都失败。
第 77 页的『dsmBindMC』	将管理类与传递的对象相关联或绑定。
第 78 页的『dsmChangePW』	更改 IBM Spectrum Protect 密码。
第 79 页的『dsmCleanUp』	如果已调用 **dsmSetUp**，那么会使用此调用。
第 79 页的『dsmDeleteAccess』	删除对象的备份版本或归档副本的当前授权规则。
第 80 页的『dsmDeleteFS』	删除存储器中的文件空间。
第 81 页的『dsmDeleteObj』	关闭备份对象或删除存储器中的归档对象。
第 82 页的『dsmEndGetData』	结束用于从存储器中获取对象的 **dsmBeginGetData** 会话。
第 82 页的『dsmEndGetDataEx』	提供已发送的不依赖 LAN 的总字节数。
第 83 页的『dsmEndGetObj』	结束用于获取指定对象的数据的 **dsmGetObj** 会话。
第 83 页的『dsmEndQuery』	表示 **dsmBeginQuery** 操作结束。
第 84 页的『dsmEndSendObj』	指示发送到存储器的数据结束。
第 84 页的『dsmEndSendObjEx』	提供压缩信息和已发送字节数。
第 85 页的『dsmEndTxn』	结束 IBM Spectrum Protect 事务。

表 *19. API* 函数调用 *(续)*

函数调用和位置	描述
第 86 页的『dsmEndTxnEx』	提供要用于 **dsmGroupHandlerfunction** 调用的组长对象标识信息。
第 87 页的『dsmGetData』	从 IBM Spectrum Protect 获取数据的字节流，并将其放置在调用者的缓冲区中。
第 88 页的『dsmGetBufferData』	从 IBM Spectrum Protect 服务器获取 IBM Spectrum Protect 分配的数据缓冲区。
第 89 页的『dsmGetNextQObj』	从先前的 **dsmBeginQuery** 调用中获取下一个查询响应，并且将其放置在调用者的缓冲区中。
第 91 页的『dsmGetObj』	从数据流中获取所请求的对象数据，并且将其放置在调用者的缓冲区中。
第 92 页的『dsmGroupHandler』	根据给定的输入对逻辑文件组执行操作。
第 93 页的『dsmInit』	启动 API 会话并将客户机连接到存储器。
第 96 页的『dsmInitEx』	使用其他准许扩展验证的参数来启动 API 会话。
第 100 页的『dsmLogEvent』	将用户消息记录到服务器日志文件和/或本地错误日志。
第 100 页的『dsmLogEventEx』	将用户消息记录到服务器日志文件和/或本地错误日志。
第 101 页的『dsmQueryAccess』	查询服务器以获取对象的备份版本或归档副本的所有访问授权规则。
第 102 页的『dsmQueryApiVersion』	针对应用程序客户机访问的 API 库版本执行查询请求。
第 102 页的『dsmQueryApiVersionEx』	针对应用程序客户机访问的 API 库版本执行查询请求。
第 103 页的『dsmQueryCliOptions』	查询用户选项文件中的重要选项值。
第 104 页的『dsmQuerySessInfo』	启动对 IBM Spectrum Protect 的查询请求，以获取与 **dsmHandle** 中指定会话的操作相关的信息。
第 104 页的『dsmQuerySessOptions』	查询 **dsmHandle** 中指定会话中有效的重要选项值。
第 105 页的『dsmRCMsg』	获取与 API 返回码关联的消息文本。
第 106 页的『dsmRegisterFS』	向服务器注册新文件空间。
第 107 页的『dsmReleaseBuffer』	返回 IBM Spectrum Protect 分配的缓冲区。
第 108 页的『dsmRenameObj』	对高级或低级对象名进行重命名。
第 109 页的『dsmRequestBuffer』	获取 IBM Spectrum Protect 分配的缓冲区以进行缓冲区副本消除。
第 110 页的『dsmRetentionEvent』	将对象标识列表发送到服务器，并且将对这些对象执行保留事件操作。
第 111 页的『dsmSendBufferData』	从 IBM Spectrum Protect 分配的缓冲区发送数据。
第 112 页的『dsmSendData』	通过缓冲区将数据的字节流发送到 IBM Spectrum Protect。
第 113 页的『dsmSendObj』	启动请求以将单个对象发送到存储器。

表 19. API 函数调用 (续)

函数调用和位置	描述
第 116 页的『dsmSetAccess』	为其他用户或节点提供对对象的备份版本或归档副本的访问权、对所有对象的访问权或对选择性集合的访问权。
第 117 页的『dsmSetUp』	覆盖环境变量值。
第 118 页的『dsmTerminate』	结束服务器会话并清除 IBM Spectrum Protect 环境。
第 118 页的『dsmUpdateFS』	更新存储器中的文件空间。
第 119 页的『dsmUpdateObj』	更新与服务器上已有的活动备份对象关联的 objInfo 信息，或者更新归档对象。
第 121 页的『dsmUpdateObjEx』	更新与特定归档对象关联的 objInfo 信息（即使有多个同名对象也如此），或者更新活动备份对象。

相关参考

API 返回码

dsmBeginGetData

dsmBeginGetData 函数调用将对存储器中的对象列表启动恢复或检索操作。 此对象列表包含在 **dsmGetList** 结构中。 应用程序将使用调用 **dsmBeginGetData** 之前执行的查询中的值来创建此列表。

调用者必须首先使用从对象查询获取的恢复顺序字段来对此调用中包含的列表排序。 这将确保尽可能以最有效的方法从存储器中恢复对象，而无需对数据磁带进行倒带或重新安装。

获取整个对象时，最大 *dsmGetList.numObjID* 为 DSM_MAX_GET_OBJ。 获取部分对象时，最大值为 DSM_MAX_PARTIAL_GET_OBJ。

在对 **dsmBeginGetData** 的调用之后执行一个或多个对 **dsmGetObj** 的调用可获取列表中的每个对象。 获取每个对象或不再需要对象的其他数据之后，将发送 **dsmEndGetObj** 调用。

获取所有对象或取消 **dsmEndGetObj** 时，将发送 **dsmEndGetData** 调用。 然后，可以再次开始循环。

语法

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

参数

dsUint32_t dsmHandle (I)
将此调用与之前的 **dsmInitEx** 调用关联的句柄。

dsBool_t mountWait (I)
布尔值 true 或 false 指示，在所需数据当前脱机的情况下，应用程序客户机是否等待安装脱机介质。 如果 mountWait 为 true，那么应用程序将等待服务器安装所需介质。 直到介质安装完成或请求取消，应用程序将一直处于等待状态。

dsmGetType getType (I)
一种枚举类型，包括 **gtBackup** 和 **gtArchive**，指示要获取的对象类型。

dsmGetList *dsmGetObjListP (I)
该结构包含有关要复原或检索的对象或部分对象的信息。 该结构指向对象标识的列表，如果是部分对象复原或检索，还指向关联的偏移量和长度的列表。 如果应用程序使用部分对象恢复或检索功能，请将

dsmGetList.stVersion 字段设置为 **dsmGetListPORVersion**。 在部分对象恢复或检索功能中，发送数据时无法进行压缩。 要强制实施此操作，请将 **ObjAttr.objCompressed** 设置为 *bTrue*。

请参阅第 58 页的图 19 和第 133 页的『附录 B API 类型定义源文件』以获取有关此结构的更多信息。

请参阅第 52 页的『部分对象恢复或检索』以获取有关部分对象恢复或检索的更多信息。

返回码

返回码编号用括号 ( ) 括起。

表 *20. dsmBeginGetData* 的返回码

返回码	说明
DSM_RC_ABORT_INVALID_OFFSET (33)	部分对象检索期间指定的偏移量大于对象的长度。
DSM_RC_ABORT_INVALID_LENGTH (34)	部分对象检索期间指定的长度大于对象的长度，否则长度加上偏移量会延长到超出对象的尾部。
DSM_RC_NO_MEMORY (102)	没有剩余的 RAM 来完成请求。
DSM_RC_NUMOBJ_EXCEED (2029)	**dsmGetList.numObjId** 大于 DSM_MAX_GET_OBJ。
DSM_RC_OBJID_NOTFOUND (2063)	找不到对象标识。 该对象未恢复。
DSM_RC_WRONG_VERSION_PARM (2065)	应用程序客户机的 API 版本不同于 IBM Spectrum Protect 库版本。

dsmBeginQuery

dsmBeginQuery 函数调用将启动对服务器的查询请求，以获取有关数据、文件空间和管理类的信息。

具体来说，**dsmBeginQuery** 可以查询：

- 归档数据
- 备份数据
- 活动备份数据
- 文件空间
- 管理类

从调用返回的查询数据由对 **dsmGetNextQObj** 的一个或多个调用获取。 查询完成时，将发送 **dsmEndQuery** 调用。

语法

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

参数

dsUint32_t dsmHandle (I)

将此调用与之前的 **dsmInitEx** 调用关联的句柄。

dsmQueryType queryType (I)

标识要运行的查询类型。 分配以下某个选项：

qtArchive

查询归档对象。

qtBackup

查询备份对象。

qtBackupActive

仅查询您传递的整个文件空间名称的活动备份对象。 此查询称为“快速路径”，是从存储器查询活动对象的高效方法。

先决条件: 在 UNIX 或 Linux 操作系统上，您必须以 root 用户身份登录。

qtFilespace

查询注册的文件空间。

qtMC

查询定义的管理类。

qtBackupGroups

查询关闭的组。

qtOpenGroups

查询打开的组。

qtProxyNodeAuth

查询此节点可以代理的节点。

qtProxyNodePeer

查询具有相同目标的对等节点。

dsmQueryBuff *queryBuffer (I)

识别指向映射到特定数据结构的缓冲区的指针。 此结构与所传递的查询类型关联。 这些结构包括每个查询类型的选择条件。 在每个结构中填写字段以指定要运行的查询的作用域。 每个结构中的 stVersion 字段包含结构版本号。

数据结构及其相关字段包含以下项:

qryArchiveData

objName

完整的对象名。 可以在名称的高级或低级部分使用通配符，如星号 (*) 或问号 (?)。 星号与 0 个或 0 个以上的字符匹配，而问号与一个字符匹配。 objName 的 objType 字段可以具有以下某个值:

- DSM_OBJ_FILE
- DSM_OBJ_DIRECTORY
- DSM_OBJ_ANY_TYPE

有关高级和低级名称的更多信息，请参阅以下主题: 第 19 页的『高级名称和低级名称』。

owner

对象的所有者名称。

insDateLowerBound

已归档对象的插入日期的下边界。 对于无限制的下边界，将 year 组件设置为 DATE_MINUS_INFINITE。

insDateUpperBound

已归档对象的插入日期的上边界。 对于无限制的上边界，将 year 组件设置为 DATE_PLUS_INFINITE。

expDateLowerBound

截止日期的下边界。 对于无限制的下边界，将 year 组件设置为 DATE_MINUS_INFINITE。

expDateUpperBound

到期日期的上边界。 要匹配 NOLIMIT 的管理类 **RETVer** 设置，请将 year 组件设置为 DATE_PLUS_INFINITE。

descr

归档描述。 输入星号 (*) 以搜索所有描述。

qryBackupData

objName

完整的对象名。 可以在名称的高级或低级部分使用通配符，如星号 (*) 或问号 (?)。 星号与 0 个或 0 个以上的字符匹配，而问号与一个字符匹配。 objName 的 objType 字段可以具有以下某个值：

- DSM_OBJ_FILE
- DSM_OBJ_DIRECTORY
- DSM_OBJ_ANY_TYPE

有关高级和低级名称的更多信息，请参阅以下主题：第 19 页的『高级名称和低级名称』。

owner

对象的所有者名称。

objState

您可以查询以下某个对象状态：

- DSM_ACTIVE
- DSM_INACTIVE
- DSM_ANY_MATCH

pitDate

时间点值。 具有此字段的查询将返回此日期和时间之前备份的最近对象。 objState 可以为 active 或 inactive。 不会返回在 pitDate 之前删除的对象。 例如：

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

在周五，使用时间点值星期三 12:00:00 a.m. 调用查询。此调用将返回以下信息：

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

调用不会返回 DEF，因为该对象在该时间点值之前已删除。

qryABackupData

objName

完整的对象名。 可以在名称的高级或低级部分使用通配符，如星号 (*) 或问号 (?)。 星号与 0 个或 0 个以上的字符匹配，而问号与一个字符匹配。 objName 的 objType 字段可以具有以下某个值：

- DSM_OBJ_FILE
- DSM_OBJ_DIRECTORY
- DSM_OBJ_ANY_TYPE

有关高级和低级名称的更多信息，请参阅以下主题：第 19 页的『高级名称和低级名称』。

qryFSData

fsName

在此字段中输入特定文件空间的名称，或输入星号 (*) 以检索有关所有注册文件空间的信息。

qryMCData

mcName

输入特定管理类的名称，或输入空字符串 (“ ”) 以检索有关所有管理类的信息。

注: 不能使用星号 (*)。

mcDetail

确定是否返回有关管理类的备份和归档副本组的信息。 以下值有效：

- bTrue
- bFalse

qryBackupGroup：

groupType

组类型为 DSM_GROUPTYPE_PEER。

fsName

文件空间名称。

所有者

所有者标识。

groupLeaderObjId

组长对象标识。

objType

对象类型。

qryProxyNodeAuth：

targetNodeName

目标节点名。

peerNodeName

对等节点名。

hlAddress

高级名称的对等地址。

llAddress

低级名称的对等地址。

qryProxyNodePeer：

targetNodeName

目标节点名。

peerNodeName

对等节点名。

hlAddress

高级名称的对等地址。

llAddress

低级名称的对等地址。

返回码

下表描述了 **dsmBeginQuery** 函数调用的返回码。

表 *21. dsmBeginQuery* 的返回码

返回码	返回码编号	说明
DSM_RC_NO_MEMORY	102	没有足够内存来完成请求。
DSM_RC_FILE_SPACE_NOT_FOUND	124	找不到指定文件空间。
DSM_RC_NO_POLICY_BLK	2007	服务器策略信息不可用。
DSM_RC_INVALID_OBJTYPE	2010	对象类型无效。
DSM_RC_INVALID_OBJOWNER	2019	对象所有者名称无效。
DSM_RC_INVALID_OBJSTATE	2024	对象条件无效。
DSM_RC_WRONG_VERSION_PARM	2065	应用程序客户机的 API 版本不同于 IBM Spectrum Protect 库版本。

dsmBeginTxn

dsmBeginTxn 函数调用将开始进行一个或多个 IBM Spectrum Protect 事务（此类事务将开始进行完整的操作）；要么所有操作都成功，要么都失败。 操作可以是单个调用，也可以是一系列调用。 例如，后跟多个 **dsmSendData** 调用的一个 **dsmSendObj** 调用可以视为单个操作。 类似地，带有 **dataBlkPtr**（指示包含要备份的对象的数据区域）的 **dsmSendObj** 调用也会被视为单个操作。

在单个事务中尝试将多个对象组合在一起以进行数据传输操作。 对象分组会大大改善 IBM Spectrum Protect 系统的性能。 从客户机和服务器的角度来看，启动和结束每个事务都会引发一定量的开销。

对于单个事务中可以执行的操作，存在限制。 这些限制包括：

- 可以在单个事务中发送或删除的最大对象数。 此限制位于 **dsmQuerySessInfo** 在 *ApiSessInfo.maxObjPerTxn* 字段中返回的数据中。 它对应于 *TxnGroupMax* 服务器选项。
- 在单个事务中发送到服务器的所有对象（备份或归档）必须具有同一副本目标（在对象的管理类绑定中定义）。 此值位于 **dsmBindMC** 在 **mcBindKey.backup_copy_dest** 或 **mcBindKey.archive_copy_dest** 字段中返回的数据中。

使用 API，应用程序客户机可以监视和控制这些限制，或者 API 可以监视这些限制。 如果 API 正在监视限制，那么 API 调用的相应返回码会在达到一个或多个限制时通知应用程序客户机。

始终使 **dsmBeginTxn** 调用和 **dsmEndTxn** 调用匹配，以优化一对 **dsmBeginTxn** 和 **dsmEndTxn** 调用的操作集。

语法

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

参数

dsUint32_t dsmHandle (I)
将此调用与之前的 **dsmInitEx** 调用相关联的句柄。

返回码

返回码编号用括号 ( ) 括起。

表 *22. dsmBeginTxn* 的返回码

返回码	说明
DSM_RC_ABORT_NODE_NOT_AUTHORIZED (36)	TXN 操作不允许 FROMNODE 或 FROMOWNER。

dsmBindMC

dsmBindMC 函数调用可将管理类关联或绑定到传递的对象。 该对象通过包含/排除列表传递到选项文件中。 如果在特定管理类的包含列表中找不到匹配项，那么将指定缺省管理类。 排除列表会阻止对象备份，但不会阻止其归档。

应用程序客户机可以使用 mcBindKey 结构中返回的参数，来确定是应该备份还是归档此对象，或者是否必须启动新事务（因为副本目标不同）。 请参阅 **dsmBeginTxn** 以获取更多信息。

先调用 **dsmBindMC**，再调用 **dsmSendObj**，因为每个对象都必须有与其关联的管理类。 此调用可以在事务内或事务外执行。 例如，在多对象事务内，如果 **dsmBindMC** 指示该对象的副本目标与先前对象不同，那么必须结束该事务并启动新的事务。 在此情况下，不需要另一 **dsmBindMC**，因为已对该对象执行了一次。

语法

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

参数

dsUint32_t dsmHandle (I)
将此调用与之前的 **dsmInitEx** 调用相关联的句柄。

dsmObjName *objNameP (I)
指向包含文件空间名称、高级对象名、低级对象名和对象类型的结构的指针。

dsmSendType sendType (I)
确定是否为归档或备份发送执行此管理类绑定。 此调用可能的值包括：

Name	描述
stBackup	备份对象
stArchive	归档对象
stBackupMountWait	备份对象

Name	描述
stArchiveMountWait	归档对象

对于 **dsmBindMC** 调用，stBackup 和 stBackupMountWait 是等效的，stArchive 和 stArchiveMountWait 是等效的。

mcBindKey *mcBindKeyP (O)
这是返回管理类信息的 mcBindKey 结构的地址。 应用程序客户机可以使用此处返回的信息来确定此对象在多对象事务中是否适合，或对绑定到对象的管理类执行管理类查询。

返回码

返回码编号用括号 ( ) 括起。

表 *23. dsmBindMC* 的返回码

返回码	说明
DSM_RC_NO_MEMORY (102)	没有剩余的 RAM 来完成请求。
DSM_RC_INVALID_PARM (109)	传递的某个参数具有无效值。
DSM_RC_TL_EXCLUDED (185)	备份对象已排除，无法发送。
DSM_RC_INVALID_OBJTYPE (2010)	对象类型无效。
DSM_RC_INVALID_SENDTYPE (2022)	发送类型无效。
DSM_RC_WRONG_VERSION_PARM (2065)	应用程序客户机 API 版本不同于 IBM Spectrum Protect 库版本。

dsmChangePW

dsmChangePW 函数调用可更改 IBM Spectrum Protect 密码。在多用户操作系统（如 UNIX 或 Linux）上，仅 root 用户或授权用户可以使用此调用。

在 Windows 操作系统上，您可以在 dsm.opt 文件中指定密码。 在此情况下，**dsmChangePW** 不会更新 dsm.opt 文件。 对 **dsmChangePW** 进行调用后，必须单独更新 dsm.opt 文件。

如果 **dsmInitEx** 返回 DSM_RC_VERIFIER_EXPIRED，那么此调用必须成功处理。 如果 **dsmChangePW** 调用在此情况下失败，那么会话将结束。

如果出于其他某种原因调用了 **dsmChangePW**，那么无论返回码是什么，会话都会保持打开。

语法

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

参数

dsUint32_t dsmHandle (I)
将此调用与之前的 **dsmInitEx** 调用相关联的句柄。

char *oldPW (I)
调用者的旧密码。 最大长度是 DSM_MAX_VERIFIER_LENGTH。

char *newPW (I)
调用者的新密码。 最大长度是 DSM_MAX_VERIFIER_LENGTH。

返回码

返回码编号用括号 ( ) 括起。

表 *24. dsmChangePW* 的返回码

返回码	说明
DSM_RC_ABORT_BAD_VERIFIER (6)	输入的密码不正确。
DSM_RC_AUTH_FAILURE (137)	认证失败。 旧密码不正确。
DSM_RC_NEWPW_REQD (2030)	必须输入新密码的值。
DSM_RC_OLDPW_REQD (2031)	必须输入旧密码的值。
DSM_RC_PASSWD_TOOLONG (2103)	指定的密码过长。
DSM_RC_NEED_ROOT (2300)	API 调用者必须是 root 用户或授权用户。

dsmCleanUp

如果已调用 **dsmSetUp**，那么会使用 **dsmCleanUp** 函数调用。 **dsmCleanUp** 函数调用应在 **dsmTerminate** 之后进行调用。 调用 **dsmCleanUp** 之后，不能进行任何其他调用。

不存在特定于此调用的返回码。

语法

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

参数

dsBool_t mtFlag (I)
此参数指定 API 已在单线程或多线程方式下使用。 可能的值包括：

- DSM_SINGLETHREAD
- DSM_MULTITHREAD

dsmDeleteAccess

dsmDeleteAccess 函数调用将删除对象的备份版本或归档副本的当前授权规则。 删除授权规则时，将撤销用户对于规则指定的任何文件具有的访问权。

使用 **dsmDeleteAccess** 时，一次只能删除一个规则。 通过 **dsmQueryAccess** 命令获取规则标识。

不存在特定于此调用的返回码。

语法

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

参数

dsUint32_t dsmHandle (I)
将此调用与之前的 **dsmInitEx** 调用关联的句柄。

dsUint32_t ruleNum (I)
删除的访问权规则的规则标识。 此值从 **dsmQueryAccess** 函数调用获取。

dsmDeleteFS

dsmDeleteFS 函数调用将从存储器删除文件空间。 要删除文件空间，必须具有您的 IBM Spectrum Protect 管理员授予您的相应许可权。 要确定是否具有必需的许可权，请调用 **dsmQuerySessInfo**。 此函数调用将返回 *ApiSessInfo* 类型的数据结构，该结构包括两个字段：*archDel* 和 *backDel*。

注:

- 在 UNIX 或 Linux 操作系统上，仅 root 用户或授权用户可删除文件空间。
- 如果需要删除的文件空间包含备份版本，那么必须具有备份删除权限 (***backDel*** = BACKDEL_YES)。 如果文件空间包含归档副本，那么必须具有归档删除权限 (*archDel* = ARCHDEL_YES)。 如果文件空间同时包含备份版本和归档版本，那么必须同时具有这两种删除权限。
- 使用归档管理器服务器时，无法实际除去文件空间。 尽管未实际删除文件空间，但是此函数调用将返回 *rc=0*。 验证文件空间是否已删除的唯一方法是向服务器发出文件空间查询。
- IBM Spectrum Protect 服务器删除文件空间功能是后台进程。如果发生了传递返回码之前检测到的错误以外的错误，会将这些错误记录在 IBM Spectrum Protect 服务器日志中。

语法

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

参数

dsUint32_t dsmHandle (I)
将此调用与之前的 **dsmInitEx** 调用相关联的句柄。

char *fsName (I)
指向要删除的文件空间名称的指针。 不允许使用通配符。

unsigned char repository (I)
指示要删除的文件空间是备份存储库和/或归档存储库。 此字段可能的值包括：

```
DSM_ARCHIVE_REP     /* archive repository   */
DSM_BACKUP_REP      /* backup repository    */
DSM_REPOS_ALL       /* all repository types */
```

返回码

返回码编号用括号 ( ) 括起。

表 *25. dsmDeleteFS* 的返回码

返回码	说明
DSM_RC_ABORT_NOT_AUTHORIZED (27)	您不具有删除文件空间所需的权限。
DSM_RC_INVALID_REPOS (2015)	存储库的值无效。
DSM_RC_FSNAME_NOTFOUND (2060)	找不到文件空间名称。
DSM_RC_NEED_ROOT (2300)	API 调用者必须为 root 用户。

dsmDeleteObj

dsmDeleteObj 函数调用会取消激活备份对象，删除备份对象或删除存储器中的归档对象。 **dtBackup** 类型将仅取消激活当前活动的备份副本。 **dtBackupID** 类型将从服务器除去指定的任何对象标识。 从事务中调用此函数。

请参阅 **dsmBeginTxn** 以获取更多信息。

限制: 您无法删除保留集中包含的备份对象。要满足长期数据保留时间需求，这些文件保留在服务器存储器中，并且会根据保留集自己的到期日期到期，到期后即可删除。

在发送 **dsmDeleteObj** 之前，请发送第 26 页的『查询 IBM Spectrum Protect 系统』中描述的查询序列，以获取 **delInfo** 的信息。 对 **dsmGetNextQObj** 的调用将返回名为 **qryRespBackupData**（针对备份查询）或 **qryRespArchiveData**（针对归档查询）的数据结构。 这些数据结构包括 **delInfo** 的所需信息。

maxObjPerTxn 的值确定可以在单个事务中删除的最大对象数。 要获取此值，请调用 **dsmQuerySessInfo**。

提示: 您的节点必须具有管理员设置的相应许可权。要删除归档对象，必须具有归档删除权限。 您不需要备份删除权限来取消激活备份对象。

语法

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

参数

dsUint32_t dsmHandle (I)
将此调用与之前的 **dsmInitEx** 调用相关联的句柄。

dsmDelType delType (I)
指示要删除的对象类型（备份或归档）。 可能的值包括:

Name	描述
dtArchive	先前已归档要删除的对象。
dtBackup	先前已备份要取消激活的对象。
dtBackupID	先前已备份要删除的对象。 **限制:** 将此 **delType** 与 *objID* 配合使用会从服务器除去备份对象。 仅对象的所有者才能将其删除。 您可以删除对象的任何版本（现行或非现行）。 服务器将协调这些版本。 如果删除现行版本的对象，那么第一个非现行版本将变为现行版本。 如果删除对象的非现行版本，那么所有更旧的版本都将前移。 必须注册具有 **backDel** 许可权的节点。

dsmDelInfo delInfo (I)
一种结构，其字段用于标识对象。 根据对象是备份对象还是归档对象，字段会有所不同。 取消激活备份对象的结构 delBack 包含对象名和对象副本组。 归档对象的结构 delArch 包含对象标识。

用于除去备份对象的结构 delBackID 包含对象标识。

返回码

返回码编号用括号 ( ) 括起。

表 *26. dsmDeleteObj* 的返回码

返回码	说明
DSM_RC_FS_NOT_REGISTERED (2061)	未注册文件空间名称。
DSM_RC_WRONG_VERSION_PARM (2065)	应用程序客户机 API 版本不同于 IBM Spectrum Protect 库版本。

dsmEndGetData

dsmEndGetData 函数调用将结束从存储器获取对象的 **dsmBeginGetData** 会话。

dsmEndGetData 函数调用将在希望恢复的所有对象得到处理后开始，并提前结束获取进程。 调用 **dsmEndGetData** 以结束 **dsmBeginGetData** 会话，才能继续进行其他处理。

根据 **dsmEndGetData** 的调用时间，API 可能需要完成部分数据流的处理，才能停止进程。 因此，调用者不应期望立即从此调用返回。 如果应用程序需要结束会话并立即结束恢复，请使用 **dsmTerminate**。

不存在特定于此调用的返回码。

语法

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

参数

dsUint32_t dsmHandle (I)
将此调用与之前的 **dsmInitEx** 调用相关联的句柄。

dsmEndGetDataEx

dsmEndGetDataEx 函数调用可提供已发送的不依赖 LAN 的总字节数。 这是 **dsmEndGetData** 函数调用的扩展功能。

语法

不存在特定于此调用的返回码。

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

参数

dsmEndGetDataExIn_t *dsmEndGetDataExInP (I)
传递 end get 对象 dsmHandle，该对象可识别会话并将其与后续调用相关联。

dsmEndGetDataExOut_t *dsmEndGetDataExOutP (O)
此结构包含此输入参数：

totalLFBytesRecv
已接收的不依赖 LAN 的总字节数。

dsmEndGetObj

dsmEndGetObj 函数调用可结束获取指定对象数据的 **dsmGetObj** 会话。

在收到有关对象的结束数据后，开始 **dsmEndGetObj** 调用。 这表明已接收了所有数据，或不再接收有关该对象的数据。 在可以开始另一个 **dsmGetObj** 调用前，您必须先调用 **dsmEndGetObj**。

根据调用 **dsmEndGetObj** 的时间，API 在此过程停止前，可能需要先处理部分数据流。 不要期望该调用会立即返回。

语法

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

参数

dsUint32_t dsmHandle (I)
将此调用与之前的 **dsmInitEx** 调用相关联的句柄。

返回码

返回码编号用括号 ( ) 括起。

表 *27. dsmEndGetObj* 的返回码

返回码	说明
DSM_RC_NO_MEMORY (102)	没有剩余的 RAM 来完成请求。

dsmEndQuery

dsmEndQuery 函数调用表明 **dsmBeginQuery** 操作结束。 应用程序客户机会发送 **dsmEndQuery** 以完成查询。 此调用既可在通过 **dsmGetNextQObj** 获得所有查询响应后发送，也可在返回所有数据前发送以结束查询。

提示: 这种情况下，IBM Spectrum Protect 继续将查询数据从服务器发送到客户机，但是，API 将丢弃任何剩余数据。

一旦发送 **dsmBeginQuery** 后，在开始任何其他活动前，必须先发送 **dsmEndQuery**。

不存在特定于此调用的返回码。

语法

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

参数

dsUint32_t dsmHandle (I)
将此调用与之前的 **dsmInitEx** 调用相关联的句柄。

dsmEndSendObj

dsmEndSendObj 函数调用指示向存储器发送数据结束。

输入 **dsmEndSendObj** 函数调用以指示 **dsmSendObj** 和 **dsmSendData** 调用的数据结束。 如果不执行此操作，将发生协议违例。

语法

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

参数

dsUint32_t dsmHandle (I)
将此调用与之前的 **dsmInitEx** 调用相关联的句柄。

返回码

返回码编号用括号 ( ) 括起。

表 *28.* 为 *dsmEndSendObj* 返回代码

返回码	说明
DSM_RC_NO_MEMORY (102)	没有完成此请求所需的剩余 RAM。

dsmEndSendObjEx

dsmEndSendObjEx 函数调用提供有关已处理的字节数的附加信息。 此信息包括：已发送的总字节数、压缩信息、不依赖 LAN 的字节数和重复数据删除信息。

dsmEndSendObjEx 函数调用是 **dsmEndSendObj** 函数调用的扩展。

语法

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

参数

dsmEndSendObjExIn_t *dsmEndSendObjExInP (I)
此参数传递结束发送对象 dsmHandle，该对象可识别会话并将其与后续调用相关联。

dsmEndSendObjExOut_t *dsmEndSendObjExOutP (O)
此参数传递结束发送对象的信息：

Name	描述
totalBytesSent	从该应用程序读取的总字节数。
objCompressed	显示是否压缩对象的标志。
totalCompressedSize	压缩后的总字节大小。
totalLFBytesSent	已发送的不依赖 LAN 的总字节数。

Name	描述
objDeduplicated	显示是否由 API 删除重复数据的对象的标志。
totalDedupSize	删除重复数据后发送的总字节数。

返回码

返回码编号用括号 ( ) 括起。

表 *29. dsmEndSendObjEx* 的返回码

返回码	说明
DSM_RC_NO_MEMORY (102)	没有完成此请求所需的剩余 RAM。

dsmEndTxn

dsmEndTxn 函数调用结束 IBM Spectrum Protect 事务。

将 **dsmEndTxn** 函数调用与 **dsmBeginTxn** 配对以识别作为事务的一个或一组调用。

应用程序客户机可以在 **dsmEndTxn** 调用上指定必须落实还是结束事务。

在事务的界限内执行以下所有调用：

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

语法

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

参数

dsUint32_t dsmHandle (I)
将此调用与之前的 **dsmInitEx** 调用关联的句柄。

dsUint8_t vote (I)
指示应用程序客户机是否落实在先前的 **dsmBeginTxn** 调用和此次调用之间执行的所有操作。 以下是可能的值：

```
DSM_VOTE_COMMIT /* 提交当前事物 */
DSM_VOTE_ABORT /* 回滚当前事物 */
```

仅当应用程序找到停止事务的原因时才使用 DSM_VOTE_ABORT。

dsUint16_t *reason (O)
如果对 **dsmEndTxn** 的调用因出错而结束，或者 vote 的值未达成一致，那么此参数具有指示表决失败原因的原因码。 用于此调用的返回码可能为零，原因码可能不为零。 因此，应用程序客户机在假设成功完成前，必须始终检查返回码和原因码是否有错误 (if (rc || reason))。

如果应用程序指定 vote 为 DSM_VOTE_ABORT，那么原因码为 DSM_RS_ABORT_BY_CLIENT (3)。 关于可能的原因码列表，请参阅第 123 页的『附录 A API 返回码源文件：dsmrc.h』。 返回码列表中的数

字 1 到 50 将保留为原因码。 如果服务器结束此事务，那么返回码为 DSM_RC_CHECK_REASON_CODE。 在此情况下，原因值包含有关终止原因的更多信息。

返回码

返回码编号用括号 ( ) 括起。

表 *30.* 为 *dsmEndTxn* 返回代码

返回码	说明
DSM_RC_ABORT_CRC_FAILED (236)	从服务器接收的 CRC 与由客户机计算出的 CRC 不匹配。
DSM_RC_INVALID_VOTE (2011)	为 vote 值指定的值无效。
DSM_RC_CHECK_REASON_CODE (2302)	事务终止。 检查原因字段。
DSM_RC_ABORT_STGPOOL_COPY_CONT_NO (241)	写入某个副本存储池失败，IBM Spectrum Protect 存储池选项 COPYCONTINUE 设置为 NO。事务将终止。
DSM_RC_ABORT_RETRY_SINGLE_TXN (242)	此终止代码指示当前事务由于在存储操作期间出现问题而终止。 可通过在单个事务中发送每个文件来解决此问题。 此错误在以下情况下很典型: · 下一个存储池具有不同的副本存储池列表。 · 操作在事务中间切换到该池。 .

dsmEndTxnEx

dsmEndTxnEx 函数调用提供组引导符对象标识信息以供您用于 **dsmGroupHandler** 函数调用。 这是 **dsmEndTxn** 函数调用的扩展。

语法

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

参数

dsmEndTxnExIn_t *dsmEndTxnExInP (I)
此结构包含以下参数:

dsmHandle
识别会话并将其与后续 IBM Spectrum Protect 调用相关联的句柄。

dsUint8_t vote (I)
指示应用程序客户机是否落实在之前的 **dsmBeginTxn** 调用和此次调用之间执行的所有操作。 可能的值有:

```
DSM_VOTE_COMMIT /* 提交当前事物 */
DSM_VOTE_ABORT /* 回滚当前事物 */
```

仅当应用程序找到停止事务的原因时才使用 DSM_VOTE_ABORT。

dsmEndTxnExOut_t *dsmEndTxnExOutP (O)
此结构包含以下参数:

dsUint16_t *reason (O)
如果 **dsmEndTxnEx** 调用由于发生错误而结束，或不同意 表决 的值，那么此参数具有指示此表决失败的原因码。

提示: 用于此调用的返回码可能为零，原因码可能不为零。 因此，应用程序客户机在假设成功完成前，必须始终检查返回码和原因码是否有错误 (`if (rc || reason)`)。

如果应用程序指定表决为 DSM_VOTE_ABORT，那么原因码为 DSM_RS_ABORT_BY_CLIENT (3)。 关于可能的原因码列表，请参阅第 123 页的『附录 A API 返回码源文件：dsmrc.h』。 返回码列表中的数字 1 到 50 将保留为原因码。 如果服务器结束此事务，那么，返回码为 DSM_RC_CHECK_REASON_CODE。 在此情况下，原因值包含有关终止原因的更多信息。

groupLeaderObjId
当 DSM_ACTION_OPEN 标志与 **dsmGroupHandler** 调用一起使用时返回的组引导符对象标识。

返回码

返回码编号用括号 ( ) 括起。

表 *31. dsmEndTxnEx* 的返回码

返回码	说明
DSM_RC_INVALID_VOTE (2011)	为表决指定的值无效。
DSM_RC_CHECK_REASON_CODE (2302)	事务终止。 检查原因字段。
DSM_RC_ABORT_STGPOOL_COPY_CONT_NO (241)	写入其中一个副本存储池失败，IBM Spectrum Protect 存储池选项 COPYCONTINUE 设为 NO。事务终止。
DSM_RC_ABORT_RETRY_SINGLE_TXN (242)	同步写入操作期间，事务中的对象将转至带有不同副本存储池的目标。 结束当前事务，并在其自己的事务中再次发送每个对象。

dsmGetData

dsmGetData 函数调用从 IBM Spectrum Protect 获取数据字节流，并将其放在调用者缓冲区中。 要从之前的 **dsmGetObj** 或 **dsmGetData** 接收更多数据时，应用程序客户机将调用 **dsmGetData**。

语法

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

参数

dsUint32_t dsmHandle (I)
将此调用与之前的 **dsmInitEx** 调用相关联的句柄。

DataBlk *dataBlkPtr (I/O)
指向一个结构，该结构包括指向已接收数据的缓冲区的指针和缓冲区大小。 返回时，此结构包含实际转移的字节数。 关于类型定义的信息，请参阅第 133 页的『附录 B API 类型定义源文件』。

返回码

返回码编号用括号 ( ) 括起。

表 32. 为 *dsmGetData* 返回代码

返回码	说明
DSM_RC_ABORT_INVALID_OFFSET (33)	部分对象检索期间指定的偏移量大于对象的长度。
DSM_RC_ABORT_INVALID_LENGTH (34)	部分对象检索期间指定的长度大于对象长度，或除长度以外的偏移量超出对象结尾。
DSM_RC_FINISHED (121)	已完成处理。 已收到最后一个缓冲区。 检查数据量的 numBytes，然后，调用 IBM Spectrum ProtectdsmEndGetObj。
DSM_RC_NULL_DATABLKPTR (2001)	数据块指针为空。
DSM_RC_ZERO_BUFLEN (2008)	针对数据块指针的缓冲区长度为零。
DSM_RC_NULL_BUFPTR (2009)	针对数据块指针的缓冲区指针为空。
DSM_RC_WRONG_VERSION_PARM (2065)	应用程序客户机的 API 版本与 IBM Spectrum Protect 库的版本不同。
DSM_RC_MORE_DATA (2200)	还可获取更多数据。

dsmGetBufferData

dsmGetBufferData 函数调用通过缓冲区接收来自于 IBM Spectrum Protect 的字节流。每次调用后，该应用程序需要复制此数据，并通过调用 **dsmReleaseBuffer** 发布缓冲区。 如果该应用程序所占的缓冲区数等于 **dsmInitEx** 调用中指定的 numTsmBufferscall，那么，在调用 **dsmReleaseBuffer** 前，**dsmGetBufferData** 函数将被阻塞。

语法

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

参数

getDataExIn_t * dsmGetBufferDataExInP (I)
此结构包含以下输入参数。

dsUint32_t dsmHandle
识别会话并将其与之前的 **dsmInitEx** 调用相关联的句柄。

getDataExOut_t * dsmGetBufferDataExOutP (0)
此结构包含以下输出参数。

dsUint8_t tsmBufferHandle(0)
识别已接收的缓冲区的句柄。

char *dataPtr(0)
已写入数据的地址。

dsUint32_t numBytes(0)
IBM Spectrum Protect 所写的实际字节数。

返回码

返回码编号用括号 ( ) 括起。

表 33. *dsmGetBufferData* 的返回码

返回码	说明
DSM_RC_BAD_CALL_SEQUENCE (2041)	此调用没有在适当的状态下发出。
DSM_RC_OBJ_ENCRYPTED (2049)	此函数无法用于加密对象。
DSM_RC_OBJ_COMPRESSED (2048)	此函数无法用于压缩对象。
DSM_RC_BUFF_ARRAY_ERROR (2045)	发生缓冲区数组错误。

dsmGetNextQObj

dsmGetNextQObj 函数调用从先前的 **dsmBeginQuery** 调用中获取下一个查询响应，并将响应放置在调用者缓冲区中。

dsmGetNextQObj 调用既可调用一次，也可调用多次。 每次调用函数时，都将检索到单个查询记录，否则将返回错误或 DSM_RC_FINISHED 原因码。 如果返回了 DSM_RC_FINISHED，那么无需处理更多数据。 如果已检索到所有查询数据或者无需更多查询数据，请发送 **dsmEndQuery** 调用以结束查询过程。

dataBlkPtr 参数必须指向使用 **qryResp*Data** 结构类型定义的缓冲区。 调用 **dsmGetNextQObj** 所在的上下文确定查询响应中输入的结构类型。

语法

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

参数

dsUint32_t dsmHandle (I)
将此调用与之前的 **dsmInitEx** 调用相关联的句柄。

DataBlk *dataBlkPtr (I/O)
指向包括用于要接收的数据的缓冲区的指针和缓冲区大小的结构。 该缓冲区是 **qryResp*Data** 响应结构。 返回时，此结构包含转移的字节数。 下表中描述了与每个查询类型相关联的结构。 有关 **DataBlk** 的类型定义的更多信息，请参阅以下主题：第 133 页的『附录 B API 类型定义源文件』。

表 34. *DataBlk* 指针结构

查询	响应结构	特别注意的字段
qtArchive	**qryRespArchiveData**	**sizeEstimate** 包含在先前的 **dsmSendObj** 调用上传递的值。 **mediaClass** 如果对象位于磁盘上，那么值可以为 MEDIA_FIXED，或者如果对象位于磁带上，那么值可以为 MEDIA_LIBRARY。 **clientDeduplicated** 指示客户机是否对此对象进行重复数删除。

表 *34. DataBlk* 指针结构 *(续)*

查询	响应结构	特别注意的字段
qtBackup	**qryRespBackupData**	**restoreOrderExt** 类型为 dsUint16_t。在 **dsmBeginGetData** 调用上复原多个对象时，对此字段进行排序。此调用的排序代码示例位于 API 样本 dapiqry.c 中。要获取排序示例，请参阅以下主题：第 54 页的图 16。 **sizeEstimate** 包含在先前的 **dsmSendObj** 调用上传递的值。 **mediaClass** 如果对象位于磁盘上，那么值可以为 MEDIA_FIXED，或者如果对象位于磁带上，那么值可以为 MEDIA_LIBRARY。 **clientDeduplicated** 指示客户机是否对此对象进行重复数删除。
qtBackupActive	**qryARespBackupData**	
qtBackupGroups	**qryRespBackupData**	**dsBool_t isGroupLeader** 如果为 true，那么表明此对象是组引导符。
qtOpenGroups	**qryRespBackupData**	**dsBool_t isOpenGroup;** 如果为 true，那么表明此组已打开且未完成。
qtFilespace	**qryRespFSData**	**backStartDate** 包含使用 **backStartDate** 操作更新文件空间时的服务器时间戳记。 **backCompleteDate** 包含使用 **backCompleteDate** 操作更新文件空间时的服务器时间戳记。 **lastReplStartDate** 包含上次在服务器上启动复制的时间戳记。 **lastReplCmpltDate** 包含上次完成复制的时间戳记，即使存在故障。 **lastBackOpDateFromServer** 包含上次在服务器上进行保存的存储时间戳记。 **lastBackOpDateFromLocal** 包含上次在客户机上进行保存的存储时间戳记。
qtMC	**qryRespMCData** **qryRespMCDetailData**	

表 34. *DataBlk* 指针结构 (续)

查询	响应结构	特别注意的字段
qtProxyNodeAuth	**qryRespProxyNodeData** **targetNodeName** **peerNodeName** **hlAddress** **llAddress**	
qtProxyNodePeer	**qryRespProaxyNodeData** **targetNodeName** **peerNodeName** **hlAddress** **llAddress**	

返回码

下表描述了 **dsmGetNextQObj** 函数调用的返回码。

表 35. ***dsmGetNextQObj*** 函数调用的返回码

返回码	返回码编号	描述
DSM_RC_ABORT_NO_MATCH	2	没有与请求的查询相匹配的结果。
DSM_RC_FINISHED	121	已完成处理（开始 **dsmEndQuery**）。 无需处理其他数据。
DSM_RC_UNKNOWN_FORMAT	122	IBM Spectrum Protect 试图恢复或检索的文件具有未知格式。
DSM_RC_COMM_PROTOCOL_ERROR	136	通信协议错误。
DSM_RC_NULL_DATABLKPTR	2001	指针没有指向数据块。
DSM_RC_INVALID_MCNAME	2025	无效的管理类名称。
DSM_RC_BAD_CALL_SEQUENCE	2041	调用顺序无效。
DSM_RC_WRONG_VERSION_PARM	2065	应用程序客户机 API 的版本不同于 IBM Spectrum Protect 库版本。
DSM_RC_MORE_DATA	2200	还可获取更多数据。
DSM_RC_BUFF_TOO_SMALL	2210	缓冲区过小。

dsmGetObj

dsmGetObj 函数调用从 IBM Spectrum Protect 数据流获取所需的对象数据，并将其放在调用者缓冲区中。 **dsmGetObj** 调用使用对象标识从数据流中获取下一个对象或部分对象。

指定对象的数据放在指向 **DataBlk** 的缓冲区中。 如果要使用更多数据，您必须对 **dsmGetData** 进行一个或多个调用以便在返回 DSM_RC_FINISHED 返回码前接收剩余的对象数据。 在 **DataBlk** 中检查 numBytes 字段以查看缓冲区中是否有任何剩余数据。

对象的排列要求按照 **dsmBeginGetData** 调用中 **dsmGetList** 参数列出的顺序。 当应用程序客户机需要在数据流中传递对象以便稍后在列表中获取对象时会出现例外。 如果由对象标识指定的对象不是该数据流中的

下一个对象，那么，在找到该对象或完成该数据流前会一直处理该数据流。 使用此功能时请务必谨慎，因为它可能需要处理和丢弃大量数据来查找请求的对象。

要求: 如果 **dsmGetObj** 返回故障代码（NOT FINISHED 或 MORE_DATA），那么必须终止此会话以停止复原操作。在使用加密和接收 RC_ENC_WRONG_KEY 时尤为重要。必须使用适当的密钥启动新会话。

语法

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

参数

dsUint32_t dsmHandle (I)
将此调用与之前的 **dsmInitEx** 调用相关联的句柄。

ObjID *objIdP (I)
指向要恢复的对象标识的指针。

DataBlk *dataBlkPtr (I/O)
指向放置已恢复数据的缓冲区的指针。

返回码

返回码编号用括号 ( ) 括起。

表 *36.* 为 *dsmGetObj* 返回代码

返回码	说明
DSM_RC_ABORT_INVALID_OFFSET (33)	部分对象检索期间指定的偏移量大于对象长度。
DSM_RC_ABORT_INVALID_LENGTH (34)	部分对象检索期间指定的长度大于对象长度，或除长度以外的偏移量超出对象结尾。
DSM_RC_FINISHED (121)	已完成处理（开始 **dsmEndGetObj**）。
DSM_RC_WRONG_VERSION_PARM (2065)	应用程序客户机的 API 版本与 IBM Spectrum Protect 库的版本不同。
DSM_RC_MORE_DATA (2200)	还可获取更多数据。
RC_ENC_WRONG_KEY (4580)	**dsmInitEx** 调用中提供的密钥或已保存的密钥与用于加密此对象的密钥不匹配。 终止此会话，并提供正确的密钥。

dsmGroupHandler

dsmGroupHandler 函数调用可根据指定的输入在逻辑文件组上执行操作。 客户机将许多单个对象关联在一起以便在 IBM Spectrum Protect 服务器上作为逻辑组引用和管理。

有关更多信息，请参阅第 50 页的『文件分组』。

语法

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

参数

dsmGroupHandlerIn_t *dsmGroupHandlerInP (I)
将组属性传递到 API。

groupType
组类型。 值包括：

· DSM_GROUPTYPE_PEER - peer group

actionType
要执行的操作。 值包括：

· DSM_GROUP_ACTION_OPEN - 创建新组
· DSM_GROUP_ACTION_CLOSE - 提交和保存打开的组
· DSM_GROUP_ACTION_ADD - 添加到组
· DSM_GROUP_ACTION_ASSIGNTO - 分配到其他组
· DSM_GROUP_ACTION_REMOVE- 从组中除去成员

memberType.
对象的组类型。 值包括：

· DSM_MEMBERTYPE_LEADER - 组引导符
· DSM_MEMBERTYPE_MEMBER - 组成员

***uniqueGroupTagP**
与组相关联的唯一的字符串标识。

leaderObjId
组引导符的对象标识。

***objNameP**
指向组引导符的对象名称。

memberObjList
要除去或分配的对象列表。

dsmGroupHandlerOut_t *dsmGroupHandlerOutP (O)
传递 API 已完成的结构的地址。 结构版本已返回。

返回码

返回码编号用括号 ( ) 括起。

表 *37.* 为 *dsmGroupHandler* 返回代码

返回码	说明
DSM_RC_ABORT_INVALID_GROUP_ACTION (237)	试图在组引导符或成员上进行无效操作。

dsmInit

dsmInit 函数调用开始 API 会话，并将客户机连接至 IBM Spectrum Protect 存储库。 此应用程序客户机一次只能进行一个活动会话。 要使用其他参数打开另一个会话，请先使用 **dsmTerminate** 调用来结束当前会话。

要允许交叉结点查询并恢复和检索，请使用 *-fromnode* 和 *-fromowner* 字符串选项。 请参阅第 20 页的『跨节点和所有者访问对象』以获取更多信息。

语法

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

参数

dsUint32_t *dsmHandle (O)
识别此初始化会话将其与后续 IBM Spectrum Protect 调用相关联的句柄。

dsmApiVersion *dsmApiVersionP (I)
识别 API 版本的数据结构的指针，该 API 版本就是应用程序客户机正在用于此会话的版本。 此结构包含三个在 dsmapitd.h 中设置的常量：DSM_API_VERSION、DSM_API_RELEASE 和 DSM_API_LEVEL。 文件。 必须执行之前对 **dsmQueryApiVersion** 的调用以确保应用程序客户机 API 版本和安装在用户工作站中的 API 库的版本间存在兼容性。

char *clientNodeNameP (I)
此参数指向 IBM Spectrum Protect 会话的节点。 所有会话必须具有与它们相关联的节点名。 dsmapitd.h 文件中的常量 DSM_MAX_NODE_LENGTH 设置节点名所允许的最大大小。

节点名不区分大小写。

如果此参数都设置为 NULL，且 *passwordaccess* 设置为 *prompt*，那么，API 将试图首先从已传递的选项字符串中获取节点名。 如果节点名没有位于其中，那么，API 将试图从配置文件或选项文件中获取节点名。 如果这些试图找到节点名的操作失败，那么，UNIX 或 Linux API 将使用系统主机名，而其他操作系统上的 API 将返回 DSM_RC_REJECT_ID_UNKNOWN 代码。

如果此参数为 NULL，并且 dsm.opt 文件（对于 Windows API）或 dsm.sys 文件（对于 UNIX 或 Linux API）中的 passwordaccess 选项设置为 generate，那么 API 使用 nodename 选项值或系统主机名。

char *clientOwnerNameP (I)
此参数指向 IBM Spectrum Protect 会话的所有者。 如果开始的会话所在的操作系统是多用户操作系统，那么，名称为 NULL 的所有者（root 用户）具有备份、归档、恢复或检索任何属于此应用程序的对象的权限，而无需考虑对象的所有者。

所有者名称不区分大小写。

如果此参数为 NULL 并且 dsm.sys 文件中的 passwordaccess 选项设置为 generate，那么 API 使用登录用户标识。

注: 在多用户操作系统上，所有者名称没有必要与运行此应用程序的会话的活动用户标识相匹配。

char *clientPasswordP (I)
此参数指向运行 IBM Spectrum Protect 会话的节点的密码。 dsmapitd.h 文件中的常量 DSM_MAX_VERIFIER_LENGTH 设置密码所允许的最大大小。

密码是不区分大小写的。

除非先启动密码文件，否则，当 *passwordaccess* 设置为 *generate* 时，将忽略此参数的值。

char *applicationType (I)
此参数识别正在运行此会话的应用程序。 应用程序客户机定义该值。

每次 API 应用程序客户机与服务器开始会话时，都会在该服务器上更新此客户机的应用程序类型（或平台）。 我们建议该应用程序类型值应包含操作系统的缩写，因为该值将输入在服务器的 **platform** 字段中。 字符串长度最大为 DSM_MAX_PLATFORM_LENGTH。

要查看应用程序类型的当前值，请调用 **dsmQuerySessInfo**。

char *configfile (I)
此参数指向包含 API 配置文件的标准名称的字符串。 API 配置文件中指定的选项会覆盖它们在客户机选项文件中的规格。 在安装 IBM Spectrum Protect（客户机或 API）时定义选项文件。

char *options (I)
指向可包含用户选项的字符串，如：

- *Compressalways*
- *Servername*（仅适用于 UNIX 或 Linux）
- *TCPServeraddr*
- *Fromnode*
- *Fromowner*
- *EnableClientEncryptKey*
- *Passwordaccess*

应用程序客户机可以使用选项列表来覆盖配置文件设置的这些选项值。

选项的格式为：

1. 选项列表中指定的每个选项都以连字符 (-) 开头，后面是选项的关键字。
2. 然后，关键字后是等号 (=)，再后面是选项参数。
3. 如果选项参数包含空格，那么，请用单引号或双引号将参数括起。
4. 如果指定了多个选项，请使用空格分隔。

如果选项为 NULL，那么，所有选项的值都将从用户选项文件或 API 配置文件中获取。

返回码

返回码编号用括号 ( ) 括起。

表 *38. dsmInit* 的返回码

返回码	说明
DSM_RC_ABORT_SYSTEM_ERROR (1)	服务器已检测到系统错误，且已通知客户机。
DSM_RC_REJECT_VERIFIER_EXPIRED (52)	密码已过期，必须更新。
DSM_RC_REJECT_ID_UNKNOWN (53)	无法找到节点名。
DSM_RC_AUTH_FAILURE (137)	认证失败。
DSM_RC_NO_STARTING_DELIMITER (148)	模式中没有开始定界符。
DSM_RC_NEEDED_DIR_DELIMITER (149)	“匹配目录” “”元字符串 (“...”) 前后都需要紧跟目录定界符，但是找不到目录定界符。
DSM_RC_NO_PASS_FILE (168)	密码文件不可用。
DSM_RC_UNMATCHED_QUOTE (177)	选项字符串中有一个不匹配的引号。
DSM_RC_NLS_CANT_OPEN_TXT (0610)	无法打开消息文本文件。
DSM_RC_INVALID_OPT (400)	选项字符串中的条目无效。
DSM_RC_INVALID_DS_HANDLE (2014)	无效的 DSM 句柄。

表 38. *dsmInit* 的返回码 (续)

返回码	说明
DSM_RC_NO_OWNER_REQD (2032)	当 *passwordaccess* 设置为 *generate* 时，所有者参数必须为 NULL。
DSM_RC_NO_NODE_REQD (2033)	当 *passwordaccess* 设置为 *generate* 时，节点参数必须为 NULL。
DSM_RC_WRONG_VERSION (2064)	此应用程序客户机的 API 版本高于 IBM Spectrum Protect 版本。
DSM_RC_PASSWD_TOOLONG (2103)	指定的密码过长。
DSM_RC_NO_OPT_FILE (2220)	无法找到配置文件。
DSM_RC_INVALID_KEYWORD (2221)	选项字符串中指定的关键字无效。
DSM_RC_PATTERN_TOO_COMPLEX (2222)	包含/排除模式对于 IBM Spectrum Protect 而言太过复杂以至于无法解释。
DSM_RC_NO_CLOSING_BRACKET (2223)	模式中没有右方括号。
DSM_RC_INVALID_SERVER (2225)	对于多用户环境，系统配置文件中没有找到服务器。
DSM_RC_NO_HOST_ADDR (2226)	没有足够的信息来连接至主机。
DSM_RC_MACHINE_SAME (2227)	选项文件中定义的节点名不能与系统主机名相同。
DSM_RC_NO_API_CONFIGFILE (2228)	无法打开配置文件。
DSM_RC_NO_INCLEXCL_FILE (2229)	没有找到包含/排除文件。
DSM_RC_NO_SYS_OR_INCLEXCL (2230)	找不到 dsm.sys 文件或包含/排除文件。

相关概念
客户机选项文件概述
处理选项

dsmInitEx

dsmInitEx 函数调用使用其他参数启动 API 会话以进行扩展验证。

语法

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

参数

dsUint32_t *dsmHandleP (O)
识别此初始化会话将其与后续 IBM Spectrum Protect 调用相关联的句柄。

dsmInitExIn_t *dsmInitExInP
此结构包含以下输入参数:

dsmApiVersion *dsmApiVersionP (I)
此参数是指向数据结构的指针，该数据结构识别应用程序客户机对此会话使用的 API 版本。 此结构包含在 dsmapitd.h 文件 () 中设置的四个常量的值：DSM_API_VERSION、DSM_API_RELEASE、DSM_API_LEVEL 和 DSM_API_SUBLEVEL。调用 **dsmQueryApiVersionEx** 并验证应用程序客户机的 API 版本与用户工作站上安装的 API 库版本兼容。

char *clientNodeNameP (I)
此参数指向 IBM Spectrum Protect 会话的节点。 所有会话必须与一个节点名相关联。dsmapitd.h 文件中的 DSM_MAX_NODE_LENGTH 常量设置节点名的最大大小。

节点名不区分大小写。

如果此参数设置为 NULL，且 **passwordaccess** 设置为 prompt，那么 API 将试图先从已传递的选项字符串中获取节点名。 如果节点名没有位于其中，那么，API 将试图从配置文件或选项文件中获取节点名。 如果查找节点名的这些尝试失败，那么 UNIX 或 Linux API 使用系统主机名，而其他操作系统的 API 将返回 DSM_RC_REJECT_ID_UNKNOWN。

如果此参数为 NULL，并且 dsm.opt 文件（对于 Windows API）或 dsm.sys 文件（对于 UNIX 或 Linux API）中的 **passwordaccess** 选项设置为 generate，那么 API 使用 nodename 选项值或系统主机名。

char *clientOwnerNameP (I)
此参数指向 IBM Spectrum Protect 会话的所有者。 如果操作系统是多用户平台，那么 NULL（root 用户）的所有者名称有权备份、归档、复原或检索属于应用程序的任何对象，而无论对象的所有者。

所有者名称不区分大小写。

如果此参数为 NULL 并且 dsm.sys 文件中的 **passwordaccess** 选项设置为 generate，那么 API 使用登录用户标识。

注: 在多用户平台上，所有者名称没有必要与运行此应用程序的会话的活动用户标识相匹配。

char *clientPasswordP (I)
指向运行 IBM Spectrum Protect 会话的节点的密码。 dsmapitd.h 文件中的 DSM_MAX_VERIFIER_LENGTH 常量设置密码所允许的最大大小。

密码是不区分大小写的。

除非先启动密码文件，否则，当 **passwordaccess** 设置为 generate 时，将忽略此参数的值。

char *userNameP;
指向对此节点具有客户机权限的管理用户名称。

char *userPasswordP;
指向 **userName** 参数的密码的指针（如果提供值）。

char *applicationType (I)
识别正在运行 IBM Spectrum Protect 会话的应用程序。 应用程序客户机识别该值。

每次 API 应用程序客户机与服务器开始会话时，都会在该服务器上更新此客户机的应用程序类型（或操作系统）。 在服务器上的 **platform** 字段中输入此值。请考虑在值中使用操作系统标识。在 DSM_MAX_PLATFORM_LENGTH 常量中定义最大字符串长度。

要查看应用程序类型的当前值，请调用 **dsmQuerySessInfo**。

char *configfile (I)
指向包含 API 配置文件的标准名称的字符串。API 配置文件中指定的选项会覆盖它们在客户机选项文件中的规范。在安装 IBM Spectrum Protect（客户机或 API）时定义选项文件。

char *options (I)
指向可包含用户选项的字符串，如:

- Compressalways
- Servername（仅限 UNIX 和 Linux 系统）
- TCPServeraddr（不适用于 UNIX 系统）
- Fromnode
- Fromowner
- Passwordaccess

应用程序客户机可以使用选项列表来覆盖配置文件设置的这些选项值。

选项具有以下格式：

1. 选项列表中指定的每个选项都以连字符 (-) 开头，后面是选项的关键字。
2. 关键字后是等号 (=)，然后是选项参数。
3. 如果选项参数包含空格，那么请用单引号或双引号将参数括起来。
4. 如果指定了多个选项，请使用空格分隔。

如果选项为 NULL，那么将从用户选项文件或 API 配置文件获取所有选项的值。

dirDelimiter
目录定界符，放在文件空格、高级别或低级别名称前。 仅当应用程序覆盖系统缺省值时，必须指定 **dirDelimiter** 参数。在 UNIX 或 Linux 环境中，缺省值为正斜杠 (/)。在 Windows 环境中，缺省值为反斜杠 (\)。

useUnicode
布尔标志，指示是否启用 Unicode。**useUnicode** 标志必须为 `false` 以在 UNIX 和 Windows 系统之间实现跨平台互操作性。

bCrossPlatform
必须设置的布尔标志 (`bTrue`)，用于在 UNIX 和 Windows 系统之间实现跨平台互操作性。在设置 `bCrossPlatform` 标志时，API 确保文件空间非 Unicode 并且应用程序不使用 Unicode。使用 Unicode 的 Windows 应用程序不兼容使用非 Unicode 编码的应用程序。不得针对使用 Unicode 的 Windows 应用程序设置 `bCrossPlatform` 标志。

UseTsmBuffers
指示是否使用缓冲器副本消除。

numTsmBuffers
`useTsmBuffers=bTrue` 时的缓冲区数量。

bEncryptKeyEnabled
指示是否使用应用程序管理的密钥加密。

encryptionPasswordP
加密密码。

限制: 在 `encryptkey=save` 时，如果存在加密密钥，那么将忽略在 **encryptionPasswordP** 中指定的值。

dsmAppVersion *appVersionP (I)
此参数是指向数据结构的指针，该数据结构标识启动 API 会话的应用程序的版本信息。此结构包含在 `tsmapitd.h` 文件中设置的四个常量的值：applicationVersion、applicationRelease、applicationLevel 和 applicationSubLevel。

dsmInitExOut_t *dsmInitExOut P
此结构包含输出参数。

dsUint32_t *dsmHandle (0)
识别此初始化会话并将其与后续 API 调用相关联的句柄。

infoRC
关于返回码的附加信息。 检查函数返回码和 **infoRC** 的值。**infoRC** 值 DSM_RC_REJECT_LASTSESS_CANCELED (69)，IBM Spectrum Protect 指示管理员已取消上次的会话。

返回码

返回码编号用括号 ( ) 括起。

表 *39.* 为 ***dsmInitEx*** 返回代码

返回码	说明
DSM_RC_ABORT_SYSTEM_ERROR (1)	IBM Spectrum Protect 服务器检测到系统错误并且已通知客户机。
DSM_RC_REJECT_VERIFIER_EXPIRED (52)	密码已到期并且必须进行更新。下一个调用必须为带有此调用返回的句柄的 **dsmChangePW**。
DSM_RC_REJECT_ID_UNKNOWN (53)	无法找到节点名。
DSM_RC_TA_COMM_DOWN (103)	通信链路已关闭。
DSM_RC_AUTH_FAILURE (137)	认证失败。
DSM_RC_NO_STARTING_DELIMITER (148)	模式中没有开始定界符。
DSM_RC_NEEDED_DIR_DELIMITER (149)	“匹配目录” “” 元字符串 (“...”) 前后都需要紧跟目录定界符，但是找不到。
DSM_RC_NO_PASS_FILE (168)	密码文件不可用。
DSM_RC_UNMATCHED_QUOTE (177)	选项字符串中有一个不匹配的引号。
DSM_RC_NLS_CANT_OPEN_TXT (0610)	无法打开消息文本文件。
DSM_RC_INVALID_OPT (2013)	选项字符串中的条目无效。
DSM_RC_INVALID_DS_HANDLE (2014)	无效的 DSM 句柄。
DSM_RC_NO_OWNER_REQD (2032)	当 **passwordaccess** 设置为 `generate` 时，所有者参数必须为 NULL。
DSM_RC_NO_NODE_REQD (2033)	在 **passwordaccess** 设置为 generate 时，节点参数必须为 NULL。
DSM_RC_WRONG_VERSION (2064)	此应用程序客户机的 API 版本高于 IBM Spectrum Protect 版本。
DSM_RC_PASSWD_TOOLONG (2103)	指定的密码过长。
DSM_RC_NO_OPT_FILE (2220)	找不到配置文件。
DSM_RC_INVALID_KEYWORD (2221)	选项字符串中指定的关键字无效。
DSM_RC_PATTERN_TOO_COMPLEX (2222)	包含/排除模式过于复杂以至于 IBM Spectrum Protect 无法进行解释。
DSM_RC_NO_CLOSING_BRACKET (2223)	模式中没有右方括号。
DSM_RC_INVALID_SERVER (2225)	对于多用户环境，系统配置文件中没有找到服务器。
DSM_RC_NO_HOST_ADDR (2226)	没有足够的信息来连接至主机。
DSM_RC_MACHINE_SAME (2227)	选项文件中定义的节点名不能与系统主机名相同。
DSM_RC_NO_API_CONFIGFILE (2228)	无法打开配置文件。
DSM_RC_NO_INCLEXCL_FILE (2229)	没有找到包含/排除文件。
DSM_RC_NO_SYS_OR_INCLEXCL (2230)	找不到 `dsm.sys` 或包含/排除文件。

相关概念

客户机选项文件概述

处理选项

dsmLogEvent

dsmLogEvent 函数调用可将用户消息 (ANE4991 I) 记录到服务器日志文件和/或本地错误日志中。 **logInfo** 的类型结构在此调用中传递。 此调用必须在在会话的 **InSession** 状态时执行。 不要在进行发送、获取或查询操作时执行。 要检索服务器上记录的消息，可通过管理客户机使用 **query actlog** 命令。

请参阅摘要状态图：第 61 页的图 20。

语法

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

参数

dsUint32_t dsmHandle(I)
将此调用与之前的 **dsmInitEx** 调用相关联的句柄。

logInfo *logInfoP (I)
传递消息和目标。 应用程序客户机负责分配结构的存储。

logInfo 结构中的字段为：

消息
要记录的消息文本。 这必须是以 null 结尾的字符串。 最大长度为 DSM_MAX_RC_MSG_LENGTH。

dsmLogtype
指定要记录消息的位置。 可能的值包括：**logServer**、**logLocal**、**logBoth**。

返回码

返回码编号用括号 ( ) 括起。

表 *40. dsmLogEvent* 的返回码

返回码	说明
DSM_RC_STRING_TOO_LONG (2120)	此消息字符串过长。

dsmLogEventEx

dsmLogEventEx 函数调用将用户消息记录到服务器日志文件和/或本地错误日志中。 此调用必须在处于会话中的 **InSession** 状态时进行。 此调用无法在 send、get 或 query 调用中进行。

摘要状态图: 要获取会话交互的概述，请参阅以下主题中的摘要状态图：

第 61 页的图 20

严重性决定 IBM Spectrum Protect 消息编号。 要查看服务器上记录的消息，请通过管理客户机使用 **query actlog** 命令。 如果应用程序生成许多写入客户机日志的客户机消息（dsmLogType 为 logLocal 或 logBoth），请使用 IBM Spectrum Protect 客户机选项 errorlogretention 来修剪客户机错误日志文件。 有关更多信息，请参阅 IBM Spectrum Protect 服务器文档。

语法

```
extern dsInt16_t DSMLINKAGE  dsmLogEventEx(
       dsUint32_t            dsmHandle,
       dsmLogExIn_t          *dsmLogExInP,
```

```
        dsmLogExOut_t         *dsmLogExOutP
);
```

参数

dsUint32_t dsmHandle(I)
将此调用与之前的 **dsmInitEx** 调用相关联的句柄。

dsmLogExIn_t *dsmLogExInP
此结构包含输入参数。

dsmLogSeverity severity;
此参数位于事件严重性中。 可能的值有：

```
   logSevInfo,       /* information ANE4990 */
   logSevWarning,    /* warning     ANE4991 */
   logSevError,      /* Error       ANE4992 */
   logSevSevere      /* severe      ANE4993 */
```

char appMsgID[8];
此参数是用于识别特定应用程序消息的字符串。 合适的格式为三个字符后跟四个数字，例如：DSM0250。

dsmLogType logType;
此参数指定指向此事件的位置。 该参数具有以下可能的值：

- logServer
- logLocal
- logBoth

char *message;
此参数是要记录的事件消息的文本。 该文本必须是以 null 结尾的字符串。 最大长度为 DSM_MAX_RC_MSG_LENGTH。

限制: 进入服务器的消息必须采用英语。 非英语消息将无法正确显示。

dsmLogExOut_t *dsmLogExOutP
此结构包含输出参数。 当前没有任何输出参数。

返回码

返回码编号用括号 ( ) 括起。

表 *41. dsmLogEventEx* 的返回码

返回码	说明
DSM_RC_STRING_TOO_LONG (2120)	此消息字符串过长。

dsmQueryAccess

dsmQueryAccess 函数调用查询服务器以获取对备份版本或对象的归档备份的所有访问权限规则。 将指向访问规则组的指针传递到调用中，并返回完成的组。 指向传递规则数的指针，从而指示该组中规则的数量。

不存在特定于此调用的返回码。

语法

```
dsInt16_t DSMLINKAGE dsmQueryAccess
                   (dsUint32_t            dsmHandle),
                    qryRespAccessData     **accessListP,
                    dsUint16_t            *numberOfRules) ;
```

参数

dsUint32_t dsmHandle (I)
将此调用与之前的 **dsmInitEx** 调用相关联的句柄。

qryRespAccessData **accessListP (O)
指向 qryRespAccessData 元素的指针，这些是 API 库分配的元素。 每个元素与访问规则相对应。 该数组中的元素数在 **numberOfRules** 参数中返回。 每个 qryRespAccessData 元素中返回的信息包括以下内容：

Name	描述
ruleNumber	访问规则的标识。 它可识别删除的规则。
AccessType	备份会归档类型。
Node	被授予访问权的节点。
Owner	被授予访问权的用户。
objName	高级别或低级别文件空间描述符。

dsUint32_t *numberOfRules (O)
返回 accessList 组的规则数。

dsmQueryApiVersion

dsmQueryApiVersion 函数调用可执行应用程序客户机访问的 API 库版本的查询请求。

对 API 的所有更新都以向上兼容格式进行。 对于其版本或发行版低于或等于最终用户工作站上的 API 库的 API，其操作不会有任何更改。 请注意，在继续该操作前，**dsmQueryApiVersion** 调用应返回旧于该应用程序客户机的版本或发行版，一些 API 可能可通过最终用户较旧的 API 版本不支持的方式来增强功能。

应用程序 API 版本号存储在 dsmapitd.h 头文件中 作为常量 DSM_API_VERSION、DSM_API_RELEASE 和 DSM_API_LEVEL。

不存在特定于此调用的返回码。

语法

```
void dsmQueryApiVersion  (dsmApiVersion *apiVersionP);
```

参数

dsmApiVersion *apiVersionP (O)
此参数是指向包含 API 库版本、发行版和级别组件的结构的指针。 例如，如果该库是 V1.1.0，那么，从此调用返回后，结构字段包含以下值：

```
dsmApiVersionP->version  = 1
dsmApiVersionP->release  = 1
dsmApiVersionP->level    = 0
```

dsmQueryApiVersionEx

dsmQueryApiVersionEx 函数调用可执行应用程序客户机访问的 API 库版本的查询请求。

对 API 的所有更新都以向上兼容格式进行。 对于其版本或发行版低于或等于最终用户工作站上的 API 库的 API，其操作不会有任何更改。 关于向上兼容性的异常，请参阅 README_api_enu 文件中的代码更改的摘要。 如果 **dsmQueryApiVersionEx** 调用返回与应用程序客户机不同的版本或发行版，请注意，在继续该操作前，一些 API 可能可通过最终用户较旧的 API 版本不支持的方式来增强功能。

应用程序 API 版本号作为常量 DSM_API_VERSION、DSM_API_RELEASE、DSM_API_LEVEL 和 DSM_API_SUBLEVEL 存储在 dsmapitd.h 头文件中。

不存在特定于此调用的返回码。

语法

```
void dsmQueryApiVersionEx (dsmApiVersionEx *apiVersionP);
```

参数

dsmApiVersionEx *apiVersionP (O)
此参数是指向包含 API 库的版本、发行版、级别和自级别组件的结构的指针。 例如，如果该库为 V5.5.0.0，那么从调用返回后，结构字段包含以下值：

- ApiVersionP->version = 5
- ApiVersionP->release = 5
- ApiVersionP->level = 0
- ApiVersionP->subLevel = 0

dsmQueryCliOptions

dsmQueryCliOptions 函数调用可查询用户的选项文件中重要选项的值。 **optStruct** 类型的结构在此调用中传递，并包含此信息。 此调用在调用 **dsmInitEx** 前执行，并在此会话前决定设置。

不存在特定于此调用的返回码。

语法

```
dsInt16_t dsmQueryCliOptions
    (optStruct     *optstructP);
```

参数

optStruct *optstructP (I/O)
此参数传递 API 完成的结构的地址。 应用程序客户机负责分配结构的存储。 成功返回后，将在结构的字段中输入相应信息。

optStruct 结构中将返回以下信息：

Name	描述
dsmiDir	环境 DSMI_DIR 变量的值。
dsmiConfig	由 DSMI_CONFIG 环境变量指定的客户机选项文件。
serverName	IBM Spectrum Protect 服务器的名称。
commMethod	选中的通信方法。 请参阅 dsmapitd.h 文件中的 #defines for DSM_COMM_*。
serverAddress	基于通信方法的服务器的地址。
nodeName	客户机节点（机器）名。
compression	此字段提供有关压缩选项的信息。
passwordAccess	这些值为：*bTrue* 用于生成，*bFalse* 用于提示。

相关概念
处理选项

dsmQuerySessInfo

dsmQuerySessInfo 函数调用启动对 IBM Spectrum Protect 的查询请求以获取有关 **dsmHandle** 中指定会话的操作的信息。 **ApiSessInfo** 类型的结构传递到此调用，并输入所有可用会话相关的信息。 此调用在成功进行 **dsmInitEx** 调用后开始。

在 **ApiSessInfo** 结构中返回的信息包括以下内容：

- 服务器信息：端口号、日期和时间和类型
- 客户机缺省值：应用程序类型、删除权限、定界符和事务限制
- 会话信息：登录标识和所有者
- 策略数据：域、活动策略集和保留宽限期

关于已传递的结构上下文以及其中的每个字段的信息，请参阅第 133 页的『附录 B API 类型定义源文件』。

语法

```
dsInt16_t dsmQuerySessInfo  (dsUint32_t         dsmHandle,
   ApiSessInfo   *SessInfoP);
```

参数

dsUint32_t dsmHandle (I)
将此调用与之前的 **dsmInitEx** 调用相关联的句柄。

ApiSessInfo *SessInfoP (I/O)
此参数传递 API 输入的结构的地址。 应用程序客户机负责分配结构的存储，并负责完成指示使用的结构版本的字段条目。 成功返回后，将在结构的字段中填写相应信息。 **adsmServerName** 是在 IBM Spectrum Protect 服务器上的 **define server** 命令中指定的名称。如果 archiveRetentionProtection 字段为 true，那么服务器将启用保留保护，

返回码

返回码编号用括号 ( ) 括起。

表 *42.* 为 *dsmQuerySessInfo* 返回代码

返回码	说明
DSM_RC_NO_SESS_BLK (2006)	没有服务器会话块信息。
DSM_RC_NO_POLICY_BLK (2007)	没有可用的服务器策略信息。
DSM_RC_WRONG_VERSION_PARM (2065)	应用程序客户机的 API 版本与 IBM Spectrum Protect 库的版本不同。

dsmQuerySessOptions

dsmQuerySessOptions 函数调用可查询 **dsmHandle** 的指定会话中有效的重要选项值。 `optStruct` 类型的结构在此调用中传递，并包含此信息。

此调用在成功进行 **dsmInitEx** 调用后开始。 返回的值可能与在 **dsmQueryCliOptions** 调用中返回的值不同，这由传递到 **dsmInitEx** 调用的值决定，主要是 `optString` 和 `optFile`。 关于选项优先顺序的信息，请参阅第 1 页的『了解配置文件和选项文件』。

不存在特定于此调用的返回码。

语法

```
dsInt16_t dsmQuerySessOptions
    (dsUint32_t    dsmHandle,
    optStruct      *optstructP);
```

参数

dsUint32_t dsmhandle(I)
将此调用与之前的 **dsmInitEx** 调用相关联的句柄。

optStruct *optstructP (I/O)
此参数传递 API 完成的结构的地址。 应用程序客户机负责分配结构的存储。 成功返回后，将在结构的字段中填写相应信息。

optStruct 结构中返回的信息为：

Name	描述
dsmiDir	DSMI_DIR 环境变量的值。
dsmiConfig	DSMI_CONFIG 环境变量指定的 dsm.opt 文件。
serverName	选项文件中 IBM Spectrum Protect 服务器节的名称。
commMethod	选中的通信方法。 请参阅 dsmapitd.h 文件中的 #defines for DSM_COMM_*。
serverAddress	基于通信方法的服务器的地址。
nodeName	客户机节点的名称（机器）。
压缩 (compression)	压缩选项的值（bTrue=on 和 bFalse=off）。
compressAlways	compressalways 选项的值（bTrue=on 和 bFalse=off）。
passwordAccess	值 bTrue 用于生成，bFalse 用于提示。

相关概念

处理选项

dsmRCMsg

dsmRCMsg 函数调用可获取与 API 返回码相关联的消息文本。

要点: 如果在 **dsmInitEx** 失败后调用 **dsmRCMsg**，那么将忽略 Servername 选项，并且将在缺省情况下针对服务器定义的日志文件中打印一个错误。

msg 参数在括号 ( ) 中显示消息前缀返回码，后接消息文本。 例如，对 **dsmRCMsg** 的调用可能返回以下内容：

```
ANS0264E (RC2300) 仅 root 用户可执行 dsmChangePW 或 dsmDeleteFS。
```

对于某些特征在 ANSII 和 OEM 代码页中有所不同的语言，可能必须在打印前，将字符串从 ANSII 转换为 OEM（例如，东欧单字节字符集）。 示例如下：

```
    dsmRCMsg(dsmHangle, rc, msgBuf);
    #ifdef WIN32
    #ifndef WIN64
```

```
    CharToOemBuff(msgBuf, msgBuf, strlen(msgBuf));
    #endif
    #endif
    printf("
```

语法

```
dsInt16_t dsmRCMsg  (dsUint32_t       dsmHandle,
   dsInt16_t        dsmRC,
   char        *msg);
```

参数

dsUint32_t dsmHandle (I)
将此调用与之前的 **dsmInitEx** 调用相关联的句柄。

dsInt16_t dsmRC (I)
相关联的消息文本的 API 返回码。 dsmrc.h 文件中列出了 API 返回码。 请参阅第 123 页的『附录 A API 返回码源文件：dsmrc.h』以获取更多信息。

char *msg (O)
此参数是与返回码 **dsmRC** 相关联的消息文本。 调用者负责为消息文本分配足够的空间。

msg 的最大长度如 DSM_MAX_RC_MSG_LENGTH 中所定义。

在具有本地语言支持和语言消息文件选择的平台上，API 将以本地语言返回消息字符串。

返回码

返回码编号用括号 ( ) 括起。

表 *43.* 为 *dsmRCMsg* 返回代码

返回码	说明
DSM_RC_NULL_MSG (2002)	针对 dsmRCMsg 调用的 **msg** 参数是 NULL 指针。
DSM_RC_INVALID_RETCODE (2021)	传递到 **dsmRCMsg** 调用的返回码是无效代码。
DSM_RC_NLS_CANT_OPEN_TXT (0610)	无法打开消息文本文件。

dsmRegisterFS

dsmRegisterFS 函数调用向 IBM Spectrum Protect 服务器注册新的文件空间。 在将任何数据备份到文件空间前，请先注册该文件空间。

应用程序客户机不应与备份/归档客户机使用相同的文件空间。

· 在 UNIX 或 Linux 上，为这些名称运行 **df** 命令。

· 在 Windows 上，这些名称通常是与系统中不同驱动器相关联的卷标。

语法

```
dsInt16_t dsmRegisterFS  (dsUint32_t         dsmHandle,
   regFSData     *regFilespaceP);
```

参数

dsUint32_t dsmHandle (I)
将此调用与之前的 **dsmInitEx** 调用相关联的句柄。

regFSData *regFilespaceP (I)
此参数传递需要向 IBM Spectrum Protect 服务器注册的文件空间的名称及相关信息。

提示: *fstype* 字段包括前缀 “**API:**”。 所有文件空间查询均显示此字符串。 例如，如果用户在 **dsmRegisterFS** 中传递 *fstype* 的 *myfstype*，那么服务器上的实际值字符串在查询时将返回为 API:myfstype。 此前缀可区分 API 对象和备份/归档对象。

fsInfo 的可用区域现在为 DSM_MAX_USER_FSINFO_LENGTH。

返回码

返回码编号用括号 ( ) 括起。

表 *44.* 为 *dsmRegisterFS* 返回代码

返回码	说明
DSM_RC_INVALID_FSNAME (2016)	无效的文件空间名称。
DSM_RC_INVALID_DRIVE_CHAR (2026)	盘符不是字母字符。
DSM_RC_NULL_FSNAME (2027)	Null 文件空间名称。
DSM_RC_FS_ALREADY_REGED (2062)	文件空间已注册。
DSM_RC_WRONG_VERSION_PARM (2065)	应用程序客户机的 API 版本与 IBM Spectrum Protect 库的版本不同。
DSM_RC_FSINFO_TOOLONG (2106)	文件空间信息过长。

dsmReleaseBuffer

dsmReleaseBuffer 函数返回 IBM Spectrum Protect 的缓冲区。应用程序在调用 **dsmGetDataEx** 后，且已将所有数据移出缓冲区并准备发布时调用 **dsmReleaseBuffer**。 **dsmReleaseBuffer** 要求在 ***UseTsmBuffers*** 设为 *btrue* 的情况下调用 **dsmInitEx**，并为 *numTsmBuffers* 提供非零的值。 如果应用程序即将调用 **dsmTerminate** 并且仍保留数据缓冲区，那么还应调用 **dsmReleaseBuffer**。

dsmReleaseBufferSyntax

```
dsInt16_t  dsmReleaseBuffer  (releaseBufferIn_t     *dsmReleaseBufferInP,
                              releaseBufferOut_t    *dsmReleaseBufferOutP) ;
```

参数

releaseBufferIn_t * dsmReleaseBufferInP (I)
此结构包含以下输入参数。

dsUint32_t dsmHandle (I)
将此调用与之前的 **dsmInitEx** 调用相关联的句柄。

dsUint8_t tsmBufferHandle(I)
识别此缓冲区的句柄。

char *dataPtr(I)
写入此应用程序的地址。

返回码

返回码编号用括号 ( ) 括起。

表 *45.* 为 *dsmReleaseBuffer* 返回代码

返回码	说明
DSM_RC_BAD_CALL_SEQUENCE	此调用没有在适当的状态下发出。
DSM_RC_INVALID_TSMBUFFER	句柄或 **dataPtr** 的值无效。
DSM_RC_BUFF_ARRAY_ERROR	发生缓冲区数组错误。

dsmRenameObj

dsmRenameObj 函数调用可重命名高级别或低级别对象名称。 对于备份对象，将传入当前对象名称并更改高级别或低级别对象名称。 对于归档对象，将传入当前对象文件空间名称和对象标识，并更改高级别或低级别对象名称。 在 **dsmBeginTxn** 和 **dsmEndTxn** 调用中使用此函数调用。

合并标志决定重复的备份对象名称是否与现有备份合并。 如果新的名称与现有对象相对应，且合并为 true，那么当前对象将转换为新的名称，并当具有该名称的现有活动对象成为对象最不活动的副本时变为新名称的活动版本。 如果新名称与现有对象相对应，且合并为 false，那么此函数将返回返回码 DSM_RC_ABORT_DUPLICATE_OBJECT。

限制:

- 仅对象的所有者可以重命名该对象。
- 如果在 IBM Spectrum Protect 服务器上启用数据保留保护，或者如果连接到 IBM Spectrum Protect for Data Retention 服务器，那么不支持 **dsmRenameObj** 函数。

dsmRenameObj 函数对以下三种合并条件调用测试:

- 当前 **dsmObjName** 对象和新的高级别或低级别对象必须与所有者、副本组和管理类匹配。
- 当前 **dsmObjName** 的备份必须新于使用新名称的当前活动对象。
- 必须只有当前 **dsmObjName** 的活动副本，而没有不活动的副本。

语法

```
dsInt16_t dsmRenameObj  (dsmRenameIn_t     *dsmRenameInP,
                         dsmRenameOut_t    *dsmRenameOutP);
```

参数

dsUint32_t dsmHandle (I)
将此调用与之前的 **dsmInitEx** 调用相关联的句柄。

dsmRenameIn_t *dsmRenameInP
此结构包含输入参数。

dsUint8_t repository (I);
此参数指示要删除的文件空间是位于备份存储库中还是归档存储库中。

dsmObjName *objNameP (I);
此参数是指向包含当前文件空间名称、高级别对象名称、低级别对象名称和对象类型的结构的指针。

char newHl [DSM_MAX_HL_LENGTH + 1];
此参数指定新的高级别名称。

char newLl [DSM_MAX_LL_LENGTH + 1];
此参数指定新的低级别名称。

dsBool_t merge;
此参数决定备份对象是否与重复命名的对象合并。 该值为 true 或 false。

ObjID;
归档对象的对象标识。

dsmRenameOut_t *dsmRnameOutP
此结构包含输出参数。

注: 没有任何输出参数。

返回码

返回码编号用括号 ( ) 括起。

表 *46. dsmRenameObj* 的返回码

返回码	说明
DSM_RC_ABORT_MERGE_ERROR (45)	服务器检测到合并错误。
DSM_RC_ABORT_DUPLICATE_OBJECT (32)	对象已存在，合并为 false。
DSM_RC_ABORT_NO_MATCH (2)	没有找到对象。
DSM_RC_REJECT_SERVER_DOWNLEVEL (58)	IBM Spectrum Protect 服务器必须为 V3.7.4.0 或更高版本时此函数才能运行。

dsmRequestBuffer

dsmRequestBuffer 函数返回 IBM Spectrum Protect 的缓冲区。应用程序在调用 **dsmGetDataEx** 后，且已将所有数据移出缓冲区并准备发布时调用 **dsmRequestBuffer**。

dsmReleaseBuffer 要求在 *UseTsmBuffers* 设为 *btrue* 的情况下调用 **dsmInitEx**，并为 *numTsmBuffers* 提供非零的值。 如果应用程序将调用 **dsmTerminate** 且仍保留 IBM Spectrum Protect 缓冲区，那么，还应调用 **dsmReleaseBuffer**。

语法

```
dsInt16_t  dsmRequestBuffer   (getBufferIn_t      *dsmRequestBufferInP,
                               getBufferOut_t     *dsmRequestBufferOutP) ;
```

参数

getBufferIn_t * dsmRequestBufferInP (I)
此结构包含以下输入参数:

dsUint32_t dsmHandle
识别会话并将其与之前的 **dsmInitEx** 调用相关联的句柄。

getBufferOut_t *dsmRequestBufferOut P (0)
此结构包含输出参数。

dsUint8_t tsmBufferHandle(0)
识别此缓冲区的句柄。

char *dataPtr(0)
写入应用程序的地址。

dsUint32_t *bufferLen(0)
可以写入此缓冲区的最大字节数。

返回码

返回码编号用括号 ( ) 括起。

表 *47. dsmRequestBuffer* 的返回码

返回码	说明
DSM_RC_BAD_CALL_SEQUENCE (33)	此调用没有在适当的状态下发出。
DSM_RC_SENDDATA_WITH_ZERO_SIZE (34)	如果正在发送的对象长度为 0，将不允许调用 **dsmReleaseBuffer**。
DSM_RC_BUFF_ARRAY_ERROR (121)	无法获取有效的缓冲区。

dsmRetentionEvent

dsmRetentionEvent 函数调用将对象标识列表发送到 IBM Spectrum Protect 服务器，并对这些对象执行保留事件操作。在 **dsmBeginTxn** 和 **dsmEndTxn** 调用中使用此函数调用。

注: 服务器必须为 V5.2.2.0 或更高版本时此函数才能运行。

一个调用中的最大对象数限制为 *maxObjPerTxn*，该值是从 **dsmQuerySessInfo** 调用的 *ApisessInfo* 结构中返回的。

仅对象的所有者可以对该对象发送事件。

以下为可能的事件:

eventRetentionActivate
仅可对与基于事件的管理类绑定的对象发送。 发送此事件可为该对象激活事件，且该对象的保留状态将从 DSM_ARCH_RETINIT_PENDING 更改为 DSM_ARCH_RETINIT_STARTED。

eventHoldObj
此事件将发出保留或暂时禁用删除该对象的消息，从而在发布此下达前，该对象不会过期且无法删除。

eventReleaseObj
此事件只能针对 ***objectHeld*** 字段中值为 DSM_ARCH_HELD_TRUE 的对象发出，并除去继续使用原保留策略的对象的暂时禁用。

发送 ***dsmRetentionEvent*** 前，需先发送第 26 页的『查询 IBM Spectrum Protect 系统』中所述的查询顺序以获取关于该对象的信息。 对 **dsmGetNextQObj** 的调用将返回名为 **qryRespArchiveData** 的数据结构以便归档查询。 此数据结构包含 **dsmRetentionEvent** 所需的信息。

语法

```
extern dsInt16_t DSMLINKAGE dsmRetentionEvent(
  dsmRetentionEventIn_t          *ddsmRetentionEventInP,
  dsmRetentionEventOut_t         *dsmRetentionEventOutP
   );
```

参数

dsmRetentionEventIn_t *dsmRetentionEventP
此结构包含以下输入参数:

dsUint16_t stVersion;
此参数指示结构版本。

dsUint32_t dsmHandle (I)
将此调用与之前的 **dsmInitEx** 调用相关联的句柄。

dsmEventType_t evenType (I);
此参数指示事件类型。 关于以下可能值的意义的信息，请参阅本节的开头：
eventRetentionActivate、**eventHoldObj**、**eventReleaseObj**

dsmObjList_t objList;
此参数指示要发信号的对象标识的列表。

返回码

返回码编号用括号 ( ) 括起。

表 *48. dsmRetentionEvent* 的返回码

返回码	说明
DSM_RC_ABORT_NODE_NOT_AUTHORIZED (36)	节点或用户不具有适当权限。
DSM_RC_ABORT_TXN_LIMIT_EXCEEDED (249)	该事务中的对象太多。
DSM_RC_ABORT_OBJECT_ALREADY_HELD (250)	该对象已保留，无法发出其他保留消息。
DSM_RC_REJECT_SERVER_DOWNLEVEL (58)	服务器必须为 V5.2.2.0 或更高版本时此函数才能运行。

dsmSendBufferData

dsmSendBufferData 函数调用通过在先前的 **dsmReleaseBuffer** 调用中提供的缓冲区将数据的字节流发送至 IBM Spectrum Protect。应用程序客户机可在服务器的存储库上传递任何数据类型。 通常，此数据是文件数据，但并不仅限于文件数据。 如果发送的数据字节流过大，您可以多次调用 **dsmSendBufferData**。 无论调用成功与否，都将释放缓冲区。

限制: 使用 *useTsmBuffers* 选项时，即使压缩包含一个对象，也不会压缩此对象。

语法

```
dsInt16_t  dsmSendBufferData   (sendBufferDataIn_t        *dsmSendBufferDataExInP,
                                sendBufferDataOut_t       *dsmSendBufferDataOutP) ;
```

参数

sendBufferDataIn_t * dsmSendBufferDataInP (I)
此结构包含以下输入参数。

dsUint32_t dsmHandle (I)
将此调用与之前的 **dsmInitEx** 调用相关联的句柄。

dsUint8_t tsmBufferHandle(I)
识别要发送的缓冲区的句柄。

char *dataPtr(I)
写入应用程序数据的地址。

dsUint32_t numBytes(I)
由应用程序写入的实际字节数（应始终小于 **dsmReleaseBuffer** 中提供的值）。

返回码

返回码编号用括号 ( ) 括起。

表 *49. dsmSendBufferData* 的返回码

返回码	说明
DSM_RC_BAD_CALL_SEQUENCE (2041)	此调用没有在适当的状态下发出。
DSM_RC_INVALID_TSMBUFFER (2042)	句柄或 **dataPtr** 的值无效。
DSM_RC_BUFF_ARRAY_ERROR (2045)	发生缓冲区数组错误。
DSM_RC_TOO_MANY_BYTES (2043)	***numBytes*** 的值大于 **dsmReleaseBuffer** 调用中提供的缓冲区的大小。

dsmSendData

dsmSendData 函数调用可通过缓冲区将数据字节流发送至 IBM Spectrum Protect。 应用程序客户机可在服务器的存储库上传递任何数据类型。 通常，这些数据是文件数据，但不仅限于文件数据。 如果想要发送的数据字节流过大，您可以多次调用 **dsmSendData**。

限制: 应用程序客户机在 **dsmSendData** 调用返回前，无法复用 **dsmSendData** 中指定的缓冲区。

提示: 如果 IBM Spectrum Protect 返回代码 157 (DSM_RC_WILL_ABORT)，将开始对 **dsmEndSendObj** 的调用，然后再使用 DSM_VOTE_COMMIT 的表决调用 **dsmEndTxn**。 然后，应用程序检索返回码 2302 (DSM_RC_CHECK_REASON_CODE) 并将原因码传回给应用程序用户。这可告知用户服务器结束此事务的原因。

语法

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

参数

dsUint32_t dsmHandle (I)
将此调用与之前的 **dsmInitEx** 调用相关联的句柄。

DataBlk *dataBlkPtr (I/O)
此参数指向一个结构，该结构包括指向要从中发送数据的缓冲区的指针及此缓冲区的大小。 返回时，此结构包含实际转移的字节数。 关于类型定义的信息，请参阅第 133 页的『附录 B API 类型定义源文件』。

返回码

返回码编号用括号 ( ) 括起。

表 *50.* 为 *dsmSendData* 返回代码

返回码	说明
DSM_RC_NO_COMPRESS_MEMORY (154)	可用内存不足，无法执行数据压缩或展开。
DSM_RC_COMPRESS_GREW (155)	压缩期间，压缩数据的大小相较于原数据会不断增加。
DSM_RC_WILL_ABORT (157)	发生未知和意外错误，将导致该事务停止。

表 *50.* 为 *dsmSendData* 返回代码 *(续)*

返回码	说明
DSM_RC_WRONG_VERSION_PARM (2065)	应用程序客户机的 API 版本与 IBM Spectrum Protect 库版本不同。
DSM_RC_NEEDTO_ENDTXN (2070)	需要结束该事务。
DSM_RC_OBJ_EXCLUDED (2080)	包含/排除列表排除此对象。
DSM_RC_OBJ_NOBCG (2081)	此对象没有备份副本组，不会发送至服务器。
DSM_RC_OBJ_NOACG (2082)	此对象没有归档副本组，不会发送至服务器。
DSM_RC_SENDDATA_WITH_ZERO_SIZE (2107)	此对象无法发送 *sizeEstimate* 为零字节的数据。

dsmSendObj

dsmSendObj 函数调用可开始将单个对象发送至存储库的请求。 由于性能原因，可在事务的边界内执行多个 **dsmSendObj** 调用及相关的 **dsmSendData** 调用。

dsmSendObj 调用可在内存缓冲器中传递字节流时处理该对象的数据。 **dsmSendObj** 调用中的 **dataBlkPtr** 参数运行应用程序客户机执行以下任意操作：

· 在单个调用中传递数据及对象的属性（这些属性通过 **objAttrPtr** 传递）。

· 通过 **dsmSendObj** 调用指定对象数据的部分，通过一个或多个 **dsmSendData** 调用指定数据的剩余部分。

或者，应用程序客户机可以通过 **dsmSendObj** 调用仅指定属性，并通过对 **dsmSendData** 的一个或多个调用指定对象数据。 对于此方法，可在 **dsmSendObj** 调用中将 **dataBlkPtr** 设置为 NULL。

提示: 对于某些对象类型，字节流数据可能无法与数据相关联；例如，没有扩展属性的目录项。

调用 **dsmSendObj** 前，必须先执行前面的 **dsmBindMC** 调用以便将管理类正确绑定到想要备份或归档的对象。 API 会保留此绑定，以便在将其发送至服务器时，可将正确的管理类与对象相关联。 如果允许在 **dsmSendObj** 调用中绑定的管理类作为目录的对象类型 (DSM_OBJ_DIRECTORY) 的缺省值，那么该缺省值可能不是缺省的管理类。 相反，可以使用保留时间最长的管理类。 如果存在多个具有此保留时间的管理类，那么，将使用第一个遇到的管理类。

dsmEndSendObj 调用后接所有发送至存储库的对象数据。 如果您没有要发送至服务器的对象数据，或所有数据均包含在 **dsmSendObj** 调用中，请在开始其他 **dsmSendObj** 调用前先开始 **dsmEndSendObj** 调用。 如果要求通过 **dsmSendData** 调用进行多个数据的发送，**dsmEndSendObj** 将跟在最后发送的数据后以指示状态更改。

提示: 如果 IBM Spectrum Protect 返回代码 157 (DSM_RC_WILL_ABORT)，将开始使用 DSM_VOTE_COMMIT 的表决调用 **dsmEndTxn**。 应用程序接收返回码 2302 (DSM_RC_CHECK_REASON_CODE) 并将原因码传回给应用程序用户。这可告知用户服务器结束此事务的原因。

如果原因码为 11 (DSM_RS_ABORT_NO_REPOSIT_SPACE)，可能是由于 *sizeEstimate* 对于数据的实际数量而言过小。 应用程序需要确定更精确的 *sizeEstimate*，然后再次发送此数据。

语法

```
dsInt16_t dsmSendObj (dsUint32_t       dsmHandle,
   dsmSendType  sendType,
   void        *sendBuff,
   dsmObjName  *objNameP,
   ObjAttr     *objAttrPtr,
   DataBlk     *dataBlkPtr);
```

参数

dsUint32_t dsmHandle (I)
将此调用与之前的 **dsmInitEx** 调用相关联的句柄。

dsmSendType sendType (I)
此参数指定正在执行的发送类型。 可能的值包括：

Name	描述
stBackup	发送至服务器的备份对象。
stArchive	发送至服务器的归档对象。
stBackupMountWait	希望服务器等待，直至必要设备（如磁带）安装完成的备份对象。
stArchiveMountWait	希望服务器等待，直至必要设备（如磁带）安装完成的归档对象。

注: 如果应用程序用户有可能将数据发送至磁带，那么请使用 **MountWait** 类型。

void *sendBuff (I)
此参数是结构的指针，该结构包含特定于调用中的 **sendType** 的其他信息。 当前，仅 **stArchive** 的 **sendType** 有相关联的结构。 此结构称为 **sndArchiveData**，它包含归档描述。

dsmObjName *objNameP (I)
此参数是结构的指针，该结构包含文件空间名称、高级别对象名称、低级别对象名称和对象类型。 请参阅第 18 页的『对象名和标识』以获取更多信息。

ObjAttr *objAttrPtr (I)
此参数将传递应用程序感兴趣的对象属性。 关于类型定义的信息，请参阅第 133 页的『附录 B API 类型定义源文件』。

这些属性为：

- **owner** 指对象的所有者。 在从 IBM Spectrum Protect 存储库取回对象时，确定所有者是被声明为特定名称还是空字符串非常重要。 请参阅第 20 页的『以会话所有者身份访问对象』以获取更多信息。
- **sizeEstimate** 是要发送至服务器的数据对象的总大小的准确估计。 对于此大小请尽可能的准确，因为服务器将使用此属性在其存储资源中进行合理的资源分配和对象放置。

  如果指定的大小估计明显小于发送的实际字节数，服务器可能很难分配足够的空间，并将结束此事务，原因码为 11 (DSM_RS_ABORT_NO_REPOSIT_SPACE)。

  注: 大小估计是针对数据对象的总大小（以字节表示）。

  不会压缩小于 DSM_MIN_COMPRESS_SIZE 的对象。

  如果对象没有位数据（只有来自此调用的属性信息），那么 **sizeEstimate** 应为零。

  注: 从 V5.1.0 开始，不会对长度为零的对象的检查事物中的复制目的地以保持一致性。
- **objCompressed** 是布尔值，说明是否已压缩该对象数据。

  如果已压缩该对象（对象 *compressed=bTrue*），IBM Spectrum Protect 不会再次尝试对它进行压缩。 如果未进行压缩，IBM Spectrum Protect 会根据管理员在 API 配置源中设置的 compression 选项的值来决定是否压缩对象。

  如果应用程序计划使用部分对象恢复或检索，在发送该数据时您无法压缩它。 要强制实施此操作，请将 *ObjAttr.objCompressed* 设置为 *bTrue*。
- **objInfo** 会保存有关特殊对象的信息。

  限制: 信息不会自动存储在此处。 使用此属性时，必须设置属性 *objInfoLength* 以显示 *objInfo* 的长度。
- **mcNameP** 包含覆盖从 **dsmBindMC** 获取的管理类的管理类名称。
- **disableDeduplication** 是布尔值。 当它设置为 true 时，客户机不会对此对象进行重复数据删除。

DataBlk *dataBlkPtr (I/O)
此参数指向包括指向要备份或归档的数据缓冲区的指针及该缓冲区的大小的结构。 此参数仅适用于 **dsmSendObj**。 如果想要在后续 **dsmSendData** 调用而非 **dsmSendObj** 调用中发送数据，请在 DataBlk 结构中将缓冲区指针设为 NULL。 返回时，此结构包含实际转移的字节数。 关于类型定义的信息，请参阅第 133 页的『附录 B API 类型定义源文件』。

返回码

返回码编号用括号 ( ) 括起。

表 *51.* 为 *dsmSendObj* 返回代码

返回码	说明
DSM_RC_NO_COMPRESS_MEMORY (154)	可用内存不足，无法执行数据压缩或展开。
DSM_RC_COMPRESS_GREW (155)	压缩期间，压缩数据的大小相较于原数据会不断增加。
DSM_RC_WILL_ABORT (157)	发生未知和意外错误，将导致该事务停止。
DSM_RC_TL_NOACG (186)	此文件的管理类不具有针对发送类型有效的副本组。
DSM_RC_NULL_OBJNAME (2000)	Null 对象名称。
DSM_RC_NULL_OBJATTRPTR (2004)	Null 对象属性指针。
DSM_RC_INVALID_OBJTYPE (2010)	对象类型无效。
DSM_RC_INVALID_OBJOWNER (2019)	无效的对象所有者。
DSM_RC_INVALID_SENDTYPE (2022)	发送类型无效。
DSM_RC_WILDCHAR_NOTALLOWED (2050)	不允许通配符。
DSM_RC_FS_NOT_REGISTERED (2061)	没有注册文件空间。
DSM_RC_WRONG_VERSION_PARM (2065)	应用程序客户机的 API 版本与 IBM Spectrum Protect 库的版本不同。
DSM_RC_NEEDTO_ENDTXN (2070)	需要结束事务。
DSM_RC_OBJ_EXCLUDED (2080)	包含/排除列表排除了此对象。
DSM_RC_OBJ_NOBCG (2081)	此对象没有备份副本组，不会发送至服务器。
DSM_RC_OBJ_NOACG (2082)	此对象没有归档备份组，不会发送至服务器。
DSM_RC_DESC_TOOLONG (2100)	描述过长。
DSM_RC_OBJINFO_TOOLONG (2101)	对象信息过长。
DSM_RC_HL_TOOLONG (2102)	高级别限定符过长。
DSM_RC_FILESPACE_TOOLONG (2104)	文件空间名称过长。
DSM_RC_LL_TOOLONG (2105)	低级别限定符过长。
DSM_RC_NEEDTO_CALL_BINDMC (2301)	必须先调用 **dsmBindMC**。

dsmSetAccess

dsmSetAccess 函数调用可授予其他用户或节点对象的备份版本或归档副本的访问权、所有对象的访问权或选择设置的访问权。 在授予另一用户访问权时，该用户可以查询、恢复或检索您的文件。 此命令支持以下字段的通配符：*fs*、*hl*、*ll*、*node*、*owner*。

注: 使用单个命令不能同时授予对于备份版本和归档副本的访问权。 必须指定备份或归档。

语法

```
dsInt16_t DSMLINKAGE dsmSetAccess
        (dsUint32_t          dsmHandle,
         dsmSetAccessType    accessType,
         dsmObjName          *objNameP,
         char                *node,
         char                *owner);
```

参数

dsUint32_t dsmHandle (I)
将此调用与之前的 **dsmInitEx** 调用相关联的句柄。

dsmAccessType accessType (I)
此参数指定您想要授予访问权的对象类型。 可能的值包括：

Name	描述
atBackup	指定要为备份对象设置的放问权。
atArchive	指定要为归档对象设置的访问权。

dsmObjName *objNameP (I)
此参数是指向包含文件空间名称、高级别对象名称和低级别对象名称的结构的指针。

注: 要指定所有文件空间，请在文件空间名称位置使用星号 (*)。

char *node (I)
此参数是指向授予访问权的节点名的指针。 对于任何节点，均请指定星号 (*)。

char *owner (I)
此参数是指向授予访问权的节点上的用户名的指针。 对于所有用户，均请指定星号 (*)。

返回码

返回码编号用括号 ( ) 括起。

表 *52.* 为 *dsmSetAccess* 返回代码

返回码	说明
DSM_RC_INVALID_ACCESS_TYPE (2110)	指定了无效的访问类型。
DSM_RC_FILE_SPACE_NOT_FOUND (124)	服务器上找不到指定的文件空间。
DSM_RC_QUERY_COMM_FAILURE (2111)	服务器查询期间的通信错误。
DSM_RC_NO_FILES_BACKUP (2112)	没有为此文件空间备份文件。
DSM_RC_NO_FILES_ARCHIVE (2113)	没有为此文件空间归档文件。
DSM_RC_INVALID_SETACCESS (2114)	设置访问权的无效表述。

dsmSetUp

dsmSetUp 函数调用可覆盖环境变量值。 在调用 **dsmInitEx** 前先调用 **dsmSetUp**。 在 **envSetUp** 结构中传递的值可覆盖现有的环境变量或缺省值。 如果为字段指定 NULL，将从环境中获取值。 如果没有设置值，将从缺省值中获取值。

需求:

1. 如果使用 **dsmSetUp**，将始终在调用 **dsmCleanUp** 前调用 **dsmTerminate**。
2. 如果测试标记在配置文件中设置 INSTRUMENT: API，并在应用程序中使用 **dsmSetUp** 或 **dsmCleanUp**，将只能激活 API 指令。

语法

```
dsInt16_t DSMLINKAGE dsmSetUp
        (dsBool_t    mtFlag,
         envSetUp    *envSetUpP);
```

参数

dsBool_t mtFlag (I)
此参数指定将在单个县城还是多线程模式中使用 API。 值包括:

```
DSM_SINGLETHREAD
DSM_MULTITHREAD
```

要求: 必须打开多线程标志才会出现不依赖 LAN 的数据传输。

envSetUp *envSetUpP(I)
此参数是指向保留覆盖值的结构的指针。 如果不想覆盖现有的环境变量，请指定 NULL。 **envSetUp** 中的字段 结构包括:

Name	描述
dsmiDir	包含 UNIX 或 Linux 上的消息文件的标准目录路径。 其还指定 dsm.sys 目录。
dsmiConfig	客户机选项文件的标准名称。
dsmiLog	错误日志目录的标准路径。
argv	如果应用程序必须使用授权用户权限运行，那么请传递调用程序的 argv[0] 名称。请参阅第 17 页的『控制对密码文件的访问』以获取更多信息。
logName	应用程序没有使用 dsierror.log 时错误日志的文件名称。
inclExclCaseSensitive	指示包含/排除规则是区分大小写还是不区分大小写。 此参数只能在 Windows 上使用，其他任何位置上都将被忽略。

返回码

返回码编号用括号 ( ) 括起。

表 *53.* 为 *dsmSetUp* 返回代码

返回码	说明
DSM_RC_ACCESS_DENIED (106)	对指定文件或目录的访问被拒绝。
DSM_RC_INVALID_OPT (0400)	没有找到无效的选项。

表 53. 为 *dsmSetUp* 返回代码 (续)

返回码	说明
DSM_RC_NO_HOST_ADDR (0405)	没有在系统选项文件的服务器名称节中定义该服务器的 TCPSERVERADDRESS。
DSM_RC_NO_OPT_FILE (0406)	没有找到由文件名指定的选项文件。
DSM_RC_MACHINE_SAME (0408)	选项文件中定义的 NODENAME 不能与系统 *HostName* 相同。
DSM_RC_INVALID_SERVER (0409)	系统选项文件不包含 SERVERNAME 选项。
DSM_RC_INVALID_KEYWORD (0410)	在 **dsmInitEx** 配置文件中找到了无效的选项关键字，选项字符串为 dsm.sys 或 dsm.opt。
DSM_RC_PATTERN_TOO_COMPLEX (0411)	发出的包含或排除模式太过复杂以至于 IBM Spectrum Protect 无法进行准确地解释。
DSM_RC_NO_CLOSING_BRACKET (0412)	包含或排除模式的构成不正确。 缺少右方括号。
DSM_RC_NLS_CANT_OPEN_TXT (0610)	系统无法打开消息文本文件。
DSM_RC_NLS_INVALID_CNTL_REC (0612)	系统无法使用消息文本文件。
DSM_RC_NOT_ADSM_AUTHORIZED (0927)	您必须是授权用户才能获取多线程和 *passwordaccess* generate。
DSM_RC_NO_INCLEXCL_FILE (2229)	没有找到包含/排除文件。
DSM_RC_NO_SYS_OR_INCLEXCL (2230)	既没有找到 dsm.sys 文件，也没有找到包含/排除文件。

dsmTerminate

dsmTerminate 函数调用可结束 IBM Spectrum Protect 服务器的会话并清除了 IBM Spectrum Protect 环境。

语法

不存在特定于此调用的返回码。

```
dsInt16_t dsmTerminate (dsUint32_t dsmHandle);
```

参数

dsUint32_t dsmHandle (I)
将此调用与之前的 **dsmInitEx** 调用相关联的句柄。

dsmUpdateFS

dsmUpdateFS 函数调用将更新 IBM Spectrum Protect 存储器中的文件空间。 此更新确保管理员拥有文件空间的当前记录。

语法

```
dsInt16_t dsmUpdateFS  (dsUint32_t      dsmHandle,
   char         *fs,
   dsmFSUpd     *fsUpdP,
   dsUint32_t    fsUpdAct);
```

参数

dsUint32_t dsmHandle (I)
将此调用与之前的 **dsmInitEx** 调用关联的句柄。

char *fs (I)
此参数是指向文件空间名称的指针。

dsmFSUpd *fsUpdP (I)
此参数是一个指向结构的指针，该结构中包含针对您所需更新的正确字段。 仅完成那些需要更新的字段。

dsUint32_t fsUpdAct (I)
指示需要更新的字段的二字节位映射。 位掩码具有以下值：

- DSM_FSUPD_FSTYPE
- DSM_FSUPD_FSINFO

  提示: 对于 Windows 操作系统，在选中 **FSINFO** 时还会更新 **dsmDOSAttrib** 中的盘符值。
- DSM_FSUPD_OCCUPANCY
- DSM_FSUPD_CAPACITY
- DSM_FSUPD_BACKSTARTDATE
- DSM_FSUPD_BACKCOMPLETEDATE

有关这些位掩码的描述，请参阅以下主题中的 DSM_FSUPD 定义：第 133 页的『附录 B API 类型定义源文件』。

返回码

下表列出了 **dsmUpdateFS** 函数调用的返回码。

表 *54.* 为 *dsmUpdateFS* 返回代码

返回码	返回码编号	描述
DSM_RC_FS_NOT_REGISTERED	2061	未注册文件空间名称。
DSM_RC_WRONG_VERSION_PARM	2065	应用程序客户机的 API 版本不同于 IBM Spectrum Protect 库版本。
DSM_RC_FSINFO_TOOLONG	2106	文件空间信息过长。

dsmUpdateObj

dsmUpdateObj 函数调用可更新与服务器上已有的活动备份或归档对象相关联的元信息。 应用程序位数据不会受到影响。 要更新对象，必须指定特定的非通配符名称。 要更新已归档的对象，请将 **dsmSendType** 设为 **stArchive**。 仅更新新命名的归档对象。

可以在会话状态中仅开始 **dsmUpdateObj** 调用；无法在事务内进行调用，因为它仅执行自己的事务。 此外，您一次只能更新一个对象。

限制: 在 UNIX 或 Linux 操作系统中，如果更改了所有者的字段，除非您是 root 用户，否则无法查询或恢复对象。

语法

```
dsInt16_t  dsmUpdateObj
   (dsUint32_t      dsmHandle,
    dsmSendType     sendType,
    void           *sendBuff,
    dsmObjName     *objNameP,
    ObjAttr        *objAttrPtr,  /* objInfo */
    dsUint16_t      objUpdAct);  /* action bit vector */
```

参数

字段描述与 **dsmSendObj** 中的描述相同，以下是例外情况：

dsmObjName *objNameP (I)
无法使用通配符。

ObjAttr *objAttrPtr (I)
此调用将忽略 **objCompressed** 字段。

其他差异为：

- **owner**。 如果指定新的 **owner** 字段，将更改所有者。
- **sizeEstimate**。 如果指定非零的值，该值应为实际发送的数据量（以字节表示）。 该值存储在 IBM Spectrum Protect 元数据中以供将来使用。
- **objInfo**。 此属性包含要放在 **objInfo** 字段中的新信息。 将 **objInfoLength** 设为新的 **obiInfo** 的长度。

dsUint16_t objUpdAct
objUpdAct 的位掩码和可能的操作包括：

DSM_BACKUPD_MC
更新该对象的管理类。

DSM_BACKUPD_OBJINFO
更新 **objInfo**、**objInfoLength** 和 **sizeEstimate**。

DSM_BACKUPD_OWNER
更新该对象的所有者。

DSM_ARCHUPD_DESCR
更新 **Description** 字段。 通过 **SendBuff** 参数输入新描述的值。 关于正确的使用方法，请参阅样本程序。

DSM_ARCHUPD_OBJINFO
更新 **objInfo**、**objInfoLength** 和 **sizeEstimate**。

DSM_ARCHUPD_OWNER
更新该对象的所有者。

返回码

返回码编号用括号 ( ) 括起。

表 *55. dsmUpdateObj* 的返回码

返回码	说明
DSM_RC_INVALID_ACTION (2232)	无效的操作。
DSM_RC_FS_NOT_REGISTERED (2061)	没有注册文件空间。

表 55. *dsmUpdateObj* 的返回码 *(续)*

返回码	说明
DSM_RC_BAD_CALL_SEQUENCE (2041)	调用顺序无效。
DSM_RC_WILDCHAR_NOTALLOWED (2050)	不允许使用通配符。
DSM_RC_ABORT_NO_MATCH (2)	与之前的查询不匹配。

dsmUpdateObjEx

dsmUpdateObjEx 函数调用可更新与服务器上已有的活动备份或归档对象相关联的元信息。 应用程序位数据不会受到影响。 要更新对象，必须指定非通配符名称，或指定对象标识以更新特定的归档对象。 指定名称时，无法使用通配符。 要更新备份对象，请将 **dsmSendType** 参数设置为 **stBackup**。 要更新已归档的对象，请将 **dsmSendType** 参数设置为 **stArchive**。

可以在会话状态中仅开始 **dsmUpdateObjEx** 调用；无法在事务内进行调用，因为它仅执行自己的事务。 一次只能更新一个对象。

限制: 在 UNIX 或 Linux 操作系统中，如果更改了所有者的字段，除非您是 root 用户，否则无法查询或恢复对象。 只能更新备份对象的当前归档版本。

语法

```
dsInt16_t  dsmUpdateObjEx
   (dsmUpdateObjExIn_t   *dsmUpdateObjExInP,
    dsmUpdateObjExOut_t  *dsmUpdateObjExOutP);
```

参数

dsmUpdateObjExIn_t *dsmUpdateObjExInP
此结构包含以下输入参数：

dsUint16_t stVersion (I)
所使用的结构的当前版本。

dsUint32_t dsmHandle (I)
将此调用与之前的 **dsmInitEx** 调用相关联的句柄。

dsmSendType sendType (I)
正在执行的发送类型。 该值可以为：

stBackup
发送至服务器的备份对象。

stArchive
发送至服务器的归档对象。

dsmObjName *objNameP (I)
指向包含文件空间名称、高级别对象名称、低级别对象名称和对象类型的结构的指针。 无法使用通配符。

ObjAttr *objAttrPtr (I)
将对象属性传递给应用程序。 更新的值取决于 **objUpdAct** 字段中的标志。 此调用将忽略 **objCompressed** 属性。

这些属性为：

· 如果输入新的名称，**owner** 将更改所有者。

- **sizeEstimate** 是实际发送数据量（以字节表示）。 该值存储在 IBM Spectrum Protect 元数据中以便将来使用。
- **objCompressed** 是布尔值，说明是否已压缩该对象数据。
- **objInfo** 是包含要放在 **objInfo** 字段中的新信息的属性。 将 **objInfoLength** 设为新的 **objInfo** 的长度。
- **mcNameP** 包含覆盖从 **dsmBindMC** 获取的管理类的管理类名称。

dsUint32_t objUpdAct
指定 **objUpdAct** 的位掩码和操作：

DSM_BACKUPD_MC
更新该对象的管理类。

DSM_BACKUPD_OBJINFO
更新备份对象的信息对象 (**objInfo**)、该信息对象的长度 (**objInfoLength**) 和发送的数据量 (**sizeEstimate**)。

DSM_BACKUPD_OWNER
更新备份对象的所有者。

DSM_ARCHUPD_DESCR
更新归档对象的 **Description** 字段。 通过 **sendBuff** 参数输入新描述的值。

DSM_ARCHUPD_OBJINFO
更新归档对象的信息对象 (**objInfo**)、信息对象的长度 (**objInfoLength**) 和发送的数据量 (**sizeEstimate**)。

DSM_ARCHUPD_OWNER
更新归档对象的所有者。

ObjID archObjId
为特定归档对象指定唯一的对象标识。 由于多个归档对象可以有多个相同的名称，因此此参数可识别特定的一个对象。 可以使用查询归档调用获取对象标识。

dsmUpdateObjExOut_t *dsmUpdateObjExOutP
此结构包含输出参数：

dsUint16_t stVersion (I)
所使用的结构的当前版本。

返回码

下表中返回码编号用括号 ( ) 括起。

表 *56. Return codes for dsmUpdateObjEx*

返回码	说明
DSM_RC_INVALID_ACTION (2012)	无效的操作。
DSM_RC_FS_NOT_REGISTERED (2061)	没有注册文件空间。
DSM_RC_BAD_CALL_SEQUENCE (2041)	调用顺序无效。
DSM_RC_WILDCHAR_NOTALLOWED (2050)	不允许使用通配符。
DSM_RC_ABORT_NO_MATCH (2)	与之前的查询不匹配。

附录 A API 返回码源文件：dsmrc.h

dsmrc.h 头文件包含 API 可以返回给应用程序的所有返回码。

此处提供的信息包含随 API 一起分发的 dsmrc.h 文件的时间点副本。查看 API 发行软件包中的文件以获取最新版本。

```
/*****************************************************************************
*       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                  *
*****************************************************************************/

/*****************************************************************************/
/* 头文件名称：dsmrc.h                                                      */
/*                                                                             */
/* 描述性名称：来自于 IBM Spectrum Protect API 的返回码               */
/*****************************************************************************/
#ifndef _H_DSMRC
#define _H_DSMRC


#ifndef DSMAPILIB

#ifndef _H_ANSMACH
typedef int RetCode ;
#endif

#endif


#define DSM_RC_SUCCESSFUL                0 /* 成功完成                */
#define DSM_RC_OK                        0 /* 成功完成                */

#define DSM_RC_UNSUCCESSFUL             -1 /* 未成功完成              */

/* dsmEndTxn reason code */
#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    /* 部分对象检索            */
#define DSM_RS_ABORT_INVALID_LENGTH          34    /* 部分对象检索            */
#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


/* RETURN CODE */

#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

/* 服务器登录拒绝代码的定义                                          */
/* 这些代码错误包含在返回内（51 到 99）。                          */
#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 /* 客户机类型未获得许可    */
#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 /* 用户中止处理                          */
#define DSM_RC_NO_MEMORY           102 /* 没有剩余 RAM 来完成请求            */
#define DSM_RC_TA_COMM_DOWN       2021 /* 不再使用                               */
#define DSM_RC_FILE_NOT_FOUND      104 /* 没有找到指定的文件                    */
#define DSM_RC_PATH_NOT_FOUND      105 /* 不存在指定的路径                      */
#define DSM_RC_ACCESS_DENIED       106 /* 由于权限不正确遭拒绝                  */
#define DSM_RC_NO_HANDLES          107 /* 没有更多可用的文件句柄                */
#define DSM_RC_FILE_EXISTS         108 /* 文件已存在                            */
#define DSM_RC_INVALID_PARM        109 /* 传递了无效的参数。  CRITICAL*/
#define DSM_RC_INVALID_HANDLE      110 /* 传递了无效的文件句柄                  */
#define DSM_RC_DISK_FULL           111 /* 磁盘空间不足                          */
#define DSM_RC_PROTOCOL_VIOLATION  113 /* 调用协议违例。  CRITICAL */
#define DSM_RC_UNKNOWN_ERROR       114 /* 未知的系统错误。  CRITICAL */
#define DSM_RC_UNEXPECTED_ERROR    115 /* 意料外的错误。  CRITICAL */
#define DSM_RC_FILE_BEING_EXECUTED 116 /* 不允许写入                            */
#define DSM_RC_DIR_NO_SPACE        117 /* 无法展开目录                          */
#define DSM_RC_LOOPED_SYM_LINK     118 /* 转换路径中遇到
                                          过多的符号链接。   */
#define DSM_RC_FILE_NAME_TOO_LONG  119 /* 文件名称过长                          */
#define DSM_RC_FILE_SPACE_LOCKED   120 /* 系统锁定了文件空间                    */
#define DSM_RC_FINISHED            121 /* 完成处理                              */
#define DSM_RC_UNKNOWN_FORMAT      122 /* 未知的格式                            */
#define DSM_RC_NO_AUTHORIZATION    123 /* 客户机没有权限读取其他主机的
                                          所有者备份/归档数据时的
                                          服务器响应                            */
#define DSM_RC_FILE_SPACE_NOT_FOUND 124/* 没有找到指定的文件空间                */
#define DSM_RC_TXN_ABORTED         125 /* 中止事务                              */
#define DSM_RC_SUBDIR_AS_FILE      126 /* 子目录名称作为文件存在                */
#define DSM_RC_PROCESS_NO_SPACE    127 /* 处理没有更多磁盘空间可用。    */
#define DSM_RC_PATH_TOO_LONG       128 /* 正在构建的目录路径
                                          过长                                  */
#define DSM_RC_NOT_COMPRESSED      129 /* 想要压缩的文件
                                          实际没有压缩                          */
#define DSM_RC_TOO_MANY_BITS       130 /* 压缩文件所使用的位多于
                                          扩展器可处理的位                      */
#define DSM_RC_SYSTEM_ERROR        131 /* 内部系统错误                          */
#define DSM_RC_NO_SERVER_RESOURCES 132 /* 服务器资源不足。           */
#define DSM_RC_FS_NOT_KNOWN        133 /* 文件空间对于服务器
                                          未知                                  */
#define DSM_RC_NO_LEADING_DIRSEP   134 /* 没有主要目录分隔符                    */
#define DSM_RC_WILDCARD_DIR        135 /* 通配符在不允许的情况下位于
                                          目录路径中                            */
#define DSM_RC_COMM_PROTOCOL_ERROR 136 /* 通信协议错误                          */
#define DSM_RC_AUTH_FAILURE        137 /* 认证失败                              */
#define DSM_RC_TA_NOT_VALID        138 /* TA 不是 root 和/或 SUID 程序      */
#define DSM_RC_KILLED              139 /* 结束处理。                      */

#define DSM_RC_RETRY               143 /* 再次重试相同的操作                    */

#define DSM_RC_WOULD_BLOCK         145 /* 操作将导致系统
                                          不再等待输入。           */
#define DSM_RC_TOO_SMALL           146 /* 用于编译模式的区域较小             */
#define DSM_RC_UNCLOSED            147 /* 模式中没有右方括号                    */
#define DSM_RC_NO_STARTING_DELIMITER 148 /* 模式必须以
                                            目录分界符开头                    */
#define DSM_RC_NEEDED_DIR_DELIMITER 149 /*  “匹配目录”
                                          元字
                                          符串
                                          ("...") 前后都需要紧跟分界符，其中一处没有找到*/
#define DSM_RC_UNKNOWN_FILE_DATA_TYPE 150 /* 结构文件数据类型
                                             未知                               */
#define DSM_RC_BUFFER_OVERFLOW     151 /* 数据缓冲区溢出                        */


#define DSM_RC_NO_COMPRESS_MEMORY  154 /* 压缩/展开内存不足                     */
#define DSM_RC_COMPRESS_GREW       155 /* 压缩增加                              */
#define DSM_RC_INV_COMM_METHOD     156 /* 指定无效的通信方式                    */
#define DSM_RC_WILL_ABORT          157 /* 将中止事务                            */
#define DSM_RC_FS_WRITE_LOCKED     158 /* 写入文件空间已锁定                    */
#define DSM_RC_SKIPPED_BY_USER     159 /* 在 ABORT_DATA_OFFLINE 的情况下，
```

```
                                        用户希望跳过文件                    */
#define DSM_RC_TA_NOT_FOUND        160 /* 没有在 TA 目录中找到 TA          */
#define DSM_RC_TA_ACCESS_DENIED    161 /* 访问 TA 遭拒绝                    */
#define DSM_RC_FS_NOT_READY        162 /* 文件空间未就绪                    */
#define DSM_RC_FS_IS_BAD           163 /* 文件空间已损坏                    */
#define DSM_RC_FIO_ERROR           164 /* 文件输入/输出错误                 */
#define DSM_RC_WRITE_FAILURE       165 /* 写入文件出错                      */
#define DSM_RC_OVER_FILE_SIZE_LIMIT 166 /* 文件超出系统/用户限制            */
#define DSM_RC_CANNOT_MAKE         167 /* 无法创建文件/目录，
                                        可能名称有错误                    */
#define DSM_RC_NO_PASS_FILE        168 /* 需要密码文件且用户
                                        不是 root 用户                    */
#define DSM_RC_VERFILE_OLD         169 /* 本地存储的密码
                                        与主机的密码不匹配                */
#define DSM_RC_INPUT_ERROR         173 /* 未能读取键盘输入                  */
#define DSM_RC_REJECT_PLATFORM_MISMATCH 174 /* 平台名称与
                                           服务器识别为
                                           客户机的平台不一致             */
#define DSM_RC_TL_NOT_FILE_OWNER   175 /* 尝试备份文件的用户
                                        不是文件的所有者                  */
#define DSM_RC_COMPRESSED_DATA_CORRUPTED 176 /* 压缩的数据已损坏            */
#define DSM_RC_UNMATCHED_QUOTE     177 /* 缺少上引号或下引号                */

#define DSM_RC_SIGNON_FAILOVER_MODE 178      /* 已经故障转移到复制服务器，
                                                正在以故障转移方式运行 */
#define DSM_RC_FAILOVER_MODE_FUNC_BLOCKED 179 /* 函数被阻止，因为会话
                                                 处于故障转移方式           */

/*---------------------------------------------------------------------------*/
/* 保留返回码 180-199 以便策略集处理                                         */
/*---------------------------------------------------------------------------*/
#define DSM_RC_PS_MULTBCG          181 /* 1 MC 中有多个备份组               */
#define DSM_RC_PS_MULTACG          182 /* 1 MC 中有多个归档  备份组          */
#define DSM_RC_PS_NODFLTMC         183 /* 缺省的 MC 名称不在策略集中        */
#define DSM_RC_TL_NOBCG            184 /* 需要备份，没有备份副本组          */
#define DSM_RC_TL_EXCLUDED         185 /* 需要备份，被包含/排除过滤器过滤   */
#define DSM_RC_TL_NOACG            186 /* 需要归档，没有归档副本组          */
#define DSM_RC_PS_INVALID_ARCHMC   187 /* 归档覆盖中存在无效的 MC 名称      */
#define DSM_RC_NO_PS_DATA          188 /* 服务器中没有策略集数据            */
#define DSM_RC_PS_INVALID_DIRMC    189 /* 选项文件中指定了
                                        无效的目录 MC。                   */
#define DSM_RC_PS_NO_CG_IN_DIR_MC  190 /* 目录 MC 中没有备份副本组。
                                        必须使用 DirMC 选项指定 MC
                                        选项的设置。                          */

#define DSM_RC_WIN32_UNSUPPORTED_FILE_TYPE 280 /* 文件不是
                                                  Win32 类型 FILE_TYPE_DISK  */

/*---------------------------------------------------------------------------*/
/* 用于可信通信代理程序的返回码                                              */
/*---------------------------------------------------------------------------*/
#define DSM_RC_TCA_NOT_ROOT        161 /* 访问 TA 遭拒绝                    */
#define DSM_RC_TCA_ATTACH_SHR_MEM_ERR  200 /* 附加共享内存出错              */
#define DSM_RC_TCA_SHR_MEM_BLOCK_ERR   200 /* 共享内存块出错                */
#define DSM_RC_TCA_SHR_MEM_IN_USE      200 /* 共享内存块出错                */
#define DSM_RC_TCA_SHARED_MEMORY_ERROR 200 /* 共享内存块出错                */
#define DSM_RC_TCA_SEGMENT_MISMATCH    200 /* 共享内存块出错                */
#define DSM_RC_TCA_FORK_FAILED     292 /* 分割 TCA 处理出错                 */
#define DSM_RC_TCA_DIED            294 /* TCA 意外死机                      */
#define DSM_RC_TCA_INVALID_REQUEST 295 /* 向 TVA 发送了无效请求             */
#define DSM_RC_TCA_SEMGET_ERROR    297 /* 获取信号出错                      */
#define DSM_RC_TCA_SEM_OP_ERROR    298 /* 在设置或等待信号时出错            */
#define DSM_RC_TCA_NOT_ALLOWED     299 /* 不允许 TCA ( 多线程 )             */

/*---------------------------------------------------------------------------*/
/* 400-430 用于选项                                                          */
/*---------------------------------------------------------------------------*/
#define DSM_RC_INVALID_OPT           400 /* 无效的选项                      */
#define DSM_RC_NO_HOST_ADDR          405 /* 没有可连接服务器的 enuf 信息    */
#define DSM_RC_NO_OPT_FILE           406 /* 没有缺省用户配置文件            */
#define DSM_RC_MACHINE_SAME          408 /* -MACHINENAME 与实际名称相同     */
#define DSM_RC_INVALID_SERVER        409 /* 客户机中的无效服务器名称        */
#define DSM_RC_INVALID_KEYWORD       410 /* 无效的选项关键字                */
#define DSM_RC_PATTERN_TOO_COMPLEX   411 /* 无法匹配包含/排除项             */
#define DSM_RC_NO_CLOSING_BRACKET    412 /* 缺少右方括号 ( 包含/排除 )      */
```

```
#define DSM_RC_OPT_CLIENT_NOT_ACCEPTING 417/* 客户机不接受来自此服务器的
                                            该值                          */
#define DSM_RC_OPT_CLIENT_DOES_NOT_WANT 418/* 客户机不想要来自此服务器的
                                            该值                          */
#define DSM_RC_OPT_NO_INCLEXCL_FILE  419   /* 没有找到包含/排除文件          */
#define DSM_RC_OPT_OPEN_FAILURE      420   /* 无法打开文件                   */
#define DSM_RC_OPT_INV_NODENAME      421/* 当 CLUSTERNODE=YES 时，如果 nodename=本地
                                          机器，就用于 Windows             */
#define DSM_RC_OPT_NODENAME_INVALID  423/* 一般的无效节点名                */
#define DSM_RC_OPT_ERRORLOG_CONFLICT 424/* 指定了 logmax 和保留时间            */
#define DSM_RC_OPT_SCHEDLOG_CONFLICT 425/* 指定了 logmax 和保留时间            */
#define DSM_RC_CANNOT_OPEN_TRACEFILE 426/* 无法打开跟踪文件                */
#define DSM_RC_CANNOT_OPEN_LOGFILE   427/* 无法打开错误日志文件            */
#define DSM_RC_OPT_SESSINIT_LF_CONFLICT 428/* 指定了 sessioninit=server 及
                                            enablelanfree=yes             */
#define DSM_RC_OPT_OPTION_IGNORE     429/* 将忽略选项                      */
#define DSM_RC_OPT_DEDUP_CONFLICT    430/* 无法打开错误日志文件            */
#define DSM_RC_OPT_HSMLOG_CONFLICT 431/* 指定了 logmax 和保留时间          */


/*----------------------------------------------------------------------------*/
/* 卷标签代码为 600 至 610                                                    */
/*----------------------------------------------------------------------------*/
#define DSM_RC_DUP_LABEL           600 /* 找到重复的卷标签                    */
#define DSM_RC_NO_LABEL            601 /* 驱动器没有标签                      */

/*----------------------------------------------------------------------------*/
/* 用于消息文件处理的返回码                                                   */
/*----------------------------------------------------------------------------*/
#define DSM_RC_NLS_CANT_OPEN_TXT   610 /* 尝试打开 msg txt 文件时出错         */
#define DSM_RC_NLS_CANT_READ_HDR   611 /* 尝试读取表头时出错                  */
#define DSM_RC_NLS_INVALID_CNTL_REC 612 /* 无效的控制记录                     */
#define DSM_RC_NLS_INVALID_DATE_FMT 613 /* 无效的缺省日期格式                 */
#define DSM_RC_NLS_INVALID_TIME_FMT 614 /* 无效的缺省时间格式                 */
#define DSM_RC_NLS_INVALID_NUM_FMT 615 /* 无效的缺省数字格式                  */

/*----------------------------------------------------------------------------*/
/* 保留返回码 620-630 以便用于日志消息返回码                                  */
/*----------------------------------------------------------------------------*/
#define DSM_RC_LOG_CANT_BE_OPENED  620 /* 尝试打开错误日志时出错              */
#define DSM_RC_LOG_ERROR_WRITING_TO_LOG 621 /* 写入日志文件时
                                               出错                        */
#define DSM_RC_LOG_NOT_SPECIFIED   622 /* 没有指定错误日志文件                */


/*----------------------------------------------------------------------------*/
/* 仅返回代码 900-999 IBM SPECTRUM PROTECT CLIENT                             */
/*----------------------------------------------------------------------------*/
#define DSM_RC_NOT_ADSM_AUTHORIZED  927 /* 必须授权 ADSM 才能执行             */
                                        /* 操作：root 用户或密码授权          */
#define DSM_RC_REJECT_USERID_UNKNOWN 940 /* 服务器上的用户标识未知            */
#define DSM_RC_FILE_IS_SYMLINK      959 /* 错误日志或跟踪为信号
                                           链接
                                                                          */

#define DSM_RC_DIRECT_STORAGE_AGENT_UNSUPPORTED 961 /* 不支持直接链接到 SA             */
#define DSM_RC_FS_NAMESPACE_DOWNLEVEL 963 /* 已将长名称空间从
                                             从 Netware 卷中除去       */
#define DSM_RC_CONTINUE_NEW_CONSUMER   972 /* 使用新的使用者继续处理             */
#define DSM_RC_CONTINUE_NEW_CONSUMER_NODEDUP   973 /* 使用新的使用者继续处理，不要重复            */
#define DSM_RC_CONTINUE_NEW_CONSUMER_NOCOMPRESS 976 /* 使用新的使用者继续处理，无压缩 */


#define DSM_RC_SERVER_SUPPORTS_FUNC    994 /* 服务器支持此功能                 */
#define DSM_RC_SERVER_AND_SA_SUPPORT_FUNC 995 /* 服务器和 SA 都支持此功能      */
#define DSM_RC_SERVER_DOWNLEVEL_FUNC   996 /* 此服务器对于函数级别过低          */
#define DSM_RC_STORAGEAGENT_DOWNLEVEL  997 /* 存储库代理程序是低级别            */
#define DSM_RC_SERVER_AND_SA_DOWNLEVEL 998 /* 服务器和 SA 都是低级别            */


/* TCP/IP 错误代码   */
#define DSM_RC_TCPIP_FAILURE       -50 /* TCP/IP 通信故障                    */
#define DSM_RC_CONN_TIMEDOUT       -51 /* TCP/IP 连接尝试超时                */
#define DSM_RC_CONN_REFUSED        -52 /* TCP/IP 连接被主机拒绝              */
#define DSM_RC_BAD_HOST_NAME       -53 /* 指定了 TCP/IP 无效主机名           */
#define DSM_RC_NETWORK_UNREACHABLE -54 /* TCP/IP 主机名无法连接              */
```

```
#define DSM_RC_WINSOCK_MISSING     -55 /* 缺少 TCP/IP WINSOCK.DLL             */
#define DSM_RC_TCPIP_DLL_LOADFAILURE -56 /* LoadLibrary 中存在错误           */
#define DSM_RC_TCPIP_LOADFAILURE   -57 /* GetProcAddress 中存在错误          */
#define DSM_RC_TCPIP_USER_ABORT    -58 /* 用户在 TCP/IP 层中被中止           */

/*-----------------------------------------------------------------------------*/
/* 保留返回 (-71)-(-90) 以便用于 CommTSM 错误代码                              */
/*-----------------------------------------------------------------------------*/
#define DSM_RC_TSM_FAILURE         -71 /* IBM Spectrum Protect 命令失败      */
#define DSM_RC_TSM_ABORT           -72 /* 会话异常中止                       */

/*comm3270 错误代码 - 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                          */


/* 以下返回码适用于 Windows 的 EHLLAPI                    */
#define DSM_RC_3270COMMError_DLL        2021   /* 不再使用                    */
#define DSM_RC_3270COMMError_GetProc    2021   /* 不再使用                    */
#define DSM_RC_EHLLAPIError_DLL         2021   /* 不再使用                    */
#define DSM_RC_EHLLAPIError_GetProc     2021   /* 不再使用                    */
#define DSM_RC_EHLLAPIError_HostConnect 2021   /* 不再使用                    */
#define DSM_RC_EHLLAPIError_AllocBuff   2021   /* 不再使用                    */
#define DSM_RC_EHLLAPIError_SendKey     2021   /* 不再使用                    */
#define DSM_RC_EHLLAPIError_PacketChk   2021   /* 不再使用                    */
#define DSM_RC_EHLLAPIError_ChkSum      2021   /* 不再使用                    */
#define DSM_RC_EHLLAPIError_HostTimeOut 2021   /* 不再使用                    */
#define DSM_RC_EHLLAPIError_Send        2021   /* 不再使用                    */
#define DSM_RC_EHLLAPIError_Recv        2021   /* 不再使用                    */
#define DSM_RC_EHLLAPIError_General     2021   /* 不再使用                    */
#define DSM_RC_PC3270_MISSING_DLL       2021   /* 不再使用                    */
#define DSM_RC_3270COMM_MISSING_DLL     2021   /* 不再使用                    */


/* NETBIOS 错误代码      */
#define DSM_RC_NETB_ERROR         -151 /* 无法将节点添加至 LAN               */
#define DSM_RC_NETB_NO_DLL        -152 /* 无法加载 ACSNETB.DLL                */
#define DSM_RC_NETB_LAN_ERR       -155 /* 检测到 LAN 错误                     */
#define DSM_RC_NETB_NAME_ERR      -158 /* 添加名称时出现 Netbios 错误        */
#define DSM_RC_NETB_TIMEOUT       -159 /* Netbios 发送超时                    */
#define DSM_RC_NETB_NOTINST       -160 /* 没有安装 Netbios - DOS              */
#define DSM_RC_NETB_REBOOT        -161 /* Netbios 配置出错 - 重新引导 DOS    */

/* 命名管道错误代码   */
#define DSM_RC_NP_ERROR                       -190

/* CPIC 错误代码   */
#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            */
```

```
/*------------------------------------------------------------------------*/
/* 保留返回码 -300 至 -307 以便用于 IPX/SPX 通信                          */
/*------------------------------------------------------------------------*/
#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           */

/*=== 共享内存协议错误代码                       ===*/
#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 /* 对象名称指针为 NULL            */
#define DSM_RC_NULL_DATABLKPTR      2001 /* dataBlkPtr 为 NULL               */
#define DSM_RC_NULL_MSG             2002 /* dsmRCMsg 中的 msg parm 为 NULL   */

#define DSM_RC_NULL_OBJATTRPTR      2004 /* 对象属性指针为 NULL            */

#define DSM_RC_NO_SESS_BLK          2006 /*没有服务器会话信息              */
#define DSM_RC_NO_POLICY_BLK        2007 /* 没有策略 hdr 信息              */
#define DSM_RC_ZERO_BUFLEN          2008 /* dataBlkPtr 的 bufferLen 为零     */
#define DSM_RC_NULL_BUFPTR          2009 /* dataBlkPtr 的 bufferPtr 为 NULL  */

#define DSM_RC_INVALID_OBJTYPE      2010 /* 无效的对象类型                 */
#define DSM_RC_INVALID_VOTE         2011 /* 无效的表决                      */
#define DSM_RC_INVALID_ACTION       2012 /* 无效的操作                      */
#define DSM_RC_INVALID_DS_HANDLE    2014 /* 无效的 ADSM 句柄               */
#define DSM_RC_INVALID_REPOS        2015 /* 无效的存储库值                 */
#define DSM_RC_INVALID_FSNAME       2016 /* fs 应以目录分界符开头          */
#define DSM_RC_INVALID_OBJNAME      2017 /* 无效的完整路径名称             */
#define DSM_RC_INVALID_LLNAME       2018 /* ll 应以目录分界符开头          */
#define DSM_RC_INVALID_OBJOWNER     2019 /* 无效的对象所有者名称           */
#define DSM_RC_INVALID_ACTYPE       2020 /* 无效的操作类型                 */
#define DSM_RC_INVALID_RETCODE      2021 /* dsmRCMsg 中的 dsmRC 无效         */
#define DSM_RC_INVALID_SENDTYPE     2022 /* 无效的发送类型                 */
#define DSM_RC_INVALID_PARAMETER    2023 /* 无效的参数                      */
#define DSM_RC_INVALID_OBJSTATE     2024 /* 活动、非活动还是任何匹配？  */
#define DSM_RC_INVALID_MCNAME       2025 /* 没有找到管理类名称             */
#define DSM_RC_INVALID_DRIVE_CHAR   2026 /* 盘符不是字母                    */
#define DSM_RC_NULL_FSNAME          2027 /* 文件空间名称为 NULL            */
#define DSM_RC_INVALID_HLNAME       2028 /* hl 应以目录分界符开头          */

#define DSM_RC_NUMOBJ_EXCEED        2029 /* BeginGetData 对象数超额          */

#define DSM_RC_NEWPW_REQD           2030 /* 新密码为必填字段               */
#define DSM_RC_OLDPW_REQD           2031 /* 旧密码为必填字段               */
#define DSM_RC_NO_OWNER_REQD        2032 /* 不允许所有者。 可以是缺省值    */
#define DSM_RC_NO_NODE_REQD         2033 /* pw=generate 时不允许节点         */
#define DSM_RC_KEY_MISSING          2034 /* 没有找到密钥文件               */
#define DSM_RC_KEY_BAD              2035 /* 密钥文件的内容损坏             */

#define DSM_RC_BAD_CALL_SEQUENCE    2041 /* 不允许 DSM 调用的顺序          */
#define DSM_RC_INVALID_TSMBUFFER    2042 /* tsmbuffhandle 或 dataPtr 的值无效*/
#define DSM_RC_TOO_MANY_BYTES       2043 /* 复制到缓冲区的字节数过多       */
#define DSM_RC_MUST_RELEASE_BUFFER  2044 /* 无法退出应用程序以释放缓冲区   */
#define DSM_RC_BUFF_ARRAY_ERROR     2045 /* 内部缓冲区数组错误             */
#define DSM_RC_INVALID_DATABLK      2046 /* 使用 tsmbuff datablk 应为 null   */
#define DSM_RC_ENCR_NOT_ALLOWED     2047 /* 使用 tsmbuffers 时，不允许加密   */
#define DSM_RC_OBJ_COMPRESSED       2048 /* 在压缩对象上使用 tsmBuff 无法进行恢复*/
#define DSM_RC_OBJ_ENCRYPTED        2049 /* 使用 tsmbuff 无法恢复加密对象    */
#define DSM_RC_WILDCHAR_NOTALLOWED  2050 /* 不允许对 hl、ll 使用通配符       */
#define DSM_RC_POR_NOT_ALLOWED      2051 /* 使用 tsmBuffers 时无法使用部分对象恢复*/
#define DSM_RC_NO_ENCRYPTION_KEY    2052 /* 没有找到加密密钥               */
#define DSM_RC_ENCR_CONFLICT        2053 /* 互斥选项                         */

#define DSM_RC_FSNAME_NOTFOUND      2060 /* 没有找到文件空间名称           */
#define DSM_RC_FS_NOT_REGISTERED    2061 /* 没有注册文件空间名称           */
#define DSM_RC_FS_ALREADY_REGED     2062 /* 已注册文件空间名称             */
```

```
#define DSM_RC_OBJID_NOTFOUND      2063 /* 没有要恢复的对象标识              */
#define DSM_RC_WRONG_VERSION       2064 /* 错误的代码级别                    */
#define DSM_RC_WRONG_VERSION_PARM  2065 /* 错误的参数结构级别                */

#define DSM_RC_NEEDTO_ENDTXN       2070 /* 需要调用 dsmEndTxn                */

#define DSM_RC_OBJ_EXCLUDED        2080 /* 对象由 MC 排除                    */
#define DSM_RC_OBJ_NOBCG           2081 /* 对象没有备份副本组                */
#define DSM_RC_OBJ_NOACG           2082 /* 对象没有归档副本组                */

#define DSM_RC_APISYSTEM_ERROR     2090 /* API 内部错误                      */

#define DSM_RC_DESC_TOOLONG             2100 /* 描述过长                          */
#define DSM_RC_OBJINFO_TOOLONG          2101 /* 对象属性 objinfo 过长             */
#define DSM_RC_HL_TOOLONG               2102 /* 高级别限定符过长                  */
#define DSM_RC_PASSWD_TOOLONG           2103 /* 密码过长                          */
#define DSM_RC_FILESPACE_TOOLONG        2104 /* 文件名称过长                      */
#define DSM_RC_LL_TOOLONG               2105 /* 低级别限定符过长                  */
#define DSM_RC_FSINFO_TOOLONG           2106 /* 文件名称长度过大                  */
#define DSM_RC_SENDDATA_WITH_ZERO_SIZE  2107 /* 设为零时发送数据                  */

/*=== 用于 dsmaccess 的新的返回码   ===*/
#define DSM_RC_INVALID_ACCESS_TYPE 2110 /* 无效的访问类型                    */
#define DSM_RC_QUERY_COMM_FAILURE  2111 /* 查询期间出现通信错误              */
#define DSM_RC_NO_FILES_BACKUP     2112 /* 此 fs 没有备份文件                */
#define DSM_RC_NO_FILES_ARCHIVE    2113 /* 此 fs 没有归档文件                */
#define DSM_RC_INVALID_SETACCESS   2114 /* 无效的设置访问格式                */

/*=== 用于 dsmaccess 的新的返回码   ===*/
#define DSM_RC_STRING_TOO_LONG     2120 /* 字符串参数过长                    */

#define DSM_RC_MORE_DATA           2200 /* 存在更多要恢复的数据              */

#define DSM_RC_BUFF_TOO_SMALL      2210 /* DataBlk 缓冲区过小无法查询        */

#define DSM_RC_NO_API_CONFIGFILE   2228 /*没有找到指定的 API 配置文件        */
#define DSM_RC_NO_INCLEXCL_FILE    2229 /* 没有找到指定的包含/排除文件       */
#define DSM_RC_NO_SYS_OR_INCLEXCL  2230 /* 没有 dsm.sys 或没有在 dsm.sys 中
                                           找到指定的包含/排除文件           */
#define DSM_RC_REJECT_NO_POR_SUPPORT 2231 /* 服务器不支持 POR                */

#define DSM_RC_NEED_ROOT           2300 /* API 调用者必须为 root 用户        */
#define DSM_RC_NEEDTO_CALL_BINDMC  2301 /* 必须首先调用 dsmBindMC            */
#define DSM_RC_CHECK_REASON_CODE   2302 /* 检查 dsmEndTxn 的原因码           */
#define DSM_RC_NEEDTO_ENDTXN_DEDUP_SIZE_EXCEEDED 2303 /*超出删除重复数据的最大字节数*/

/*=== 关于由 lic 文件使用的返回码 2400 - 2410 请参阅 agentrc.h ===*/

/*=== 关于由 Oracle 代理程序使用的返回码 2410 - 2430 请参阅 agentrc.h ===*/

#define DSM_RC_ENC_WRONG_KEY       4580 /* 提供的密钥不正确                  */
#define DSM_RC_ENC_NOT_AUTHORIZED  4582 /* 不允许用户解密                    */
#define DSM_RC_ENC_TYPE_UNKNOWN    4584 /* 解密类型未知                      */

/*============================================================================
   保留返回码 (4600)-(4624) 以便进行集群
============================================================================*/
#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

/*============================================================================
   针对新的服务器 ABORT 代码保留返回码 (5200)-(5600) (dsmcomm.h)
============================================================================*/
#define DSM_RS_ABORT_CERTIFICATE_NOT_FOUND   5200

/*============================================================================
   保留返回码 (5701)-(5749) 以便用于代理模式
============================================================================*/
#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

/*============================================================================
   保留返回码 5801 - 5849 以便用于密码术/安全性
============================================================================*/

#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

/*============================================================================
   保留返回码 6300 - 6399 以便用于客户机端的重复数据删除
============================================================================*/
#define DSM_RC_DIGEST_VALIDATION_ERROR                6300  /* 端到端摘要确认出错               */
#define DSM_RC_DATA_FINGERPRINT_ERROR                 6301 /* Rabin 指纹识别时出现故障          */
#define DSM_RC_DATA_DEDUP_ERROR                       6302 /* 将数据转换为块时出错              */


#endif /* _H_DSMRC */
```

相关参考

API 返回码

附录 B API 类型定义源文件

本附录包含 API 的结构定义、类型定义和常量。

- 开始的头文件 `dsmapitd.h` 和 `tsmapitd.h` 说明了对所有操作系统公用的定义。
- 第二个头文件 `dsmapips.h` 提供特定于特定操作系统的定义示例；在本例中，为 Windows 平台。
- 第三个头文件 `release.h` 包含版本和发行版信息。

此处提供的信息包含随 API 一起分发的文件的时间点副本。查看 API 发行软件包中的文件以获取最新版本。

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                    *
***********************************************************************/

/**************************************************************************
* 表头文件名称：dsmapitd.h
*
* 环境：     ************************************************
*                  ** 这是与平台无关的源文件    **
*
*                  ************************************************
*
* 设计说明：       本文件包含可在所有客户机源文件中包括的
*                  基本数据类型和常量。  此文件中的常量
*                  应正确设置，以便适用于
*                  要运行客户机软件的
*                  特殊机器和操作系统。
*
*                  平台特定的定义包括在 dsmapips.h 中
*
* 描述性名称：IBM Spectrum Protect API 常量的定义
*-------------------------------------------------------------------------*/

#ifndef _H_DSMAPITD
#define _H_DSMAPITD


#include "dsmapips.h"      /* Platform specific definitions*/
#include "release.h"

/*=== 设置结构对齐以包装结构 ===*/
#if (_OPSYS_TYPE == DS_WINNT) && !defined(_WIN64)
#pragma pack(1)
#endif

#ifdef _MAC
/*=============================================================================
  选项包括：
http://developer.apple.com/documentation/DeveloperTools/Conceptual/PowerPCRuntime/Data/
chapter_2_section_3.html

#pragma option align=<mode>
where <mode> is power, mac68k, natural, or packed.
=============================================================================*/
#pragma options align=packed
#endif

typedef char osChar_t;

/*<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>*/
/*                     D E F I N E S                                      */
```

```
/*<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>*/
/*-------------------------------------------------------------------------+
|  要在 dsmInit() 的 dsmApiVersion 上使用的 API 版本、发行版和级别    |
+-------------------------------------------------------------------------*/
#define DSM_API_VERSION     COMMON_VERSION
#define DSM_API_RELEASE     COMMON_RELEASE
#define DSM_API_LEVEL       COMMON_LEVEL
#define DSM_API_SUBLEVEL    COMMON_SUBLEVEL

/*-------------------------------------------------------------------------+
| 最大字段长度                                                            |
+-------------------------------------------------------------------------*/
#define DSM_MAX_CG_DEST_LENGTH        30        /* 副本组目标             */
#define DSM_MAX_CG_NAME_LENGTH        30        /* 副本组名称             */
#define DSM_MAX_DESCR_LENGTH          255       /* 归档描述               */
#define DSM_MAX_DOMAIN_LENGTH         30        /* 策略域名称             */
#define DSM_MAX_FSINFO_LENGTH         500       /* 文件空间信息           */
#define DSM_MAX_USER_FSINFO_LENGTH    480       /* 最大用户文件空间信息   */
#define DSM_MAX_FSNAME_LENGTH         1024      /* 文件空间               */
#define DSM_MAX_FSTYPE_LENGTH         32        /* 文件空间类型           */
#define DSM_MAX_HL_LENGTH             1024      /* 对象高级别名称         */
#define DSM_MAX_ID_LENGTH             64        /* 会话节点名称           */
#define DSM_MAX_LL_LENGTH             256       /* 对象低级别名称         */
#define DSM_MAX_MC_NAME_LENGTH        30        /* 管理列名称             */
#define DSM_MAX_OBJINFO_LENGTH        255       /* 对象信息               */
#define DSM_MAX_EXT_OBJINFO_LENGTH    1500      /* 扩展对象信息           */
#define DSM_MAX_OWNER_LENGTH          64        /* 对象所有者名称         */
#define DSM_MAX_PLATFORM_LENGTH       16        /* 应用程序类型           */
#define DSM_MAX_PS_NAME_LENGTH        30        /* 策略集名称             */
#define DSM_MAX_SERVERTYPE_LENGTH     32        /* 服务器平台类型         */
#define DSM_MAX_VERIFIER_LENGTH       64        /* 密码                   */
#define DSM_PATH_MAX                  1024      /* API 配置文件路径       */
#define DSM_NAME_MAX                  255       /* API 配置文件名称       */
#define DSM_MAX_NODE_LENGTH           64        /* 节点/机器名称          */
#define DSM_MAX_RC_MSG_LENGTH         1024      /* dsmRCMsg 的 msg 参数   */
#define DSM_MAX_SERVER_ADDRESS        1024     /* 服务器地址              */

#define DSM_MAX_MC_DESCR_LENGTH       DSM_MAX_DESCR_LENGTH /* 管理类      */
#define DSM_MAX_SERVERNAME_LENGTH     DSM_MAX_ID_LENGTH /* 服务器名称     */
#define DSM_MAX_GET_OBJ               4080     /* BeginGetData 上的最大对象数*/
#define DSM_MAX_PARTIAL_GET_OBJ       1300 /* BeginGetData 上的最大部分对象数*/
#define DSM_MAX_COMPRESSTYPE_LENGTH   32       /* 最大压缩算法名称           */

/*-------------------------------------------------------------------------+
| 最小字段长度                                                            |
+-------------------------------------------------------------------------*/
#define DSM_MIN_COMPRESS_SIZE  2048 /* 对象的最小字节数                   */
                                    /* 允许压缩前需要                     */

/*-------------------------------------------------------------------------+
|  dsmSetup 调用中 mtFlag 的值                                            |
+-------------------------------------------------------------------------*/
#define DSM_MULTITHREAD      bTrue
#define DSM_SINGLETHREAD     bFalse

/*-------------------------------------------------------------------------+
|  dsmObjName 结构中对象类型的值                                          |
|  注意：这些值必须与 dsmcomm.h 保持同步                                  |
+-------------------------------------------------------------------------*/
#define DSM_OBJ_FILE                  0x01 /*对象具有属性信息和数据       */
#define DSM_OBJ_DIRECTORY             0x02 /*对象只有属性信息             */
#define DSM_OBJ_RESERVED1             0x04 /* 以便将来使用                */
#define DSM_OBJ_RESERVED2             0x05 /* 以便将来使用                */
#define DSM_OBJ_RESERVED3             0x06 /* 以便将来使用                */
#define DSM_OBJ_WILDCARD              0xFE /* 任何对象类型                */
#define DSM_OBJ_ANY_TYPE              0xFF /* 以便将来使用                */

/*-------------------------------------------------------------------------+
| QryResp 中 compressedState 的类型定义                                   |
+-------------------------------------------------------------------------*/
#define DSM_OBJ_COMPRESSED_UNKNOWN    0
#define DSM_OBJ_COMPRESSED_YES        1
#define DSM_OBJ_COMPRESSED_NO         2

/*---------------------------------------------------------------------+
| tsmGrouphandlerIn_t 中“组类型”字段的定义                            |
+---------------------------------------------------------------------*/
```

```
#define  DSM_GROUPTYPE_NONE         0x00   /* 不是组成员                    */
#define  DSM_GROUPTYPE_RESERVED1    0x01   /* 以便将来使用                  */
#define  DSM_GROUPTYPE_PEER         0x02   /* 同级组                         */
#define  DSM_GROUPTYPE_RESERVED2    0x03   /* 以便将来使用                  */

/*-------------------------------------------------------------------+
| tsmGrouphandlerIn_t 中“成员类型”字段的定义                          |
+-------------------------------------------------------------------*/

#define  DSM_MEMBERTYPE_LEADER      0x01   /* 组引导符      */
#define  DSM_MEMBERTYPE_MEMBER      0x02   /* 组成员        */

/*-------------------------------------------------------------------+
| tsmGrouphandlerIn_t 中“操作类型”字段定义                            |
+-------------------------------------------------------------------*/
#define DSM_GROUP_ACTION_BEGIN         0x01
#define DSM_GROUP_ACTION_OPEN          0x02 /* 创建新组                    */
#define DSM_GROUP_ACTION_CLOSE         0x03 /* 提交和保存打开的组          */
#define DSM_GROUP_ACTION_ADD           0x04 /* 附加到组       */
#define DSM_GROUP_ACTION_ASSIGNTO      0x05 /* 分配到其他组                */
#define DSM_GROUP_ACTION_REMOVE        0x06 /* 从组中除去成员              */

/*-------------------------------------------------------------------+
| 用于查询管理类响应的 DetailCG 结构中 copySer 的值                   |
+-------------------------------------------------------------------*/
#define Copy_Serial_Static          1   /*复制串行化静态                 */
#define Copy_Serial_Shared_Static   2   /*复制串行化共享静态             */
#define Copy_Serial_Shared_Dynamic  3   /*复制串行化共享动态             */
#define Copy_Serial_Dynamic         4   /*复制串行化动态                 */

/*-------------------------------------------------------------------+
| 用于查询管理类响应的 DetailCG 结构中 copyMode 的值                  |
+-------------------------------------------------------------------*/
#define Copy_Mode_Modified          1   /*复制模式已修改                 */
#define Copy_Mode_Absolute          2   /*绝对复制模式                   */

/*-------------------------------------------------------------------+
|  qryBackupData 结构中 objState 的值                                 |
+-------------------------------------------------------------------*/
#define DSM_ACTIVE            0x01      /* 仅查询活动对象               */
#define DSM_INACTIVE          0x02      /* 仅查询非活动对象             */
#define DSM_ANY_MATCH         0xFF      /* 查询所有备份对象             */

/*-------------------------------------------------------------------+
| qryArchiveData 结构中 dsmDate.year 字段的界限值                     |
+-------------------------------------------------------------------*/
#define DATE_MINUS_INFINITE         0x0000      /* 最低界限            */
#define DATE_PLUS_INFINITE          0xFFFF      /* 最高上限            */

/*-------------------------------------------------------------------+
| 用于更新 dsmUpdateFS() 上的操作参数的位掩码                         |
+-------------------------------------------------------------------*/
#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)

/*-------------------------------------------------------------------+
| 用于备份 dsmUpdateObj() 上的更新操作参数的位掩码                    |
+-------------------------------------------------------------------*/
#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)

/*-------------------------------------------------------------------+
|  dsmDeleteFS() 中的存储库参数的值                                   |
+-------------------------------------------------------------------*/
#define DSM_ARCHIVE_REP       0x0A      /* 归档存储库                 */
#define DSM_BACKUP_REP        0x0B      /* 备份存储库                 */
#define DSM_REPOS_ALL         0x01      /* 所有存储库类型             */
```

```
/*-------------------------------------------------------------------------+
|  dsmEndTxn() 中的表决参数的值                                             |
+-------------------------------------------------------------------------*/
#define DSM_VOTE_COMMIT  1             /* 提交当前事务                     */
#define DSM_VOTE_ABORT   2             /* 回滚当前事务                     */

/*-------------------------------------------------------------------------+
|  ApiSessInfo 结构中返回的各标志的值。              |
+-------------------------------------------------------------------------*/
/* 客户机压缩字段代码        */
#define COMPRESS_YES    1    /* 客户机必须压缩数据                    */
#define COMPRESS_NO     2    /* 客户机决不能压缩数据                  */
#define COMPRESS_CD     3    /* 客户机已确定                           */

/* 归档删除许可权代码。 */
#define ARCHDEL_YES     1    /* 允许归档删除                           */
#define ARCHDEL_NO      2    /* 不允许归档删除                         */

/* 备份删除许可权代码。 */
#define BACKDEL_YES     1    /* 允许备份删除                           */
#define BACKDEL_NO      2    /* 不允许备份删除                         */


/*-------------------------------------------------------------------------+
  optStruct 结构中返回的各标志的值。              |
-------------------------------------------------------------------------*/
#define DSM_PASSWD_GENERATE  1
#define DSM_PASSWD_PROMPT    0

#define DSM_COMM_TCP        1    /* tcpip         */
#define DSM_COMM_NAMEDPIPE  2    /* 命名管道      */
#define DSM_COMM_SHM        3    /* 共享内存      */

/* 旧的     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    /* 用于 IPX/SPX 支持 */
#define DSM_COMM_NETBIOS    12    /* NETBIOS */
#define DSM_COMM_400COMM    12
#define DSM_COMM_CLIO       12    /* CLIO/S */
/*-------------------------------------------------------------------------+
|  dsmInitEx 中供将来使用的 userNameAuthorities 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 中 encryptionType 的值，queryResp                        |
+-------------------------------------------------------------------------*/
#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 字段的定义。                                                  |
+-------------------------------------------------------------------------*/
/*
 *  以下常量定义了介质访问类的层次结构。
 *  数字较小指示媒介可更快速地访问数据。
 */

/* 固定：表示联机、固定媒介的类别（如
          硬盘）。 */
#define  MEDIA_FIXED          0x10

/* 库：表示可通过机器安装设备访问的
           可安装媒介的类。 */
#define  MEDIA_LIBRARY        0x20
```

```
/* 将来使用    */
#define  MEDIA_NETWORK        0x30

/* 将来使用    */
#define  MEDIA_SHELF          0x40

/* 将来使用    */
#define  MEDIA_OFFSITE        0x50

/* 将来使用    */
#define  MEDIA_UNAVAILABLE    0xF0


/*-------------------------------------------------------------------------+
| dsmBeginGetData() 的部分对象数据的类型定义                                |
+-------------------------------------------------------------------------*/
typedef struct
{
    dsUint16_t     stVersion;          /* 结构版本                              */
    dsStruct64_t   partialObjOffset;   /* 开始读取的对象中的偏移量              */
    dsStruct64_t   partialObjLength;   /* 要读取的对象数量                    */
} PartialObjData ;               /* 部分对象数据                             */

#define PartialObjDataVersion 1  /*                                          */

/*-------------------------------------------------------------------------+
|  数据结构的类型定义                                                       |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t    year;                 /* 年，16 位整数（例如，1990）       */
   dsUint8_t     month;                /* 月，8 位整数 (1 - 12)              */
   dsUint8_t     day;                  /* 日，  8 位整数 (1 - 31)            */
   dsUint8_t     hour;                 /* 小时，8 位整数 (0 - 23)            */
   dsUint8_t     minute;               /* 分钟，8 位整数 (0 - 59)            */
   dsUint8_t     second;               /* 秒，b 位整数 (0 - 59)              */
}dsmDate ;

/*-------------------------------------------------------------------------+
|  dsmGetObj() 和 dsmGetList 结构中的对象标识的类型定义                     |
+-------------------------------------------------------------------------*/
typedef dsStruct64_t  ObjID ;

/*-------------------------------------------------------------------------+
| dsmBeginQuery() 中 dsmQueryBuff 的类型定义                                |
+-------------------------------------------------------------------------*/
typedef void dsmQueryBuff ;

/*-------------------------------------------------------------------------+
| dsmBeginGetData() 中 dsmGetType 参数的类型定义                            |
+-------------------------------------------------------------------------*/
typedef enum
{
        gtBackup = 0x00,                    /* 备份处理类型               */
        gtArchive                           /* 归档处理类型               */
} dsmGetType ;

/*-------------------------------------------------------------------------+
|  dsmBeginQuery() 中 dsmQueryType 参数的类型定义                           |
+-------------------------------------------------------------------------*/
typedef enum
{
   qtArchive = 0x00,                 /* 归档查询类型                        */
   qtBackup,                         /* 备份查询类型                        */
   qtBackupActive,                   /* 活动备份文件的快速查询             */
   qtFilespace,                      /* 文件空间查询类型                    */
   qtMC,                             /* 管理类查询类型                      */
   qtReserved1,                      /* 将来使用                            */
   qtReserved2,                      /* 将来使用                            */
   qtReserved3,                      /* 将来使用                            */
   qtReserved4,                      /* 将来使用                            */
   qtBackupGroups,                   /* 特定 fs 中的组引导符               */
   qtOpenGroups,                     /* 特定 fs 中打开的组                  */
   qtReserved5,                      /* 将来使用                            */
   qtProxyNodeAuth,                  /* 其节点可代理的节点                 */
   qtProxyNodePeer,                  /* 具有相同目标的同级节点             */
   qtReserved6,                      /* 将来使用                            */
   qtReserved7,                      /* 将来使用                            */
```

```
   qtReserved8                        /* 将来使用                          */
}dsmQueryType ;

/*--------------------------------------------------------------------------+
|  dsmBindMC() 和 dsmSendObj() 中 sendType 参数的类型定义                   |
+--------------------------------------------------------------------------*/
typedef enum
{
   stBackup = 0x00,                          /* 备份处理类型              */
   stArchive,                                /* 归档处理类型              */
   stBackupMountWait,           /* 在 mountwait 中进行备份处理           */
   stArchiveMountWait           /* 在 mountwait 中进行归档处理           */
}dsmSendType ;

/*--------------------------------------------------------------------------+
|  dsmDeleteObj() 中 delType 参数的类型定义                                 |
+--------------------------------------------------------------------------*/
typedef enum
{
   dtArchive = 0x00,                              /* 归档删除类型         */
   dtBackup,                          /* 备份删除 ( 取消激活 ) 类型       */
   dtBackupID                        /* 备份删除 ( 除去 ) 类型            */
}dsmDelType ;

/*--------------------------------------------------------------------------+
|  dsmSetAccess() 中 sendType 参数的类型定义                                |
+--------------------------------------------------------------------------*/
typedef enum
{
   atBackup = 0x00,                          /* 备份处理类型              */
   atArchive                                /*  归档处理类型              */
}dsmAccessType;

/*--------------------------------------------------------------------------+
|  dsmInit() 和 dsmQueryApiVersion() 中 API 版本的类型定义                  |
+--------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t version;       /* API 版本                          */
   dsUint16_t release;       /* API 发行版                        */
   dsUint16_t level;         /* API 级别                          */
}dsmApiVersion;

/*--------------------------------------------------------------------------+
|  dsmInit() 和 dsmQueryApiVersion() 中 API 版本的类型定义                  |
+--------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t   stVersion;                /* 结构版本                                */
   dsUint16_t version;       /* API 版本                          */
   dsUint16_t release;       /* API 发行版                        */
   dsUint16_t level;         /* API 级别                          */
   dsUint16_t subLevel;      /* API 子级别                        */
   dsmBool_t  unicode;       /* 是否为 API unicode?                       */
}dsmApiVersionEx;


#define apiVersionExVer       2

/*--------------------------------------------------------------------------+
|  dsmInit() 中应用程序版本的类型定义                                       |
+--------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t   stVersion;                /* 结构版本                                */
   dsUint16_t   applicationVersion;   /* 应用程序版本号                */
   dsUint16_t   applicationRelease;   /* 应用程序发行版号              */
   dsUint16_t   applicationLevel;     /* 应用程序级别号                */
   dsUint16_t   applicationSubLevel;  /* 应用程序子级别号              */
} dsmAppVersion;

#define appVersionVer      1

/*--------------------------------------------------------------------------+
|  BindMC、发送、删除、查询时使用的对象名称的类型定义                       |
+--------------------------------------------------------------------------*/
typedef struct S_dsmObjName
{
   char       fs[DSM_MAX_FSNAME_LENGTH + 1] ;              /* 文件空间名称    */
```

```
   char       hl[DSM_MAX_HL_LENGTH + 1] ;                  /* 高级别名称       */
   char       ll[DSM_MAX_LL_LENGTH + 1] ;                   /* 低级别名称      */
   dsUint8_t  objType;          /* 关于对象类型的值，请参阅上面的定义         */
}dsmObjName;


/*-------------------------------------------------------------------------+
|  dsmDeleteObj() 中备份删除信息的类型定义                                  |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t       stVersion ;                    /* 结构版本               */
   dsmObjName       *objNameP ;                    /* 对象名                */
   dsUint32_t       copyGroup ;                    /* 副本组                  */
}delBack ;

#define  delBackVersion    1

/*-------------------------------------------------------------------------+
|  dsmDeleteObj() 中归档删除信息的类型定义                                  |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t       stVersion ;                    /* 结构版本               */
   dsStruct64_t       objId ;                        /* 对象标识               */
}delArch ;

#define  delArchVersion  1

/*-------------------------------------------------------------------------+
|  dsmDeleteObj() 中备份标识删除信息的类型定义                              |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t       stVersion ;                    /* 结构版本               */
   dsStruct64_t       objId ;                        /* 对象标识               */
}delBackID;

#define  delBackIDVersion  1

/*-------------------------------------------------------------------------+
|  dsmDeleteObj() 中删除信息的类型定义                                      |
+-------------------------------------------------------------------------*/
typedef union
{
   delBack   backInfo ;
   delArch   archInfo ;
   delBackID backIDInfo ;
}dsmDelInfo ;

/*-------------------------------------------------------------------------+
|  dsmSendObj() 中对象属性参数的类型定义                                    |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t stVersion;                          /* 结构版本                 */
   char         owner[DSM_MAX_OWNER_LENGTH + 1]; /* 对象所有者               */
   dsStruct64_t sizeEstimate;                     /* 对象的大小估计（以字节表示）      */
   dsmBool_t    objCompressed;                    /* 对象是否已压缩？ */
   dsUint16_t   objInfoLength;                    /* 与对象相关的信息的长度            */
   char         *objInfo;                         /* 与对象相关的信息                  */
   char         *mcNameP;                         /* 要覆盖的管理类名称                */
   dsmBool_t    disableDeduplication;             /* 对此对象不实施重复数据删除    */
   dsmBool_t    useExtObjInfo;                    /* 使用扩展对象信息，最多 1536 */
}ObjAttr;

#define ObjAttrVersion  4



/*-------------------------------------------------------------------------+
| dsmBindMC() 中返回的 mcBindKey 的类型定义                                 |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t  stVersion;                 /* 结构版本            */
   char        mcName[DSM_MAX_MC_NAME_LENGTH + 1];
                                          /* 与对象绑定的 mc 的名称。 */
   dsmBool_t   backup_cg_exists;                          /* True/false */
```

```
   dsmBool_t   archive_cg_exists;                          /* True/false */
   char        backup_copy_dest[DSM_MAX_CG_DEST_LENGTH + 1];
                                                 /* 备份副本目标名称       */
   char        archive_copy_dest[DSM_MAX_CG_DEST_LENGTH + 1];
                                                   /* 归档副本目标名称    */
}mcBindKey;

#define mcBindKeyVersion  1


/*--------------------------------------------------------------------------+
| dsmBeginGetData() 中对象列表的类型定义                                    |
+--------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t       stVersion ;          /* 结构版本                       */
   dsUint32_t       numObjId ;          /* 列表中对象标识数               */
   ObjID           *objId ;             /* 要恢复的对象标识列表           */
   PartialObjData  *partialObjData;     /*部分对象数据信息的列表          */
}dsmGetList ;

#define dsmGetListVersion     2  /* 没有使用部分对象数据时的缺省值        */
#define dsmGetListPORVersion  3  /* 使用部分对象数据的版本                */

/*--------------------------------------------------------------------------+
|  用于获取或发送数据的 DataBlk 的类型定义                                  |
+--------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t stVersion ;                /* 结构版本                       */
   dsUint32_t bufferLen;                 /* 下面传递的缓冲区的长度         */
   dsUint32_t numBytes;                  /* 从中读取的实际字节数           */
                                         /* 或写入缓冲区                   */
   char       *bufferPtr;                /* 数据缓冲区                     */
   dsUint32_t numBytesCompressed;        /* 发送压缩的实际字节数          */
   dsUint16_t reserved;                  /* 以便将来使用                   */
}DataBlk;

#define DataBlkVersion  3


/*--------------------------------------------------------------------------+
|  dsmBeginQuery() 中管理类 queryBuffer 的类型定义                          |
+--------------------------------------------------------------------------*/
typedef struct S_qryMCData
{
   dsUint16_t  stVersion;                 /* 结构版本             */
   char         *mcName;                        /* 管理类名称                */
                        /* 单个名称为获取一个或空字符串为获取全部           */
   dsmBool_t    mcDetail;                       /* 是否想要详细信息？ */
}qryMCData;

#define qryMCDataVersion  1

/*=== RETINIT 的值      ===*/
#define ARCH_RETINIT_CREATE 0
#define ARCH_RETINIT_EVENT  1

/*--------------------------------------------------------------------------+
|  查询 MC 响应中归档副本组详细信息的类型定义                               |
+--------------------------------------------------------------------------*/
typedef struct S_archDetailCG
{
   char         cgName[DSM_MAX_CG_NAME_LENGTH + 1];       /* 副本组名称      */
   dsUint16_t   frequency;                        /* 副本 ( 归档 ) 频率          */
   dsUint16_t   retainVers;                                /* 保留版本       */
   dsUint8_t    copySer;       /* 用于副本序列化值，参阅定义                  */
   dsUint8_t    copyMode;          /* 用于副本模式值，请参阅上面的定义        */
   char         destName[DSM_MAX_CG_DEST_LENGTH + 1];      /* 副本目标名称   */
   dsmBool_t    bLanFreeDest;            /* 目标是否有不依赖 LAN 的路径？   */
   dsmBool_t    reserved;                /* 当前没有使用                      */
   dsUint8_t    retainInit;              /* 请参阅上面的内容获取可能的值     */
   dsUint16_t   retainMin;               /* retInit 为 EVENT 时的天数        */
   dsmBool_t    bDeduplicate;            /* 目标已重复启用                   */
}archDetailCG;

/*--------------------------------------------------------------------------+
|  查询 MC 响应中备份副本组详细信息的类型定义                               |
```

```
+-------------------------------------------------------------------------*/
typedef struct S_backupDetailCG
{
   char         cgName[DSM_MAX_CG_NAME_LENGTH + 1];       /* 副本组名称      */
   dsUint16_t   frequency;                                /* 备份频率         */
   dsUint16_t   verDataExst;                             /* 存在版本数据         */
   dsUint16_t   verDataDltd;                             /* 已删除版本数据        */
   dsUint16_t   retXtraVers;                             /* 保留附加版本          */
   dsUint16_t   retOnlyVers;                             /* 仅保留版本          */
   dsUint8_t    copySer;       /* 用于副本序列化值，参阅定义                    */
   dsUint8_t    copyMode;         /* 用于副本模式值，请参阅上面的定义          */
   char         destName[DSM_MAX_CG_DEST_LENGTH + 1];      /* 副本目标名称    */
   dsBool_t     bLanFreeDest;            /* 目标是否有不依赖 LAN 的路径？    */
   dsBool_t     reserved;                /* 当前没有使用                       */
   dsBool_t     bDeduplicate;            /* 目标已重复启用                     */
}backupDetailCG;


/*-------------------------------------------------------------------------+
|  dsmGetNextQObj() 中查询管理类详细信息响应的类型定义                      |
+-------------------------------------------------------------------------*/
typedef struct S_qryRespMCDetailData
{
   dsUint16_t  stVersion;                 /* 结构版本            */
   char            mcName[DSM_MAX_MC_NAME_LENGTH + 1];        /* mc 名称  */
   char            mcDesc[DSM_MAX_MC_DESCR_LENGTH + 1]; /*mc 描述        */
   archDetailCG    archDet;                 /* 归档副本组详细信息        */
   backupDetailCG  backupDet;                /* 备份副本组详细信息       */
}qryRespMCDetailData;

#define qryRespMCDetailDataVersion  4

/*-------------------------------------------------------------------------+
| dsmGetNextQObj() 中查询管理类摘要响应的类型定义                           |
+-------------------------------------------------------------------------*/
typedef struct S_qryRespMCData
{
   dsUint16_t  stVersion;                 /* 结构版本            */
   char            mcName[DSM_MAX_MC_NAME_LENGTH + 1];        /* mc 名称  */
   char            mcDesc[DSM_MAX_MC_DESCR_LENGTH + 1]; /*mc 描述        */
}qryRespMCData;

#define qryRespMCDataVersion  1



/*-------------------------------------------------------------------------+
|  dsmBeginQuery() 中归档 queryBuffer 的类型定义                            |
+-------------------------------------------------------------------------*/
typedef struct S_qryArchiveData
{
   dsUint16_t  stVersion;                 /* 结构版本            */
   dsmObjName   *objName;                    /* 对象的完整 dsm 名称      */
   char         *owner;                                  /* 所有者名称 */
                  /* 对于最大日期界限，请参阅上面的定义             */
   dsmDate      insDateLowerBound;      /* 归档插入日期下限              */
   dsmDate      insDateUpperBound;       /* 归档插入日期上限             */
   dsmDate      expDateLowerBound;          /* 到期日期下限              */
   dsmDate      expDateUpperBound;           /* 到期日期上限             */
   char         *descr;                /* 归档描述                      */
} qryArchiveData;

#define qryArchiveDataVersion  1



/*=== 用于 retentionInitiated 字段     ===*/
#define DSM_ARCH_RETINIT_UNKNOWN 0 /* 保留已启动未知 ( 下级 srv )            */
#define DSM_ARCH_RETINIT_STARTED 1 /* 保留时钟已启动                          */
#define DSM_ARCH_RETINIT_PENDING 2 /* 保留时钟未启动                          */


/*=== objHeld 的值       ===*/
#define DSM_ARCH_HELD_UNKNOWN 0    /* 未知的保留状态 ( 下级 srv )            */
#define DSM_ARCH_HELD_FALSE   1    /* 对象未处于删除保留状态                 */
#define DSM_ARCH_HELD_TRUE    2    /* 对象处于删除保留状态                    */

/*-------------------------------------------------------------------------+
```

```
| dsmGetNextQObj() 中查询归档响应的类型定义                                    |
+-----------------------------------------------------------------------*/
typedef struct S_qryRespArchiveData
{
   dsUint16_t  stVersion;                /* 结构版本            */
   dsmObjName      objName;                     /* 文件空间名称限定符       */
   dsUint32_t      copyGroup;                        /* 副本组编号         */
   char            mcName[DSM_MAX_MC_NAME_LENGTH + 1];        /* mc 名称     */
   char            owner[DSM_MAX_OWNER_LENGTH + 1];            /* 所有者名称 */
   dsStruct64_t    objId;                                  /* 唯一的副本标识 */
   dsStruct64_t    reserved;                          /* 向后兼容               */
   dsUint8_t       mediaClass;                           /* 媒介访问类          */
   dsmDate         insDate;                         /* 归档插入日期           */
   dsmDate         expDate;                      /* 对象的到期日期             */
   char            descr[DSM_MAX_DESCR_LENGTH + 1];   /* 归档描述             */
   dsUint16_t      objInfolen;             /* 与对象相关的信息的长度         */
   char            reservedObjInfo[DSM_MAX_OBJINFO_LENGTH];  /*与对象相关的信息 */
   dsUint160_t     restoreOrderExt;                          /* 恢复顺序     */
   dsStruct64_t    sizeEstimate;              /* 用户存储的大小估计         */
   dsUint8_t       compressType;                        /* 压缩标志       */
   dsUint8_t       retentionInitiated;   /* 在保留对象事件中等待的对象      */
   dsUint8_t       objHeld;  /*处于保留“状态”的对象，请参阅上面的值       */
   dsUint8_t       encryptionType;                      /* 加密类型          */
   dsmBool_t       clientDeduplicated;    /* API 删除的重复对象*/
   char            objInfo[DSM_MAX_EXT_OBJINFO_LENGTH];  /*与对象相关的信息 */
   char            compressAlg[DSM_MAX_COMPRESSTYPE_LENGTH + 1];  /* 压缩算法名称 */
}qryRespArchiveData;

#define qryRespArchiveDataVersion  7

/*-------------------------------------------------------------------------+
|  dsmSendObj() 中归档 sendBuff 参数的类型定义                              |
+-------------------------------------------------------------------------*/
typedef struct S_sndArchiveData
{
   dsUint16_t  stVersion;                /* 结构版本            */
   char         *descr;                       /* 归档描述                  */
}sndArchiveData;

#define sndArchiveDataVersion  1

/*-------------------------------------------------------------------------+
|  dsmBeginQuery() 中备份 queryBuffer 的类型定义                            |
+-------------------------------------------------------------------------*/
typedef struct S_qryBackupData
{
   dsUint16_t  stVersion;                  /* 结构版本             */
   dsmObjName  *objName;          /* 对象的 dsm 名称           */
   char         *owner;           /* 所有者名称                */
   dsUint8_t   objState;          /* 对象状态选择器          */
   dsmDate     pitDate;           /* 及时恢复的日期值       */
                                  /* 对于可能的值，请参阅上面的定义 */
}qryBackupData;

#define qryBackupDataVersion  2

typedef struct
{
  dsUint8_t    reserved1;
  dsStruct64_t reserved2;
} reservedInfo_t;                 /* 以便将来使用           */

/*-------------------------------------------------------------------------+
| dsmGetNextQObj() 中查询备份响应的类型定义                                 |
+-------------------------------------------------------------------------*/
typedef struct S_qryRespBackupData
{
   dsUint16_t  stVersion;                /* 结构版本            */
   dsmObjName      objName;                      /* 对象的完整 dsm 名称     */
   dsUint32_t      copyGroup;                        /* 副本组编号         */
   char            mcName[DSM_MAX_MC_NAME_LENGTH + 1];        /* mc 名称     */
   char            owner[DSM_MAX_OWNER_LENGTH + 1];            /* 所有者名称 */
   dsStruct64_t    objId;                                  /* 唯一的对象标识   */
   dsStruct64_t    reserved;                          /* 向后兼容               */
   dsUint8_t       mediaClass;                           /* 媒介访问类          */
   dsUint8_t       objState;                        /* 对象状态、活动等       */
   dsmDate         insDate;                           /* 备份插入日期          */
```

```
   dsmDate         expDate;                    /* 对象的到期日期              */
   dsUint16_t      objInfolen;             /* 与对象相关的信息的长度        */
   char            reservedObjInfo[DSM_MAX_OBJINFO_LENGTH];  /*与对象相关的信息 */
   dsUint160_t     restoreOrderExt;                          /* 恢复顺序     */
   dsStruct64_t    sizeEstimate;              /* 用户存储的大小估计         */
   dsStruct64_t    baseObjId;
   dsUint16_t      baseObjInfolen;             /* 与对象相关的信息的长度  */
   dsUint8_t       baseObjInfo[DSM_MAX_OBJINFO_LENGTH];  /* 与基本对象相关的信息 */
   dsUint160_t     baseRestoreOrder;                          /* 恢复顺序       */
   dsUint32_t      fsID;
   dsUint8_t       compressType;
   dsmBool_t       isGroupLeader;
   dsmBool_t       isOpenGroup;
   dsUint8_t       reserved1;               /* 以便将来使用   */
   dsmBool_t       reserved2;               /* 以便将来使用   */
   dsUint16_t      reserved3;               /* 以便将来使用   */
   reservedInfo_t  *reserved4;              /* 以便将来使用   */
   dsUint8_t       encryptionType;         /* 加密类型        */
   dsmBool_t       clientDeduplicated;    /* API 删除的重复对象*/
   char            objInfo[DSM_MAX_EXT_OBJINFO_LENGTH];  /*与对象相关的信息 */
   char            compressAlg[DSM_MAX_COMPRESSTYPE_LENGTH + 1];  /* 压缩算法名称 */
}qryRespBackupData;

#define qryRespBackupDataVersion  8

/*--------------------------------------------------------------------------+
|  dsmBeginQuery() 中活动备份 queryBuffer 的类型定义
|
|  注意：有关活动备份的查询，只需设置 objName 的 fs（文件空间）和 objType
|          字段。   objType 只能设为
|          DSM_OBJ_FILE 或 DSM_OBJ_DIRECTORY。   DSM_OBJ_ANY_TYPE 不会在查询中
|          找到匹配。
+--------------------------------------------------------------------------*/
typedef struct S_qryABackupData
{
   dsUint16_t  stVersion;                /* 结构版本            */
   dsmObjName      *objName;                    /* 仅使用 fs 和 objtype     */
}qryABackupData;

#define qryABackupDataVersion  1

/*--------------------------------------------------------------------------+
| dsmGetNextQObj() 中查询活动备份响应的类型定义                              |
+--------------------------------------------------------------------------*/
typedef struct S_qryARespBackupData
{
   dsUint16_t  stVersion;                /* 结构版本            */
   dsmObjName      objName;                  /* 对象的完整 dsm 名称     */
   dsUint32_t      copyGroup;                /* 副本组编号              */
   char        mcName[DSM_MAX_MC_NAME_LENGTH + 1];/管理类名称            */
   char            owner[DSM_MAX_OWNER_LENGTH + 1];        /* 所有者名称 */
   dsmDate         insDate;                  /* 备份插入日期         */
   dsUint16_t      objInfolen;         /* 与对象相关的信息的长度      */
   char        reservedObjInfo[DSM_MAX_OBJINFO_LENGTH];  /*与对象相关的信息 */
   char        objInfo[DSM_MAX_EXT_OBJINFO_LENGTH];  /*与对象相关的信息 */
}qryARespBackupData;

#define qryARespBackupDataVersion  2

/*--------------------------------------------------------------------------+
|  dsmBeginQuery() 中备份 queryBuffer 的类型定义                            |
+--------------------------------------------------------------------------*/
typedef struct qryBackupGroups
{
   dsUint16_t  stVersion;                /* 结构版本            */
   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
```

```
/*--------------------------------------------------------------------------+
|  dsmBeginQuery() 中代理节点 queryBuffer 的类型定义                        |
+--------------------------------------------------------------------------*/
typedef struct qryProxyNodeData
{
   dsUint16_t  stVersion;                /* 结构版本            */
   char        *targetNodeName;            /* 目标节点名称      */
}qryProxyNodeData;

#define qryProxyNodeDataVersion  1

/*--------------------------------------------------------------------------+
| dsmGetNextQObj() 中使用的 qryRespProxyNodeData 参数的类型定义             |
+--------------------------------------------------------------------------*/

typedef struct
{
   dsUint16_t stVersion ;                                 /* 结构版本          */
   char            targetNodeName[DSM_MAX_ID_LENGTH+1]; /* 目标节点名称      */
   char            peerNodeName[DSM_MAX_ID_LENGTH+1];   /* 同级节点名称      */
   char            hlAddress[DSM_MAX_ID_LENGTH+1];      /* 同级 hlAddress    */
   char            llAddress[DSM_MAX_ID_LENGTH+1];      /* 同级 hlAddress    */
}qryRespProxyNodeData;

#define qryRespProxyNodeDataVersion  1


/*--------------------------------------------------------------------------+
|  WINNT 和 OS/2 文件空间属性的类型定义                                     |
+--------------------------------------------------------------------------*/
typedef struct
{
   char         driveLetter ;          /* 文件空间的盘符                */
   dsUint16_t   fsInfoLength;          /* 使用的 fsInfo 长度             */
   char         fsInfo[DSM_MAX_FSINFO_LENGTH];/*调用者确定的数据       */
}dsmDosFSAttrib ;

/*--------------------------------------------------------------------------+
|  UNIX 文件空间属性的类型定义                                              |
+--------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t   fsInfoLength;          /* 使用的 fsInfo 长度             */
   char         fsInfo[DSM_MAX_FSINFO_LENGTH];/*调用者确定的数据       */
}dsmUnixFSAttrib ;

/*--------------------------------------------------------------------------+
|  NetWare 文件空间属性的类型定义                                           |
+--------------------------------------------------------------------------*/
typedef dsmUnixFSAttrib dsmNetwareFSAttrib;

/*--------------------------------------------------------------------------+
|  所有文件空间调用上文件空间属性的类型定义                                 |
+--------------------------------------------------------------------------*/
typedef union
{
   dsmNetwareFSAttrib  netwareFSAttr;
   dsmUnixFSAttrib     unixFSAttr ;
   dsmDosFSAttrib      dosFSAttr ;
}dsmFSAttr ;

/*--------------------------------------------------------------------------+
|  dsmUpdateFS() 上 fsUpd 参数的类型定义
+--------------------------------------------------------------------------*/
typedef struct S_dsmFSUpd
{
   dsUint16_t stVersion ;                /* 结构版本                      */
   char            *fsType ;            /* 文件空间类型                  */
   dsStruct64_t    occupancy ;          /* 占用率估计                    */
   dsStruct64_t    capacity  ;          /* 容量估计                      */
   dsmFSAttr       fsAttr ;             /* 特定平台的属性                */
}dsmFSUpd ;

#define dsmFSUpdVersion  1

/*--------------------------------------------------------------------------+
|  dsmBeginQuery() 中文件空间 queryBuffer 的类型定义                        |
+--------------------------------------------------------------------------*/
typedef struct S_qryFSData
```

```
{
   dsUint16_t  stVersion;                 /* 结构版本            */
   char        *fsName;                   /* 文件空间名称     */
}qryFSData;

#define qryFSDataVersion  1

/*-------------------------------------------------------------------------+
| dsmGetNextQObj() 中查询文件空间响应的类型定义                             |
+-------------------------------------------------------------------------*/
typedef struct S_qryRespFSData
{
   dsUint16_t  stVersion;                 /* 结构版本            */
   char           fsName[DSM_MAX_FSNAME_LENGTH + 1];         /* 文件空间名称    */
   char           fsType[DSM_MAX_FSTYPE_LENGTH + 1] ;        /* 文件空间类型    */
   dsStruct64_t   occupancy;                        /* 占用率估计 ( 以字节表示 )。*/
   dsStruct64_t   capacity;                         /* 容量估计 ( 以字节表示 )。 */
   dsmFSAttr       fsAttr ;                  /* 特定平台的属性                  */
   dsmDate        backStartDate;             /* 开始备份日期                    */
   dsmDate        backCompleteDate;          /* 结束备份日期                    */
   dsmDate        reserved1;                  /* 以便将来使用                   */
   dsmDate        lastReplStartDate;        /* 最近一次启动复制的时间               */
   dsmDate        lastReplCmpltDate;        /* 最近一次完成复制的时间               */
                                            /*    ( 可能已发生故障，                */
                                            /*     但是仍完成 )                     */
   dsmDate        lastBackOpDateFromServer; /* 上次客户机在服务器上进行保存的       */
                                            /*     存储时间戳记                     */
   dsmDate        lastArchOpDateFromServer; /* 上次客户机在服务器上进行保存的       */
                                            /*     存储时间戳记                     */
   dsmDate        lastSpMgOpDateFromServer; /* 上次客户机在服务器上进行保存的       */
                                            /*     存储时间戳记                     */
   dsmDate        lastBackOpDateFromLocal;  /* 上次客户机在本地进行保存的           */
                                            /*     存储时间戳记                     */
   dsmDate        lastArchOpDateFromLocal;  /* 上次客户机在本地进行保存的           */
                                            /*     存储时间戳记                     */
   dsmDate        lastSpMgOpDateFromLocal;  /* 上次客户机在本地进行保存的           */
                                            /*     存储时间戳记                     */
   dsInt32_t      failOverWriteDelay;       /* 在允许存储到此复制服务器之前              */
                                            /* 客户机等待的分钟数，特殊代码：            */
                                            /* NO_ACCESS(-1)，ACCESS_RDONLY (-2)          */
}qryRespFSData;

#define qryRespFSDataVersion 4

/*-------------------------------------------------------------------------+
|  dsmRegisterFS() 中 regFilespace 参数的类型定义
+-------------------------------------------------------------------------*/
typedef struct S_regFSData
{
   dsUint16_t  stVersion;                 /* 结构版本            */
   char            *fsName;                        /* 文件空间名称     */
   char            *fsType;                        /* 文件空间类型     */
   dsStruct64_t   occupancy;                       /* 占用率估计 ( 以字节表示 )。 */
   dsStruct64_t   capacity;                         /* 容量估计 ( 以字节表示 )。 */
   dsmFSAttr       fsAttr ;                    /* 特定平台的属性                  */
}regFSData;

#define regFSDataVersion  1

/*-------------------------------------------------------------------------+
| apisessInfo 中使用的 dedupType 的类型定义                                 |
+-------------------------------------------------------------------------*/
typedef enum
{
   dedupServerOnly= 0x00,              /* 仅在服务器上执行重复数据删除          */
   dedupClientOrServer               /* 仅在客户机或服务器上执行重复数据删除   */
}dsmDedupType ;

/*-------------------------------------------------------------------------+
 | 故障转移配置和状态的类型定义
 -------------------------------------------------------------------------*/
typedef enum
{
   failOvrNotConfigured = 0x00,
   failOvrConfigured,
   failOvrConnectedToReplServer
}dsmFailOvrCfgType ;
```

```
/*-------------------------------------------------------------------------+
| dsmQuerySessionInfo() 中会话信息响应的类型定义                            |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t stVersion;                    /* 结构版本                        */
      /*-----------------------------------------------------------------*/
      /*           服务器信息                                            */
      /*-----------------------------------------------------------------*/
   char           serverHost[DSM_MAX_SERVERNAME_LENGTH+1];
                                  /* DSM 服务器的网络主机     */
   dsUint16_t     serverPort;          /* 主机上服务器的常用端口           */
   dsmDate        serverDate;          /* 服务器的日期/时间                */
   char           serverType[DSM_MAX_SERVERTYPE_LENGTH+1];
                                 /* 服务器的执行平台                       */
   dsUint16_t     serverVer;           /* 服务器的版本号                   */
   dsUint16_t     serverRel;           /* 服务器的发行号                   */
   dsUint16_t     serverLev;           /* 服务器的级别号                   */
   dsUint16_t     serverSubLev;        /* 服务器的子级别号                 */
      /*-----------------------------------------------------------------*/
      /*           客户机缺省值                                          */
      /*-----------------------------------------------------------------*/
   char           nodeType[DSM_MAX_PLATFORM_LENGTH+1]; /*节点/应用程序类型    */
   char           fsdelim;             /* 文件空间分隔符                   */
   char           hldelim;             /* 高低级别间的分隔符           */
   dsUint8_t      compression;         /* 压缩标志                         */
   dsUint8_t      archDel;             /* 归档删除权限                     */
   dsUint8_t      backDel;             /* 备份删除权限                     */
   dsUint32_t     maxBytesPerTxn;      /* 以便将来使用                     */
   dsUint16_t     maxObjPerTxn;        /* txn 中允许的最大对象数           */
      /*-----------------------------------------------------------------*/
      /*           会话信息                                              */
      /*-----------------------------------------------------------------*/
   char       id[DSM_MAX_ID_LENGTH+1];    /* 登录标识节点名称             */
   char       owner[DSM_MAX_OWNER_LENGTH+1]; /* 登陆所有者                */
                                 /*  ( 用于多用户平台 )                   */
   char       confFile[DSM_PATH_MAX + DSM_NAME_MAX +1];
                                 /* len 是平台 dep                        */
                                 /* dsInit name of appl config file  */
   dsUint8_t  opNoTrace;            /* dsInit option - NoTrace = 1       */
      /*-----------------------------------------------------------------*/
      /*           策略数据                                              */
      /*-----------------------------------------------------------------*/
   char           domainName[DSM_MAX_DOMAIN_LENGTH+1]; /* 域名             */
   char           policySetName[DSM_MAX_PS_NAME_LENGTH+1];
                                  /* 活动策略集名称                     */
   dsmDate        polActDate;          /* 策略集激活日期                */
   char           dfltMCName[DSM_MAX_MC_NAME_LENGTH+1];/* 缺省管理类      */
   dsUint16_t     gpBackRetn;          /* Grace-period 备份保留         */
   dsUint16_t     gpArchRetn;          /* Grace-period 归档保留         */
   char           adsmServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* adsm 服务器名称  */
   dsmBool_t      archiveRetentionProtection; /* 启用服务器保留保护              */
   dsStruct64_t   maxBytesPerTxn_64;         /* 以便将来使用                  */
   dsmBool_t      lanFreeEnabled;            /* 已设置不依赖 LAN 的选项         */
   dsmDedupType   dedupType;                 /* 服务器或 clientOrServer         */
   char           accessNode[DSM_MAX_ID_LENGTH+1];    /* 作为节点名称          */

   /*-----------------------------------------------------------------*/
   /*           复制和故障转移信息                                    */
   /*-----------------------------------------------------------------*/
   dsmFailOvrCfgType failOverCfgType; /* 故障转移的状态 */
   char           replServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* 复制服务器名称 */
   char           homeServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* 主服务器名称   */
   char           replServerHost[DSM_MAX_SERVERNAME_LENGTH+1]; /* DSM 服务器的网络主机名  */
   dsInt32_t      replServerPort;                              /* 主机上服务器的常用端口  */

}ApiSessInfo;

#define ApiSessInfoVersion  6



/*-------------------------------------------------------------------------+
| dsmQueryCliOptions() 和 dsmQuerySessOptions() 中查询选项                 |
|      响应的类型定义                                                     |
+-------------------------------------------------------------------------*/
```

```
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 ;

/*------------------------------------------------------------------------+
| logInfo 中使用的 LogType 的类型定义                                      |
+------------------------------------------------------------------------*/
typedef enum
{
   logServer = 0x00,             /* 仅用于服务器的日志消息           */
   logLocal,                     /* 仅用于本地错误日志的日志消息     */
   logBoth,                       /* 将消息记录到服务器和本地错误日志 */
   logNone
}dsmLogType ;

/*------------------------------------------------------------------------+
| dsmLogEvent() 中使用的 logInfo 参数的类型定义                            |
+------------------------------------------------------------------------*/

typedef struct
{
   char        *message;   /* 要记录的消息文本               */
   dsmLogType  logType;    /* 记录类型：本地和/或服务器     */
}logInfo;

/*------------------------------------------------------------------------+
| dsmQueryAccess() 中使用的 qryRespAccessData 参数的类型定义               |
+------------------------------------------------------------------------*/

typedef struct
{
   dsUint16_t stVersion ;                          /* 结构版本               */
   char              node[DSM_MAX_ID_LENGTH+1];     /* 节点名称               */
   char              owner[DSM_MAX_OWNER_LENGTH+1]; /* 所有者                 */
   dsmObjName        objName ;                      /* 对象名                */
   dsmAccessType     accessType;                    /* 归档或备份             */
   dsUint32_t        ruleNumber ;                   /* 访问规则标识           */
}qryRespAccessData;

#define qryRespAccessDataVersion  1

/*------------------------------------------------------------------------+
|  dsmSetUp() 中 envSetUp 参数的类型定义
+------------------------------------------------------------------------*/
typedef struct S_envSetUp
{
   dsUint16_t  stVersion;                 /* 结构版本            */
   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;  /* for executables name argv[0] */
   char            logName[DSM_NAME_MAX +1];
   dsmBool_t       reserved1;         /* 以便将来使用   */
   dsmBool_t       reserved2;                 /* 以便将来使用  */
}envSetUp;

#define envSetUpVersion  4

/*------------------------------------------------------------------------+
|  dsmInitExIn_t 的类型定义
+------------------------------------------------------------------------*/
typedef struct dsmInitExIn_t
{
   dsUint16_t  stVersion;                 /* 结构版本            */
   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

/*--------------------------------------------------------------------------+
|  dsmInitExOut_t 的类型定义
+--------------------------------------------------------------------------*/
typedef struct dsmInitExOut_t
{
   dsUint16_t  stVersion;                 /* 结构版本            */
   dsInt16_t          userNameAuthorities;
   dsInt16_t          infoRC;        /* 如果遇到，将有错误返回码         */
   char               adsmServerName[DSM_MAX_SERVERNAME_LENGTH+1];
   dsUint16_t         serverVer;      /* 服务器的版本号                  */
   dsUint16_t         serverRel;     /* 服务器的发行号                  */
   dsUint16_t         serverLev;     /* 服务器的级别号                  */
   dsUint16_t         serverSubLev;  /* 服务器的子级别号                */

   dsmBool_t          bIsFailOverMode; /* true，如果已发生故障转移 */
   char               replServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* 复制服务器名称 */
   char               homeServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* 主服务器名称 */

}dsmInitExOut_t;

#define dsmInitExOutVersion 3

/*--------------------------------------------------------------------------+
| logInfo 中使用的 LogType 的类型定义                                       |
+--------------------------------------------------------------------------*/
typedef enum
{
   logSevInfo = 0x00,       /* 信息         ANE4991 */
   logSevWarning,           /* 警告         ANE4992 */
   logSevError,             /* 错误         ANE4993 */
   logSevSevere,            /* 严重         ANE4994 */
   logSevLicense,           /* 许可证       ANE4995 */
   logSevTryBuy             /* 试买         ANE4996 */
}dsmLogSeverity ;


/*--------------------------------------------------------------------------+
|  dsmLogExIn_t 的类型定义
+--------------------------------------------------------------------------*/
typedef struct dsmLogExIn_t
{
   dsUint16_t  stVersion;                 /* 结构版本            */
   dsmLogSeverity     severity;
   char               appMsgID[8];
   dsmLogType  logType;    /* 记录类型：本地和/或服务器     */
   char        *message;   /* 要记录的消息文本               */
   char               appName[DSM_MAX_PLATFORM_LENGTH];
   char               osPlatform[DSM_MAX_PLATFORM_LENGTH];
   char               appVersion[DSM_MAX_PLATFORM_LENGTH];
}dsmLogExIn_t;

#define dsmLogExInVersion 2

/*--------------------------------------------------------------------------+
|  dsmlogExOut_t 的类型定义
+--------------------------------------------------------------------------*/
typedef struct dsmLogExOut_t
{
   dsUint16_t  stVersion;                 /* 结构版本            */
}dsmLogExOut_t;

#define dsmLogExOutVersion 1

/*--------------------------------------------------------------------------+
```

```
|  dsmRenameIn_t 的类型定义
+------------------------------------------------------------------------*/
typedef struct dsmRenameIn_t
{
   dsUint16_t  stVersion;                 /* 结构版本            */
   dsUint32_t       dsmHandle;                    /* 会话句柄       */
   dsUint8_t        repository;                   /* 备份或归档     */
   dsmObjName       *objNameP ;                   /* 对象名       */
   char             newHl[DSM_MAX_HL_LENGTH + 1];  /* 新的高级别名称*/
   char             newLl[DSM_MAX_LL_LENGTH + 1];  /* 新的低级别名称*/
   dsmBool_t        merge;                       /* 合并到现有名称中*/
   ObjID            objId;                        /* 归档的对象标识 */
}dsmRenameIn_t;

#define dsmRenameInVersion 1

/*------------------------------------------------------------------------+
|  dsmRenameOut_t 的类型定义
+------------------------------------------------------------------------*/
typedef struct dsmRenameOut_t
{
   dsUint16_t  stVersion;                 /* 结构版本            */
}dsmRenameOut_t;

#define dsmRenameOutVersion 1

/*------------------------------------------------------------------------+
|  dsmEndSendObjExIn_t 的类型定义
+------------------------------------------------------------------------*/
typedef struct dsmEndSendObjExIn_t
{
   dsUint16_t  stVersion;                 /* 结构版本            */
   dsUint32_t       dsmHandle;                    /* 会话句柄       */
}dsmEndSendObjExIn_t;

#define dsmEndSendObjExInVersion 1

/*------------------------------------------------------------------------+
|  dsmEndSendObjExOut_t 的类型定义
+------------------------------------------------------------------------*/
typedef struct dsmEndSendObjExOut_t
{
   dsUint16_t  stVersion;                 /* 结构版本            */
   dsStruct64_t       totalBytesSent;      /* 从应用程序读取的总字节数  */
   dsmBool_t          objCompressed;       /* 压缩对象                  */
   dsStruct64_t       totalCompressSize;   /* 压缩后的总大小            */
   dsStruct64_t       totalLFBytesSent;    /* 发送不依赖 LAN 的总字节数 */
   dsUint8_t          encryptionType;      /* 使用的加密类型            */
   dsmBool_t          objDeduplicated;     /* 为分发数据重复删除而处理的对象 */
   dsStruct64_t       totalDedupSize;      /* 删除重复数据或的总大小       */
}dsmEndSendObjExOut_t;

#define dsmEndSendObjExOutVersion 3
/*------------------------------------------------------------------------+
|  dsmGroupHandlerIn_t 的类型定义
+------------------------------------------------------------------------*/
typedef struct dsmGroupHandlerIn_t
{
   dsUint16_t  stVersion;                 /* 结构版本            */
   dsUint32_t       dsmHandle;        /* 会话句柄                          */
   dsUint8_t       groupType;        /* 组的类型                          */
   dsUint8_t       actionType;       /* 组操作的类型                      */
   dsUint8_t       memberType;       /* 成员类型：领导或成员              */
   dsStruct64_t    leaderObjId;      /* 管理成员时组领导的 OBJID          */
   char            *uniqueGroupTagP; /* 唯一的组标识                      */
   dsmObjName      *objNameP ;       /* 组领导对象名称                    */
   dsmGetList      memberObjList;    /* 要除去、分配的对象列表           */
}dsmGroupHandlerIn_t;

#define dsmGroupHandlerInVersion 1

/*------------------------------------------------------------------------+
|  dsmGroupHandlerExOut_t 的类型定义
+------------------------------------------------------------------------*/
typedef struct dsmGroupHandlerOut_t
{
   dsUint16_t  stVersion;                 /* 结构版本            */
}dsmGroupHandlerOut_t;
```

```
#define dsmGroupHandlerOutVersion 1

/*--------------------------------------------------------------------------+
|  dsmEndTxnExIn_t 的类型定义
+--------------------------------------------------------------------------*/
typedef struct dsmEndTxnExIn_t
{
   dsUint16_t  stVersion;                /* 结构版本             */
   dsUint32_t       dsmHandle;                    /* 会话句柄      */
   dsUint8_t        vote;
}dsmEndTxnExIn_t;

#define dsmEndTxnExInVersion 1

/*--------------------------------------------------------------------------+
|   dsmEndTxnExOut_t 的类型定义
+--------------------------------------------------------------------------*/
typedef struct dsmEndTxnExOut_t
{
   dsUint16_t  stVersion;                /* 结构版本             */
   dsUint16_t         reason;                 /* 原因码                          */
   dsStruct64_t       groupLeaderObjId;       /* 返回的组领导的对象标识          */
                                              /* DSM_ACTION_OPEN                 */
   dsUint8_t          reserved1;              /* 将来使用                        */
   dsUint16_t         reserved2;              /* 将来使用                        */
}dsmEndTxnExOut_t;

#define dsmEndTxnExOutVersion 1

/*--------------------------------------------------------------------------+
|  dsmEndGetDataExIn_t 的类型定义
+--------------------------------------------------------------------------*/
typedef struct dsmEndGetDataExIn_t
{
   dsUint16_t  stVersion;                /* 结构版本             */
   dsUint32_t       dsmHandle;                    /* 会话句柄      */
}dsmEndGetDataExIn_t;

#define dsmEndGetDataExInVersion 1

/*--------------------------------------------------------------------------+
|  dsmEndGetDataExOut_t 的类型定义
+--------------------------------------------------------------------------*/
typedef struct dsmEndGetDataExOut_t
{
   dsUint16_t  stVersion;                /* 结构版本             */
   dsUint16_t         reason;                 /* 原因码                          */
   dsStruct64_t   totalLFBytesRecv;          /* 接收到的不依赖 LAN 的总字节数   */
}dsmEndGetDataExOut_t;

#define dsmEndGetDataExOutVersion 1

/*--------------------------------------------------------------------------+
| dsmRetentionEvent() 中对象列表的类型定义                                  |
+--------------------------------------------------------------------------*/
typedef struct dsmObjList
{
   dsUint16_t  stVersion;                /* 结构版本             */
   dsUint32_t       numObjId ;          /* 列表中对象标识数               */
   ObjID            *objId;            /* 要以信号通知的对象标识的列表    */
}dsmObjList_t ;

#define dsmObjlistVersion 1

/*--------------------------------------------------------------------------+
| dsmRetentionEvent 中使用的 eventType 的类型定义                           |
+--------------------------------------------------------------------------*/
typedef enum
{
   eventRetentionActivate = 0x00,   /* 发送信号以告知服务器已发生事件              */
   eventHoldObj,                    /* 暂挂对象的删除/到期                          */
   eventReleaseObj                  /* 恢复正常的删除/到期处理                      */
}dsmEventType_t;

/*--------------------------------------------------------------------------+
|  dsmRetentionEvent() 的类型定义                                           |
+--------------------------------------------------------------------------*/
typedef struct dsmRetentionEventIn_t
```

```
{
   dsUint16_t  stVersion;                /* 结构版本            */
   dsUint32_t         dsmHandle;                    /* 会话句柄           */
   dsmEventType_t     eventType;                    /* 事件类型            */
   dsmObjList_t       objList;                      /* 对象标识            */
}dsmRetentionEventIn_t;

#define dsmRetentionEventInVersion 1

/*--------------------------------------------------------------------------+
|  dsmRetentionEvent() 的类型定义                        |
+--------------------------------------------------------------------------*/
typedef struct dsmRetentionEventOut_t
{
   dsUint16_t       stVersion ;                   /* 结构版本               */
}dsmRetentionEventOut_t;

#define dsmRetentionEventOutVersion 1


/*--------------------------------------------------------------------------+
| dsmRequestBuffer() 的类型定义                                             |
+--------------------------------------------------------------------------*/
typedef struct requestBufferIn_t
{
   dsUint16_t  stVersion;                /* 结构版本            */
   dsUint32_t         dsmHandle;                    /* 会话句柄           */
}requestBufferIn_t;

#define requestBufferInVersion 1

/*--------------------------------------------------------------------------+
| dsmRequestBuffer() 的类型定义                                             |
+--------------------------------------------------------------------------*/
typedef struct requestBufferOut_t
{
   dsUint16_t       stVersion ;                /* 结构版本                */
   dsUint8_t        tsmBufferHandle;           /* tsm 数据缓冲区的句柄      */
   char             *dataPtr;                  /* 要写入数据的地址           */
   dsUint32_t       bufferLen;                 /* 要写入的数据最大长度      */
}requestBufferOut_t;

#define requestBufferOutVersion 1

/*--------------------------------------------------------------------------+
|  dsmReleaseBuffer() 的类型定义                                            |
+--------------------------------------------------------------------------*/
typedef struct releaseBufferIn_t
{
   dsUint16_t  stVersion;                /* 结构版本            */
   dsUint32_t         dsmHandle;                    /* 会话句柄           */
   dsUint8_t        tsmBufferHandle;                /* tsm 数据缓冲区的句柄*/
   char             *dataPtr;                       /* 要写入数据的地址    */
}releaseBufferIn_t;

#define releaseBufferInVersion 1

/*--------------------------------------------------------------------------+
|  dsmReleaseBuffer() 的类型定义                                            |
+--------------------------------------------------------------------------*/
typedef struct releaseBufferOut_t
{
   dsUint16_t       stVersion ;                /* 结构版本                */
}releaseBufferOut_t;

#define releaseBufferOutVersion 1

/*--------------------------------------------------------------------------+
|  dsmGetBufferData() 的类型定义                                             |
+--------------------------------------------------------------------------*/
typedef struct getBufferDataIn_t
{
   dsUint16_t  stVersion;                /* 结构版本            */
   dsUint32_t         dsmHandle;                    /* 会话句柄           */
}getBufferDataIn_t;

#define getBufferDataInVersion 1

/*--------------------------------------------------------------------------+
```

```
|  dsmGetBufferData() 的类型定义                                                |
+-----------------------------------------------------------------------------*/
typedef struct getBufferDataOut_t
{
   dsUint16_t       stVersion ;                 /* 结构版本                  */
   dsUint8_t        tsmBufferHandle;           /* tsm 数据缓冲区的句柄*/
   char             *dataPtr;                 /* 要读取数据的实际地址      */
   dsUint32_t       numBytes;              /* 从 dataPtr 读取的实际字节数 */
}getBufferDataOut_t;

#define getBufferDataOutVersion 1

/*-----------------------------------------------------------------------------+
|  dsmSendBufferData() 的类型定义                                               |
+-----------------------------------------------------------------------------*/
typedef struct sendBufferDataIn_t
{
   dsUint16_t  stVersion;                 /* 结构版本             */
   dsUint32_t         dsmHandle;                    /* 会话句柄           */
   dsUint8_t        tsmBufferHandle;               /* tsm 数据缓冲区的句柄*/
   char               *dataPtr;                   /* 要发送数据的实际数据地址 */
   dsUint32_t         numBytes;                   /* 从 dataPtr* 发送的实际字节数*/
}sendBufferDataIn_t;

#define sendBufferDataInVersion 1

/*-----------------------------------------------------------------------------+
|  dsmSendBufferData() 的类型定义                                               |
+-----------------------------------------------------------------------------*/
typedef struct sendBufferDataOut_t
{
   dsUint16_t       stVersion ;                 /* 结构版本                  */
}sendBufferDataOut_t;

#define sendBufferDataOutVersion 1

/*-----------------------------------------------------------------------------+
|  dsmUpdateObjExIn_t 的类型定义
+-----------------------------------------------------------------------------*/
typedef struct dsmUpdateObjExIn_t
{
   dsUint16_t  stVersion;                 /* 结构版本             */
   dsUint32_t         dsmHandle;                    /* 会话句柄           */
   dsmSendType     sendType;                       /* 发送类型备份/归档    */
   char            *descrP;                       /* 归档描述             */
   dsmObjName      *objNameP;                    /* 对象名               */
   ObjAttr         *objAttrPtr;                  /* 属性                  */
   dsUint32_t      objUpdAct;           /* 更新操作             */
   ObjID           archObjId;          /* 归档的对象标识       */
}dsmUpdateObjExIn_t;

#define dsmUpdateObjExInVersion 1

/*-----------------------------------------------------------------------------+
|  dsmUpdateObjExOut_t 的类型定义
+-----------------------------------------------------------------------------*/
typedef struct dsmUpdateObjExOut_t
{
    dsUint16_t  stVersion;                 /* 结构版本             */
}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                    *
*****************************************************************************/

/*****************************************************************************
 * 头文件名称：tsmapitd.h
 *
 * 环境：       ***********************************************
 *                    ** 这是与平台无关的源文件     **
 *
 *                    ***********************************************
 *
 * 设计说明：         本文件包含可在所有客户机源文件中包括的
 *                    基本数据类型和常量。 此文件中的常量
 *                    应正确设置，以便适用于
 *                    要运行客户机软件的
 *                    特殊机器和操作系统。
 *
 *                    平台特定的定义包括在 dsmapips.h 中
 *
 * 描述性名称：IBM Spectrum Protect API 常量的定义
 *---------------------------------------------------------------------------*/

#ifndef _H_TSMAPITD
#define _H_TSMAPITD


/*=== 设置结构对齐以包装结构 ===*/
#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 界面的 Win32 应用程序在编译期间必须使用
 -DUNICODE 标志。
 ============================================================*/
#if _OPSYS_TYPE == DS_WINNT && !defined(DSMAPILIB)
#ifndef UNICODE
#error "使用 TSM 界面的 Win32 应用程序编译时，必须带有 -DUNICODE 标志"
#endif
#endif

/*============================================================
 使用 TSM 界面的 Mac OS X 应用程序在编译期间必须使用
 -DUNICODE 标志。
 ============================================================*/
#if _OPSYS_TYPE == DS_MACOS && !defined(DSMAPILIB)
#ifndef UNICODE
#error "使用 TSM 界面的 Mac OS X 应用程序编译时，必须带有 -DUNICODE 标志"
#endif
#endif

/*---------------------------------------------------------------------------+
 | tsmBeginGetData() 中 dsmGetType 参数的类型定义                            |
 +---------------------------------------------------------------------------*/
typedef enum
{
   gtTsmBackup = 0x00,                    /* 备份处理类型              */
   gtTsmArchive                           /* 归档处理类型             */
} tsmGetType ;
```

```
/*-------------------------------------------------------------------------+
 |  tsmBeginQuery() 中 dsmQueryType 参数的类型定义                          |
 +-------------------------------------------------------------------------*/
typedef enum
{
   qtTsmArchive = 0x00,                    /* 归档查询类型                */
   qtTsmBackup,                            /* 备份查询类型                */
   qtTsmBackupActive,                      /* 活动备份文件的快速查询     */
   qtTsmFilespace,                         /* 文件空间查询类型           */
   qtTsmMC,                                /* 管理类查询类型              */
    qtTsmReserved1,                         /* 将来使用                   */
    qtTsmReserved2,                         /* 将来使用                   */
    qtTsmReserved3,                         /* 将来使用                   */
    qtTsmReserved4,                         /* 将来使用                   */
   qtTsmBackupGroups,                      /* 特定文件空间中所有组领导   */
   qtTsmOpenGroups,                        /* 与领导相关联的所有组成员   */
    qtTsmReserved5,                         /* 将来使用                   */
   qtTsmProxyNodeAuth,                     /* 此节点可代理的节点         */
    qtTsmProxyNodePeer,                     /* 此目标节点下的同级节点     */
    qtTsmReserved6,                         /* 将来使用                   */
    qtTsmReserved7,                         /* 未来使用                   */
    qtTsmReserved8                          /* 将来使用                   */
} tsmQueryType ;

/*-------------------------------------------------------------------------+
 |  tsmBindMC() 和 tsmSendObj() 中 sendType 参数的类型定义                 |
 +-------------------------------------------------------------------------*/
typedef enum
{
   stTsmBackup = 0x00,                        /* 备份处理类型             */
   stTsmArchive,                              /* 归档处理类型             */
   stTsmBackupMountWait,           /* 在 mountwait 中进行备份处理         */
   stTsmArchiveMountWait           /* 在 mountwait 中进行归档处理         */
} tsmSendType ;

/*-------------------------------------------------------------------------+
 |  tsmDeleteObj() 中 delType 参数的类型定义                               |
 +-------------------------------------------------------------------------*/
typedef enum
{
   dtTsmArchive = 0x00,                             /* 归档删除类型       */
   dtTsmBackup,                          /* 备份删除 ( 取消激活 ) 类型    */
   dtTsmBackupID                        /* 备份删除 ( 除去 ) 类型         */
} tsmDelType ;

/*-------------------------------------------------------------------------+
 |  tsmSetAccess() 中 sendType 参数的类型                                  |
 +-------------------------------------------------------------------------*/
typedef enum
{
   atTsmBackup = 0x00,                        /* 备份处理类型             */
   atTsmArchive                              /* 归档处理类型              */
}tsmAccessType;

/*-------------------------------------------------------------------------+
 |  tsmSendObj() 中覆盖参数的类型定义                                      |
 +-------------------------------------------------------------------------*/
typedef enum
{
   owIGNORE = 0x00,
   owYES,
   owNO
}tsmOwType;


/*-------------------------------------------------------------------------+
 |  tsmInit() 和 tsmQueryApiVersion() 中 API 版本的类型定义                |
 +-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t   stVersion;                /* 结构版本                          */
   dsUint16_t version;       /* API 版本                          */
   dsUint16_t release;       /* API 发行版                        */
   dsUint16_t level;         /* API 级别                          */
   dsUint16_t subLevel;      /* API 子级别                        */
   dsmBool_t  unicode;       /* 是否为 API unicode?                  */
} tsmApiVersionEx;
```

```
#define tsmApiVersionExVer       2

/*-------------------------------------------------------------------------+
|  tsmInit() 中应用程序版本的类型定义                                      |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t   stVersion;                /* 结构版本                          */
   dsUint16_t   applicationVersion;   /* 应用程序版本号                */
   dsUint16_t   applicationRelease;   /* 应用程序发行版号              */
   dsUint16_t   applicationLevel;     /* 应用程序级别号                */
   dsUint16_t   applicationSubLevel;  /* 应用程序子级别号              */
} tsmAppVersion;

#define tsmAppVersionVer      1


/*-------------------------------------------------------------------------+
 |  BindMC、发送、删除、查询时使用的对象名称的类型定义                     |
 +-------------------------------------------------------------------------*/

typedef struct tsmObjName
{
   dsChar_t   fs[DSM_MAX_FSNAME_LENGTH + 1] ;              /* 文件空间名称   */
   dsChar_t   hl[DSM_MAX_HL_LENGTH + 1] ;                 /* 高级别名称      */
   dsChar_t   ll[DSM_MAX_LL_LENGTH + 1] ;                  /* 低级别名称 */
   dsUint8_t  objType;          /* 关于对象类型的值，请参阅上面的定义        */
   dsChar_t   dirDelimiter;
} tsmObjName;

/*-------------------------------------------------------------------------+
 |  dsmDeleteObj() 中备份删除信息的类型定义                                 |
 +-------------------------------------------------------------------------*/
typedef struct tsmDelBack
{
   dsUint16_t       stVersion ;                    /* 结构版本               */
   tsmObjName       *objNameP ;                    /* 对象名                */
   dsUint32_t       copyGroup ;                    /* 副本组                 */
} tsmDelBack ;

#define  tsmDelBackVersion    1

/*-------------------------------------------------------------------------+
 |  dsmDeleteObj() 中归档删除信息的类型定义                                 |
 +-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t       stVersion ;                    /* 结构版本               */
   dsStruct64_t       objId ;                        /* 对象标识               */
} tsmDelArch ;

#define  tsmDelArchVersion  1

/*-------------------------------------------------------------------------+
 |  dsmDeleteObj() 中备份标识删除信息的类型定义                             |
 +-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t       stVersion ;                    /* 结构版本               */
   dsStruct64_t       objId ;                        /* 对象标识               */
} tsmDelBackID;

#define  tsmDelBackIDVersion  1


/*-------------------------------------------------------------------------+
 |  dsmDeleteObj() 中删除信息的类型定义                                     |
 +-------------------------------------------------------------------------*/
typedef union
{
   tsmDelBack   backInfo ;
   tsmDelArch   archInfo ;
   tsmDelBackID backIDInfo;
} tsmDelInfo ;

/*-------------------------------------------------------------------------+
 |  dsmSendObj() 中对象属性参数的类型定义                                   |
 +-------------------------------------------------------------------------*/
```

```
typedef struct tsmObjAttr
{
   dsUint16_t   stVersion;                /* 结构版本                              */
   dsChar_t     owner[DSM_MAX_OWNER_LENGTH + 1];       /* 对象名                 */
   dsStruct64_t sizeEstimate;             /* 对象的大小估计 ( 以字节表示 )       */
   dsmBool_t    objCompressed;            /* 对象是否已压缩？      */
   dsUint16_t   objInfoLength;            /* 与对象相关的信息的长度             */
   char         *objInfo;                 /* 与对象相关的信息字节缓冲区        */
   dsChar_t     *mcNameP;                 /* 要覆盖的管理类名称                 */
   tsmOwType    reserved1;                /* 以便将来使用                       */
   tsmOwType    reserved2;                /* 以便将来使用                       */
   dsmBool_t    disableDeduplication;            /* 对此对象不实施重复数据删除    */
   dsmBool_t    useExtObjInfo;             /* 使用扩展对象信息，最多 1536        */
} tsmObjAttr;

#define tsmObjAttrVersion 5

/*-------------------------------------------------------------------------+
 | dsmBindMC() 中返回的 mcBindKey 的类型定义                               |
 +-------------------------------------------------------------------------*/
typedef struct tsmMcBindKey
{
   dsUint16_t  stVersion;                 /* 结构版本            */
   dsChar_t    mcName[DSM_MAX_MC_NAME_LENGTH + 1];
   /* 与对象绑定的 mc 的名称。 */
   dsmBool_t   backup_cg_exists;                              /* True/false */
   dsmBool_t   archive_cg_exists;                             /* True/false */
   dsChar_t    backup_copy_dest[DSM_MAX_CG_DEST_LENGTH + 1];
   /* 备份副本目标名称       */
   dsChar_t    archive_copy_dest[DSM_MAX_CG_DEST_LENGTH + 1];
   /* 归档副本目标名称   */
} tsmMcBindKey;

#define tsmMcBindKeyVersion  1


/*-------------------------------------------------------------------------+
 |  dsmBeginQuery() 中管理类 queryBuffer 的类型定义                        |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryMCData
{
   dsUint16_t  stVersion;                 /* 结构版本            */
   dsChar_t     *mcName;                       /* 管理类名称    */
   /* 单个名称为获取一个或空字符串为获取全部           */
   dsmBool_t    mcDetail;                           /* 是否想要详细信息？ */
} tsmQryMCData;

#define tsmQryMCDataVersion  1


/*-------------------------------------------------------------------------+
 |  查询 MC 响应中归档副本组详细信息的类型定义                             |
 +-------------------------------------------------------------------------*/
typedef struct tsmArchDetailCG
{
   dsChar_t     cgName[DSM_MAX_CG_NAME_LENGTH + 1];     /* 副本组名称      */
   dsUint16_t   frequency;                      /* 副本 ( 归档 ) 频率       */
   dsUint16_t   retainVers;                             /* 保留版本       */
   dsUint8_t    copySer;       /* 用于副本序列化值，参阅定义               */
   dsUint8_t    copyMode;         /* 用于副本模式值，请参阅上面的定义      */
   dsChar_t     destName[DSM_MAX_CG_DEST_LENGTH + 1];    /* 副本目标名称   */
   dsmBool_t    bLanFreeDest;            /* 目标是否有不依赖 LAN 的路径？    */
   dsmBool_t    reserved;                /* 当前没有使用                     */
   dsUint8_t    retainInit;             /*  请参阅 dsmapitd.h 以了解可能的值*/
   dsUint16_t   retainMin;               /* retInit 为 EVENT 时的天数      */
   dsmBool_t    bDeduplicate;            /* 目标已重复启用                  */
}tsmArchDetailCG;

/*-------------------------------------------------------------------------+
 |  查询 MC 响应中备份副本组详细信息的类型定义                             |
 +-------------------------------------------------------------------------*/
typedef struct tsmBackupDetailCG
{
   dsChar_t     cgName[DSM_MAX_CG_NAME_LENGTH + 1];     /* 副本组名称      */
   dsUint16_t   frequency;                              /* 备份频率         */
   dsUint16_t   verDataExst;                          /* 存在版本数据         */
   dsUint16_t   verDataDltd;                         /* 已删除版本数据        */
   dsUint16_t   retXtraVers;                         /* 保留附加版本          */
```

```
   dsUint16_t   retOnlyVers;                          /* 仅保留版本          */
   dsUint8_t    copySer;       /* 用于副本序列化值，参阅定义                 */
   dsUint8_t    copyMode;         /* 用于副本模式值，请参阅上面的定义        */
   dsChar_t     destName[DSM_MAX_CG_DEST_LENGTH + 1];    /* 副本目标名称   */
   dsmBool_t    bLanFreeDest;            /* 目标是否有不依赖 LAN 的路径？    */
   dsmBool_t    reserved;                /* 当前没有使用                     */
   dsmBool_t    bDeduplicate;            /* 目标已重复启用                   */
}tsmBackupDetailCG;


/*-------------------------------------------------------------------------+
 |  dsmGetNextQObj() 中查询管理类详细信息响应的类型定义                     |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryRespMCDetailData
{
   dsUint16_t  stVersion;                /* 结构版本            */
   dsChar_t        mcName[DSM_MAX_MC_NAME_LENGTH + 1];        /* mc 名称 */
   dsChar_t        mcDesc[DSM_MAX_MC_DESCR_LENGTH + 1]; /*mc 描述        */
   archDetailCG    archDet;                /* 归档副本组详细信息        */
   backupDetailCG  backupDet;               /* 备份副本组详细信息       */
} tsmQryRespMCDetailData;

#define tsmQryRespMCDetailDataVersion  4

/*-------------------------------------------------------------------------+
 | dsmGetNextQObj() 中查询管理类摘要响应的类型定义                          |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryRespMCData
{
   dsUint16_t  stVersion;                /* 结构版本            */
   dsChar_t        mcName[DSM_MAX_MC_NAME_LENGTH + 1];        /* mc 名称 */
   dsChar_t        mcDesc[DSM_MAX_MC_DESCR_LENGTH + 1];      /*mc 描述  */
}tsmQryRespMCData;

#define tsmQryRespMCDataVersion  1

/*-------------------------------------------------------------------------+
 |  tsmBeginQuery() 中归档 queryBuffer 的类型定义                           |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryArchiveData
{
   dsUint16_t  stVersion;                /* 结构版本            */
   tsmObjName   *objName;                    /* 对象的完整 dsm 名称      */
   dsChar_t     *owner;                                 /* 所有者名称  */
   /* 对于最大日期界限，请参阅上面的定义            */
   dsmDate      insDateLowerBound;      /* 归档插入日期下限              */
   dsmDate      insDateUpperBound;       /* 归档插入日期上限             */
   dsmDate      expDateLowerBound;          /* 到期日期下限              */
   dsmDate      expDateUpperBound;           /* 到期日期上限             */
   dsChar_t     *descr;                /* 归档描述          */
} tsmQryArchiveData;

#define tsmQryArchiveDataVersion  1

/*-------------------------------------------------------------------------+
| dsmGetNextQObj() 中查询归档响应的类型定义                                |
+-------------------------------------------------------------------------*/
typedef struct tsmQryRespArchiveData
{
    dsUint16_t  stVersion;                 /* 结构版本            */
    tsmObjName      objName;                      /* 文件空间名称限定符    */
    dsUint32_t      copyGroup;                          /* 副本组编号        */
    dsChar_t        mcName[DSM_MAX_MC_NAME_LENGTH + 1];        /* mc 名称 */
    dsChar_t        owner[DSM_MAX_OWNER_LENGTH + 1];            /* 所有者名称 */
    dsStruct64_t    objId;                                  /* 唯一的副本标识 */
    dsStruct64_t    reserved;                       /* 向后兼容               */
    dsUint8_t       mediaClass;                         /* 媒介访问类         */
    dsmDate         insDate;                       /* 归档插入日期           */
    dsmDate         expDate;                     /* 对象的到期日期           */
    dsChar_t        descr[DSM_MAX_DESCR_LENGTH + 1];   /* 归档描述            */
    dsUint16_t      objInfolen;             /* 与对象相关的信息的长度        */
    dsUint8_t       reservedObjInfo[DSM_MAX_OBJINFO_LENGTH];  /*与对象相关的信息 */
    dsUint160_t     restoreOrderExt;                        /* 恢复顺序     */
    dsStruct64_t    sizeEstimate;              /* 用户存储的大小估计         */
    dsUint8_t       compressType;                       /* 压缩标志       */
    dsUint8_t       retentionInitiated;   /* 在保留对象事件中等待的对象      */
    dsUint8_t       对象处于“保留”状态，请参阅 dsmapitd.h 了解关于值的信息*/
```

```
    dsUint8_t       encryptionType;                      /* 加密类型         */
    dsmBool_t       clientDeduplicated;    /* API 删除的重复对象*/
    dsUint8_t       objInfo[DSM_MAX_EXT_OBJINFO_LENGTH];  /*与对象相关的信息 */
    dsChar_t        compressAlg[DSM_MAX_COMPRESSTYPE_LENGTH + 1];  /* 压缩算法名称 */
} tsmQryRespArchiveData;

#define tsmQryRespArchiveDataVersion  7

/*-------------------------------------------------------------------------+
 |  dsmSendObj() 中归档 sendBuff 参数的类型定义                             |
 +-------------------------------------------------------------------------*/
typedef struct tsmSndArchiveData
{
   dsUint16_t  stVersion;                /* 结构版本            */
   dsChar_t     *descr;                /* 归档描述          */
} tsmSndArchiveData;

#define tsmSndArchiveDataVersion  1

/*-------------------------------------------------------------------------+
 |  dsmBeginQuery() 中备份 queryBuffer 的类型定义                           |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryBackupData
{
   dsUint16_t  stVersion;                /* 结构版本            */
   tsmObjName  *objName;          /* 对象的完整 dsm 名称    */
   dsChar_t     *owner;           /* 所有者名称   */
   dsUint8_t   objState;          /* 对象状态选择器          */
   dsmDate     pitDate;           /* 及时恢复的日期值       */
   /* 对于可能的值，请参阅上面的定义 */
    dsUint32_t  reserved1;
    dsUint32_t  reserved2;
} tsmQryBackupData;

#define tsmQryBackupDataVersion  3

/*-------------------------------------------------------------------------+
 | dsmGetNextQObj() 中查询备份响应的类型定义                                |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryRespBackupData
{
   dsUint16_t  stVersion;                /* 结构版本            */
   tsmObjName      objName;                      /* 对象的完整 dsm 名称     */
   dsUint32_t      copyGroup;                      /* 副本组编号         */
   dsChar_t        mcName[DSM_MAX_MC_NAME_LENGTH + 1];        /* mc 名称 */
   dsChar_t        owner[DSM_MAX_OWNER_LENGTH + 1];            /* 所有者名称 */
   dsStruct64_t    objId;                                /* 唯一的对象标识   */
   dsStruct64_t    reserved;                        /* 向后兼容               */
   dsUint8_t       mediaClass;                          /* 媒介访问类          */
   dsUint8_t       objState;                      /* 对象状态、活动等       */
   dsmDate         insDate;                        /* 备份插入日期          */
   dsmDate         expDate;                    /* 对象的到期日期             */
   dsUint16_t      objInfolen;             /* 与对象相关的信息的长度         */
   dsUint8_t       reservedObjInfo[DSM_MAX_OBJINFO_LENGTH];  /*与对象相关的信息 */
   dsUint160_t     restoreOrderExt;                          /* 恢复顺序     */
   dsStruct64_t    sizeEstimate;              /* 用户存储的大小估计         */
   dsStruct64_t    baseObjId;
   dsUint16_t      baseObjInfolen;              /* 与对象相关的信息的长度  */
   dsUint8_t       baseObjInfo[DSM_MAX_OBJINFO_LENGTH];  /* 与基本对象相关的信息 */
   dsUint160_t     baseRestoreOrder;                          /* 恢复顺序       */
   dsUint32_t      fsID;
   dsUint8_t       compressType;
   dsmBool_t       isGroupLeader;
   dsmBool_t       isOpenGroup;
   dsUint8_t       reserved1;               /* 以便将来使用   */
   dsmBool_t       reserved2;               /* 以便将来使用   */
   dsUint16_t      reserved3;               /* 以便将来使用   */
   reservedInfo_t  *reserved4;              /* 以便将来使用   */
   dsUint8_t       encryptionType;         /* 加密类型         */
   dsmBool_t       clientDeduplicated;    /* API 删除的重复对象*/
   dsUint8_t       objInfo[DSM_MAX_EXT_OBJINFO_LENGTH];  /*与对象相关的信息 */
   dsChar_t        compressAlg[DSM_MAX_COMPRESSTYPE_LENGTH + 1];  /* 压缩算法名称 */
} tsmQryRespBackupData;

#define tsmQryRespBackupDataVersion  8
/*-------------------------------------------------------------------------+
 |  dsmBeginQuery() 中活动备份 queryBuffer 的类型定义
 |
```

```
 |  注意：有关活动备份的查询，只需设置 objName 的 fs（文件空间）和 objType
 |          字段。  objType 只能设为
 |          DSM_OBJ_FILE 或 DSM_OBJ_DIRECTORY。  DSM_OBJ_ANY_TYPE 不会在查询中
 |          找到匹配。
 +----------------------------------------------------------------------------*/
typedef struct tsmQryABackupData
{
   dsUint16_t  stVersion;                /* 结构版本            */
   tsmObjName      *objName;                    /* 仅使用 fs 和 objtype     */
} tsmQryABackupData;

#define tsmQryABackupDataVersion  1

/*----------------------------------------------------------------------------+
 | dsmGetNextQObj() 中查询活动备份响应的类型定义                              |
 +----------------------------------------------------------------------------*/
typedef struct tsmQryARespBackupData
{
   dsUint16_t  stVersion;                /* 结构版本            */
   tsmObjName      objName;                     /* 对象的完整 dsm 名称     */
   dsUint32_t      copyGroup;                /* 副本组编号              */
   dsChar_t    mcName[DSM_MAX_MC_NAME_LENGTH + 1];/*管理类名称          */
   dsChar_t        owner[DSM_MAX_OWNER_LENGTH + 1];      /* 所有者名称 */
   dsmDate         insDate;                  /* 备份插入日期         */
   dsUint16_t      objInfolen;         /* 与对象相关的信息的长度      */
   dsUint8_t   reservedObjInfo[DSM_MAX_OBJINFO_LENGTH];  /*与对象相关的信息 */
   dsUint8_t  objInfo[DSM_MAX_EXT_OBJINFO_LENGTH];  /*与对象相关的信息 */
} tsmQryARespBackupData;

#define tsmQryARespBackupDataVersion  2

/*----------------------------------------------------------------------------+
 |  dsmBeginQuery() 中备份 queryBuffer 的类型定义                             |
 +----------------------------------------------------------------------------*/
typedef struct tsmQryBackupGroups
{
   dsUint16_t  stVersion;                /* 结构版本            */
   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

/*----------------------------------------------------------------------------+
 |  tsmBeginQuery() 中的代理节点 queryBuffer 的类型定义                       |
 +----------------------------------------------------------------------------*/
typedef struct tsmQryProxyNodeData
{
   dsUint16_t  stVersion;                /* 结构版本            */
   dsChar_t    *targetNodeName;             /* 目标节点名称          */
}tsmQryProxyNodeData;

#define tsmQryProxyNodeDataVersion  1

/*----------------------------------------------------------------------------+
 | tsmGetNextQObj() 中使用的 qryRespProxyNodeData 参数的类型定义              |
 +----------------------------------------------------------------------------*/

typedef struct tsmQryRespProxyNodeData
{
   dsUint16_t       stVersion ;                          /* 结构版本          */
   dsChar_t          targetNodeName[DSM_MAX_ID_LENGTH+1];  /* 目标节点名称       */
   dsChar_t          peerNodeName[DSM_MAX_ID_LENGTH+1];    /* 同级节点名称       */
   dsChar_t          hlAddress[DSM_MAX_ID_LENGTH+1];       /* 同级 hlAddress     */
   dsChar_t          llAddress[DSM_MAX_ID_LENGTH+1];       /* 同级 llAddress     */
}tsmQryRespProxyNodeData;

#define tsmQryRespProxyNodeDataVersion  1

/*----------------------------------------------------------------------------+
 |  WINNT 和 OS/2 文件空间属性的类型定义                                      |
```

```
 +-------------------------------------------------------------------------*/
typedef struct tsmDosFSAttrib
{
   osChar_t     driveLetter ;          /* 文件空间的盘符                */
   dsUint16_t   fsInfoLength;          /* 使用的 fsInfo 长度             */
   osChar_t     fsInfo[DSM_MAX_FSINFO_LENGTH];/*调用者确定的数据       */
} tsmDosFSAttrib ;

/*-------------------------------------------------------------------------+
 |  UNIX 文件空间属性的类型定义                                            |
 +-------------------------------------------------------------------------*/
typedef struct tsmUnixFSAttrib
{
   dsUint16_t   fsInfoLength;          /* 使用的 fsInfo 长度             */
   osChar_t     fsInfo[DSM_MAX_FSINFO_LENGTH];/*调用者确定的数据       */
} tsmUnixFSAttrib ;

/*-------------------------------------------------------------------------+
 |  NetWare 文件空间属性的类型定义                                         |
 +-------------------------------------------------------------------------*/
typedef tsmUnixFSAttrib tsmNetwareFSAttrib;

/*-------------------------------------------------------------------------+
 |  所有文件空间调用上文件空间属性的类型定义                               |
 +-------------------------------------------------------------------------*/
typedef union
{
   tsmNetwareFSAttrib  netwareFSAttr;
   tsmUnixFSAttrib     unixFSAttr ;
   tsmDosFSAttrib      dosFSAttr ;
} tsmFSAttr ;

/*-------------------------------------------------------------------------+
 |  dsmUpdateFS() 上 fsUpd 参数的类型定义
 +-------------------------------------------------------------------------*/
typedef struct   tsmFSUpd
{
   dsUint16_t stVersion ;                /* 结构版本                      */
   dsChar_t        *fsType ;            /* 文件空间类型                  */
   dsStruct64_t    occupancy ;          /* 占用率估计                    */
   dsStruct64_t    capacity  ;          /* 容量估计                      */
   tsmFSAttr       fsAttr ;             /* 特定平台的属性               */
} tsmFSUpd ;

#define tsmFSUpdVersion  1

/*-------------------------------------------------------------------------+
 |  dsmBeginQuery() 中文件空间 queryBuffer 的类型定义                      |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryFSData
{
   dsUint16_t  stVersion;               /* 结构版本            */
   dsChar_t    *fsName;                 /* 文件空间名称           */
} tsmQryFSData;

#define tsmQryFSDataVersion  1

/*-------------------------------------------------------------------------+
 | dsmGetNextQObj() 中查询文件空间响应的类型定义                           |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryRespFSData
{
   dsUint16_t  stVersion;               /* 结构版本            */
   dsChar_t       fsName[DSM_MAX_FSNAME_LENGTH + 1];     /* 文件空间名称   */
   dsChar_t       fsType[DSM_MAX_FSTYPE_LENGTH + 1] ;     /* 文件空间类型  */
   dsStruct64_t   occupancy;                  /* 占用率估计 ( 以字节表示 )。              */
   dsStruct64_t   capacity;                    /* 容量估计 ( 以字节表示 )。               */
   tsmFSAttr       fsAttr ;                /* 特定平台的属性                */
   dsmDate        backStartDate;          /* 开始备份日期                  */
   dsmDate        backCompleteDate;       /* 结束备份日期                  */
    dsmDate        reserved1;                  /* 以便将来使用            */
   dsmBool_t      bIsUnicode;
   dsUint32_t     fsID;
   dsmDate        lastReplStartDate;        /* 最近一次启动复制的时间               */
   dsmDate        lastReplCmpltDate;        /* 最近一次完成复制的时间               */
                                            /*    ( 可能已发生故障 ,                 */
                                            /*     但是仍完成 )                      */
   dsmDate        lastBackOpDateFromServer; /* 上次客户机在服务器上进行保存的        */
```

```
                                          /*     存储时间戳记                          */
   dsmDate        lastArchOpDateFromServer; /* 上次客户机在服务器上进行保存的        */
                                          /*     存储时间戳记                          */
   dsmDate        lastSpMgOpDateFromServer; /* 上次客户机在服务器上进行保存的        */
                                          /*     存储时间戳记                          */
   dsmDate        lastBackOpDateFromLocal;  /* 上次客户机在本地进行保存的            */
                                          /*     存储时间戳记                          */
   dsmDate        lastArchOpDateFromLocal;  /* 上次客户机在本地进行保存的            */
                                          /*     存储时间戳记                          */
   dsmDate        lastSpMgOpDateFromLocal;  /* 上次客户机在本地进行保存的            */
                                          /*     存储时间戳记                          */
   dsInt32_t      failOverWriteDelay;       /* 在允许存储到此复制服务器之前              */
                                          /* 客户机等待的分钟数，特殊代码：            */
                                          /* NO_ACCESS(-1)，ACCESS_RDONLY (-2)         */
} tsmQryRespFSData;

#define tsmQryRespFSDataVersion 5

/*-------------------------------------------------------------------------+
 |  dsmRegisterFS() 中 regFilespace 参数的类型定义
 +-------------------------------------------------------------------------*/
typedef struct tsmRegFSData
{
   dsUint16_t  stVersion;                /* 结构版本             */
   dsChar_t        *fsName;                      /* 文件空间名称          */
   dsChar_t        *fsType;                      /* 文件空间类型          */
   dsStruct64_t   occupancy;                /* 占用率估计（以字节表示）。 */
   dsStruct64_t   capacity;                  /* 容量估计（以字节表示）。 */
   tsmFSAttr       fsAttr ;               /* 特定平台的属性               */
} tsmRegFSData;

#define tsmRegFSDataVersion  1

/*-------------------------------------------------------------------------+
 | dsmQuerySessionInfo() 中会话信息响应的类型定义                          |
 +-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t   stVersion;               /* 结构版本                        */
   /*---------------------------------------------------------------------*/
   /*           服务器信息                                                */
   /*---------------------------------------------------------------------*/
   dsChar_t       serverHost[DSM_MAX_SERVERNAME_LENGTH+1];
   /* DSM 服务器的网络主机     */
   dsUint16_t     serverPort;            /* 主机上服务器的常用端口          */
   dsmDate        serverDate;            /* 服务器的日期/时间               */
   dsChar_t       serverType[DSM_MAX_SERVERTYPE_LENGTH+1];
   /* 服务器的执行平台                 */
   dsUint16_t     serverVer;             /* 服务器的版本号                  */
   dsUint16_t     serverRel;             /* 服务器的发行号                  */
   dsUint16_t     serverLev;             /* 服务器的级别号                  */
   dsUint16_t     serverSubLev;          /* 服务器的子级别号                */
   /*---------------------------------------------------------------------*/
   /*           客户机缺省值                                              */
   /*---------------------------------------------------------------------*/
   dsChar_t       nodeType[DSM_MAX_PLATFORM_LENGTH+1]; /*节点/应用程序类型    */
   dsChar_t       fsdelim;               /* 文件空间分隔符                  */
   dsChar_t       hldelim;               /* 高级别与低级别间的分隔符        */
   dsUint8_t      compression;           /* 压缩标志                        */
   dsUint8_t      archDel;               /* 归档删除权限                    */
   dsUint8_t      backDel;               /* 备份删除权限                    */
   dsUint32_t     maxBytesPerTxn;        /* 以便将来使用                    */
   dsUint16_t     maxObjPerTxn;          /* txn 中允许的最大对象数          */
   /*---------------------------------------------------------------------*/
   /*           会话信息                                                  */
   /*---------------------------------------------------------------------*/
   dsChar_t       id[DSM_MAX_ID_LENGTH+1];    /* 登录标识节点名称          */
   dsChar_t       owner[DSM_MAX_OWNER_LENGTH+1]; /* 登录所有者             */
   /*  （用于多用户平台）                */
   dsChar_t       confFile[DSM_PATH_MAX + DSM_NAME_MAX +1];
   /* len 是平台 dep                    */
   /* dsInit name of appl config file  */
   dsUint8_t      opNoTrace;             /* dsInit option - NoTrace = 1     */
   /*---------------------------------------------------------------------*/
   /*           策略数据                                                  */
   /*---------------------------------------------------------------------*/
   dsChar_t       domainName[DSM_MAX_DOMAIN_LENGTH+1]; /* 域名            */
```

```
   dsChar_t       policySetName[DSM_MAX_PS_NAME_LENGTH+1];
   /* 活动策略集名称                  */
   dsmDate        polActDate;             /* 策略集激活日期               */
   dsChar_t       dfltMCName[DSM_MAX_MC_NAME_LENGTH+1];/* 缺省管理类      */
   dsUint16_t     gpBackRetn;             /* Grace-period 备份保留        */
   dsUint16_t     gpArchRetn;             /* Grace-period 归档保留        */
   dsChar_t       adsmServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* adsm 服务器名称       */
  dsmBool_t      archiveRetentionProtection; /* 启用服务器保留保护             */
   dsUint64_t     maxBytesPerTxn_64;         /* 以便将来使用                    */
   dsmBool_t      lanFreeEnabled;            /* 已设置不依赖 LAN 的选项         */
   dsmDedupType   dedupType;                 /* 服务器或 clientOrServer         */
   dsChar_t       accessNode[DSM_MAX_ID_LENGTH+1];    /* 作为节点名称           */

   /*-------------------------------------------------------------------*/
   /*           复制和故障转移信息                                      */
   /*-------------------------------------------------------------------*/
   dsmFailOvrCfgType failOverCfgType; /* 故障转移的状态 */
   dsChar_t       replServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* 复制服务器名称 */
   dsChar_t       homeServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* 主服务器名称   */
   dsChar_t       replServerHost[DSM_MAX_SERVERNAME_LENGTH+1]; /* DSM 服务器的网络主机名  */
   dsInt32_t      replServerPort;                              /* 主机上服务器的常用端口  */

} tsmApiSessInfo;

#define tsmApiSessInfoVersion  6

/*-------------------------------------------------------------------------+
 | dsmQueryCliOptions() 和 dsmQuerySessOptions() 中查询选项                |
 |      响应的类型定义                                                     |
 +-------------------------------------------------------------------------*/

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

/*-------------------------------------------------------------------------+
 | dsmQueryAccess() 中使用的 qryRespAccessData 参数的类型定义              |
 +-------------------------------------------------------------------------*/

typedef struct
{
   dsUint16_t stVersion ;                          /* 结构版本               */
   dsChar_t          node[DSM_MAX_ID_LENGTH+1];    /* 节点名称               */
   dsChar_t          owner[DSM_MAX_OWNER_LENGTH+1]; /* 所有者                */
   tsmObjName        objName ;                     /* 对象名                */
   dsmAccessType     accessType;                   /* 归档或备份             */
   dsUint32_t        ruleNumber ;                  /* 访问规则标识           */

}tsmQryRespAccessData;

#define tsmQryRespAccessDataVersion  1

/*-------------------------------------------------------------------------+
 |  dsmSetUp() 中 envSetUp 参数的类型定义
 +-------------------------------------------------------------------------*/
typedef struct tsmEnvSetUp
{
   dsUint16_t  stVersion;                 /* 结构版本            */
   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;  /* for executables name argv[0] */
   dsChar_t        logName[DSM_NAME_MAX +1];
   dsmBool_t       reserved1;         /* 以便将来使用    */
   dsmBool_t       reserved2;                  /* 以便将来使用  */
} tsmEnvSetUp;
```

```
#define tsmEnvSetUpVersion  4

/*-------------------------------------------------------------------------+
 |  dsmInitExIn_t 的类型定义
 +-------------------------------------------------------------------------*/
typedef struct tsmInitExIn_t
{
   dsUint16_t  stVersion;                /* 结构版本            */
   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

/*-------------------------------------------------------------------------+
 |  dsmInitExOut_t 的类型定义
 +-------------------------------------------------------------------------*/
typedef struct tsmInitExOut_t
{
   dsUint16_t  stVersion;                /* 结构版本            */
   dsInt16_t          userNameAuthorities;
   dsInt16_t          infoRC;        /* 如果遇到，将有错误返回码         */
   /* adsm 服务器名称                 */
   dsChar_t           adsmServerName[DSM_MAX_SERVERNAME_LENGTH+1];
   dsUint16_t         serverVer;      /* 服务器的版本号                   */
   dsUint16_t         serverRel;     /* 服务器的发行号                    */
   dsUint16_t         serverLev;     /* 服务器的级别号                    */
   dsUint16_t         serverSubLev;  /* 服务器的子级别号                 */
   dsmBool_t          bIsFailOverMode; /* true，如果已发生故障转移 */
   dsChar_t           replServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* 复制服务器名称 */
   dsChar_t           homeServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* 主服务器名称   */
} tsmInitExOut_t;

#define tsmInitExOutVersion 3

/*-------------------------------------------------------------------------+
 |  dsmLogExIn_t 的类型定义
 +-------------------------------------------------------------------------*/
typedef struct tsmLogExIn_t
{
   dsUint16_t  stVersion;                /* 结构版本            */
   dsmLogSeverity     severity;
   dsChar_t           appMsgID[8];
   dsmLogType  logType;    /* 记录类型：本地和/或服务器     */
   dsChar_t           *message;   /* 要记录的消息文本      */
   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

/*-------------------------------------------------------------------------+
 |  dsmlogExOut_t 的类型定义
 +-------------------------------------------------------------------------*/
typedef struct tsmLogExOut_t
{
   dsUint16_t  stVersion;                /* 结构版本            */
} tsmLogExOut_t;

#define tsmLogExOutVersion 1

/*-------------------------------------------------------------------------+
```

```
 |  dsmRenameIn_t 的类型定义
 +-------------------------------------------------------------------------*/
typedef struct tsmRenameIn_t
{
   dsUint16_t  stVersion;                 /* 结构版本             */
   dsUint32_t       tsmHandle;                     /* 会话句柄           */
   dsUint8_t        repository;                    /* 备份或归档     */
   tsmObjName       *objNameP ;                    /* 对象名               */
   dsChar_t         newHl[DSM_MAX_HL_LENGTH + 1];  /* 新的高级别名称         */
   dsChar_t         newLl[DSM_MAX_LL_LENGTH + 1];  /* 新的低级别名称         */
   dsmBool_t        merge;                        /* 合并到现有名称中*/
   ObjID            objId;                         /* 归档的对象标识 */
} tsmRenameIn_t;

#define tsmRenameInVersion 1

/*-------------------------------------------------------------------------+
 |  dsmRenameOut_t 的类型定义
 +-------------------------------------------------------------------------*/
typedef struct tsmRenameOut_t
{
   dsUint16_t  stVersion;                 /* 结构版本             */
} tsmRenameOut_t;

#define tsmRenameOutVersion 1

/*-------------------------------------------------------------------------+
 |  tsmEndSendObjExIn_t 的类型定义
 +-------------------------------------------------------------------------*/
typedef struct tsmEndSendObjExIn_t
{
   dsUint16_t  stVersion;                 /* 结构版本             */
   dsUint32_t       tsmHandle;                     /* 会话句柄           */
} tsmEndSendObjExIn_t;

#define tsmEndSendObjExInVersion 1

/*-------------------------------------------------------------------------+
 |  dsmEndSendObjExOut_t 的类型定义
 +-------------------------------------------------------------------------*/
typedef struct tsmEndSendObjExOut_t
{
   dsUint16_t  stVersion;                 /* 结构版本             */
   dsStruct64_t       totalBytesSent;      /* 从应用程序读取的总字节数  */
   dsmBool_t          objCompressed;       /* 压缩对象                    */
   dsStruct64_t       totalCompressSize;   /* 压缩后的总大小             */
   dsStruct64_t       totalLFBytesSent;    /* 发送不依赖 LAN 的总字节数 */
   dsUint8_t          encryptionType;      /* 使用的加密类型             */
   dsmBool_t          objDeduplicated;     /* 为分发数据重复删除而处理的对象 */
   dsStruct64_t       totalDedupSize;      /* 删除重复数据或的总大小       */
}tsmEndSendObjExOut_t;

#define tsmEndSendObjExOutVersion 3

/*-------------------------------------------------------------------------+
 |  tsmGroupHandlerIn_t 的类型定义
 +-------------------------------------------------------------------------*/
typedef struct tsmGroupHandlerIn_t
{
   dsUint16_t  stVersion;                 /* 结构版本             */
   dsUint32_t       tsmHandle;                     /* 会话句柄           */
   dsUint8_t       groupType;        /* 组的类型                          */
   dsUint8_t       actionType;       /* 组操作的类型                      */
   dsUint8_t       memberType;       /* 成员类型：领导或成员             */
   dsStruct64_t     leaderObjId;      /* 组领导的 OBJID                    */
   dsChar_t         *uniqueGroupTagP; /* 唯一的组标识                      */
   tsmObjName       *objNameP ;       /* 组领导的对象名称                  */
   dsmGetList      memberObjList;    /* 要除去、分配的对象列表           */
} tsmGroupHandlerIn_t;

#define tsmGroupHandlerInVersion 1

/*-------------------------------------------------------------------------+
 |  tsmGroupHandlerExOut_t 的类型定义
 +-------------------------------------------------------------------------*/
typedef struct tsmGroupHandlerOut_t
{
   dsUint16_t  stVersion;                 /* 结构版本             */
```

```
} tsmGroupHandlerOut_t;

#define tsmGroupHandlerOutVersion 1

/*--------------------------------------------------------------------------+
 |  tsmEndTxnExIn_t 的类型定义
 +--------------------------------------------------------------------------*/
typedef struct tsmEndTxnExIn_t
{
   dsUint16_t  stVersion;                /* 结构版本             */
   dsUint32_t       tsmHandle;                    /* 会话句柄           */
   dsUint8_t        vote;
} tsmEndTxnExIn_t;

#define tsmEndTxnExInVersion 1

/*--------------------------------------------------------------------------+
 |  tsmEndTxnExOut_t 的类型定义
 +--------------------------------------------------------------------------*/
typedef struct tsmEndTxnExOut_t
{
   dsUint16_t  stVersion;                /* 结构版本             */
   dsUint16_t         reason;                 /* 原因码                          */
   dsStruct64_t       groupLeaderObjId;       /* 返回的组领导的对象标识         */
   /* DSM_ACTION_OPEN                */
    dsUint8_t          reserved1;              /* 将来使用                        */
    dsUint16_t         reserved2;              /* 将来使用                        */
} tsmEndTxnExOut_t;

#define tsmEndTxnExOutVersion 1

/*--------------------------------------------------------------------------+
 |  tsmEndGetDataExIn_t 的类型定义
 +--------------------------------------------------------------------------*/
typedef struct tsmEndGetDataExIn_t
{
   dsUint16_t  stVersion;                /* 结构版本             */
   dsUint32_t       tsmHandle;                    /* 会话句柄           */
}tsmEndGetDataExIn_t;

#define tsmEndGetDataExInVersion 1

/*--------------------------------------------------------------------------+
 |  tsmEndGetDataExOut_t 的类型定义
 +--------------------------------------------------------------------------*/
typedef struct tsmEndGetDataExOut_t
{
   dsUint16_t  stVersion;                /* 结构版本             */
   dsUint16_t         reason;                 /* 原因码                          */
   dsStruct64_t   totalLFBytesRecv;           /* 接收到的不依赖 LAN 的总字节数   */
}tsmEndGetDataExOut_t;

#define tsmEndGetDataExOutVersion 1

/*--------------------------------------------------------------------------+
 | tsmRetentionEvent() 的类型定义                                           |
 +--------------------------------------------------------------------------*/
typedef struct tsmRetentionEventIn_t
{
   dsUint16_t  stVersion;                /* 结构版本             */
   dsUint32_t         tsmHandle;                   /* 会话句柄             */
   dsmEventType_t     eventType;                   /* 事件类型            */
   dsmObjList_t       objList;                     /* 对象标识            */
}tsmRetentionEventIn_t;

#define tsmRetentionEventInVersion 1

/*--------------------------------------------------------------------------+
 | tsmRetentionEvent() 的类型定义                                           |
 +--------------------------------------------------------------------------*/
typedef struct tsmRetentionEventOut_t
{
   dsUint16_t       stVersion ;                  /* 结构版本               */
}tsmRetentionEventOut_t;

#define tsmRetentionEventOutVersion 1

/*--------------------------------------------------------------------------+
 |  tsmUpdateObjExIn_t 的类型定义
```

```
 +-------------------------------------------------------------------------*/
typedef struct tsmUpdateObjExIn_t
{
   dsUint16_t  stVersion;                  /* 结构版本            */
   dsUint32_t         tsmHandle;          /* 会话句柄            */
   tsmSendType     sendType;              /* 发送类型备份/归档    */
   dsChar_t        *descrP;               /* 归档描述            */
   tsmObjName      *objNameP;             /* 对象名             */
   tsmObjAttr      *objAttrPtr;           /* 属性               */
   dsUint32_t      objUpdAct;            /* 更新操作            */
   ObjID           archObjId;           /* 归档的对象标识       */
}tsmUpdateObjExIn_t;

#define tsmUpdateObjExInVersion 1

/*-------------------------------------------------------------------------+
 |  tsmUpdateObjExOut_t 的类型定义
 +-------------------------------------------------------------------------*/
typedef struct tsmUpdateObjExOut_t
{
   dsUint16_t  stVersion;                  /* 结构版本            */
}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                    *
***********************************************************************/

/**************************************************************************
* 头文件名称：dsmapips.h
*
* 环境：            *********************************************
*                    ** 这是特定于平台的源文件                 **
*                    ** 针对 Windows NT 的版本                 **
*
*                    *********************************************
*
* 设计说明：         本文件包括与平台有关的定义
*
* 描述性名称：IBM Spectrum Protect typedefs 和 LINKAGE 的定义
*-----------------------------------------------------------------------*/

#ifndef _H_DSMAPIPS
#define _H_DSMAPIPS


#ifndef _WIN64
#pragma pack(1)
#endif

/*<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>*/
/*                      T Y P E D E F S                                   */
/*<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>*/
```

```
/* V3 的新 typedef 文件          */


#if !defined(DSMAPILIB) || defined (XOPEN_BUILD)

/* 支持链接     */
#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;

/*=== 字符和字符串类型         ===*/
#ifdef UNICODE
  typedef wchar_t dsChar_t;
  #define dsTEXT(x)       L##x
#else
  typedef char dsChar_t;
  #define dsTEXT(x)       x
#endif  /* !UNICODE */

/*=== 从 dsChar_t 获取的常见 typedefs 和定义   ===*/
typedef dsChar_t            *dsString_t;

  /* 为扩展的恢复顺序添加       */
   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;        /* 最高有效 32 位。 */
      dsUint32_t lo;        /* 最低有效 32 位。 */
   } dsUint64_t;
#endif

/*-------------------------------------------------------------------------+
| for bool_t 的类型定义                                                    |
+-------------------------------------------------------------------------*/
/*
 *  创建与任何操作系统或窗口系统中任何其他预定义
 *  的版本都不冲突的 Boolean 类型。
 */
typedef enum
{
   dsmFalse = 0x00,
   dsmTrue  = 0x01
}dsmBool_t ;


/*===  向后兼容      ===*/
#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;        /* 最高有效 32 位。 */
   dsUint32_t lo;        /* 最低有效 32 位。 */
}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                                    *
*****************************************************************************/

/*****************************************************************************
* 头文件名称：release.h
*
* 环境：       ***********************************************
*                  ** 这是与平台无关的源文件     **
*                  ***********************************************
*
* 设计说明：        本文件包含有关实际版本、发行版、级别、子级别
*                  的常见信息
*
* 描述性名称：IBM Spectrum Protect 版本的定义
*
* 注意：本文件不应包含 LOG 或 CMVC 信息。 API 代码中
*       随附了此信息。
*
*----------------------------------------------------------------------------*/

#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"

/*============================================================================
   以下字符串定义用于 VERSION 信息，
   不应转换为 dsTEXT 或 osTEXT。  它们仅在链接时
   使用。

   以下内容还在 Unix 上构建 Jar 文件时使用。  请参阅
   perl 脚本工具/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     ""

/*=== product names definitions ===*/
#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
```

```
/*============================================================================
   内部版本、发行版和级别 ( 构建 ) 版本。   This
   对于产品的每个版本、发行版和 ptf 应是唯一的。
   为了便于诊断，文件属性和数据流中都记录了
   此信息。
   注意：不要修改这些值。 您只能添加新的条目！
============================================================================*/
#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

/*=== 将 VRL 定义为整数以进行位图版本比较 ====*/
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 */
```

附录 C API 函数定义源文件

本附录包含 dsmapifp.h 头文件，因此，您可以查看 API 的函数定义。

注: 针对每个操作系统，**DSMLINKAGE** 定义也有所不同。 关于特定的操作系统，请参阅 dsmapips.h 文件中的定义。

此处提供的信息包含随 API 一起分发的文件的时间点副本。查看 API 发行软件包中的文件以获取最新版本。

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                    *
*****************************************************************************/

/*****************************************************************************/
/* 头文件名称：dsmapifp.h                                                     */
/*                                                                           */
/* 描述性名称：IBM Spectrum Protect API 函数原型                             */
/*****************************************************************************/
#ifndef _H_DSMAPIFP
#define _H_DSMAPIFP

#if defined(__cplusplus)
extern "C" {
#endif

#ifdef DYNALOAD_DSMAPI

/* 将动态装入的函数                                    */
#include "dsmapidl.h"

#else

/* 将从库中隐式装入的函数                                                    */

/*===========================================================================*/
/*               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

本节包含 API 的函数定义。 它是 tsmapifp.h 头文件的副本。

注: 针对每个操作系统，**DSMLINKAGE** 定义也有所不同。 关于特定操作系统，请参阅 tsmapips.h 文件中的定义。

```
/*****************************************************************************
*       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                          *
*****************************************************************************/

/*****************************************************************************/
/* 头文件名称：tsmapifp.h                                                     */
/*                                                                              */
/* 描述性名称：IBM Spectrum Protect API 函数原型                              */
/*****************************************************************************/
#ifndef _H_TSMAPIFP
#define _H_TSMAPIFP

#if defined(__cplusplus)
extern "C" {
#endif

#ifdef DYNALOAD_DSMAPI

/* 将动态装入的函数                                    */
#include "dsmapidl.h"

#else

/* 将从库中隐式装入的函数                                                     */

/*===========================================================================*/
/*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 */
```

附录 D IBM Spectrum Protect 产品系列的辅助功能

辅助功能可帮助身体残障（如行动受限或视力不佳）的用户顺利使用信息技术内容。

概述

IBM Spectrum Protect 系列产品包括下列主要辅助功能：

· 仅键盘的操作

· 使用屏幕朗读器的操作

IBM Spectrum Protect 系列产品使用最新的 W3C 标准 WAI-ARIA 1.0 (www.w3.org/TR/wai-aria/)，以确保符合 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/)。 要利用辅助功能，请使用屏幕朗读器的最新发行版以及产品支持的最新 Web 浏览器。

针对辅助功能启用 IBM Knowledge Center 中的产品文档。 IBM Knowledge Center 帮助的“辅助功能”部分 (www.ibm.com/support/knowledgecenter/about/releasenotes.html?view=kc#accessibility) 中描述了 IBM Knowledge Center 的辅助功能。

键盘导航

此产品使用标准导航键。

界面信息

用户界面上不存在每秒闪烁 2 - 55 次的内容。

Web 用户界面依靠级联样式表来正确呈现内容和提供可用体验。 此应用程序为视力不佳的用户使用系统显示设置提供了等效方法，包括高对比度方式。 您可以使用设备或 Web 浏览器设置来控制字体大小。

Web 用户界面包含可用于快速导航至应用程序中的功能区域的 WAI-ARIA 导航地标。

供应商软件

IBM Spectrum Protect 产品系列包含 IBM 许可协议未覆盖的某些供应商软件。 IBM 对这些产品的辅助功能不作任何说明。 请联系供应商以获取其产品的辅助功能选项信息。

相关的辅助功能选项信息

除了标准 IBM 帮助台和支持 Web 站点，IBM 还提供了 TTY 电话服务以供耳聋或有听力障碍的客户用于访问销售和支持服务：

TTY 服务
800-IBM-3383 (800-426-3383)
（北美）

有关 IBM 对辅助功能所作承诺的更多信息，请参阅 IBM Accessibility (www.ibm.com/able)。

声明

本信息是为在美国国内供应的产品和服务而编写的。 您可以从 IBM 获取此资料的其他语言版本。 但是，您可能需要拥有使用该语言的产品或产品版本的副本，才能对其进行访问。

IBM 可能在其他国家或地区不提供本文档中讨论的产品、服务或功能特性。 有关您所在区域当前可获得的产品和服务的信息，请向您当地的 IBM 代表咨询。 任何对 IBM 产品、程序或服务的引用并非意在明示或默示只能使用 IBM 的产品、程序或服务。 只要不侵犯 IBM 的知识产权，任何同等功能的产品、程序或服务，都可以代替 IBM 产品、程序或服务。 但是，评估和验证任何非 IBM 产品、程序或服务的操作则由用户自行负责。

IBM 可能已拥有或正在申请与本文档内容有关的各项专利。 提供本文档并不意味着授予用户使用这些专利的任何许可。 您可以用书面方式将许可查询寄往：

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
美国

有关双字节字符集 (DBCS) 信息的许可查询，请与您所在国家或地区的 IBM 知识产权部门联系，或用书面方式将查询寄往：

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“按现状”提供本出版物，不附有任何种类的（无论是明示的还是默示的）保证，包括但不限于默示的有关不侵权、适销和适用于某特定用途的保证。 有些管辖区域在某些交易中不允许免除明示或默示的保证。因此本条款可能不适用于您。

本信息可能包含技术方面不够准确的地方或印刷错误。 本信息将定期更改；这些更改将编入本信息的新版本中。 IBM 可以随时对本出版物中描述的产品和/或程序进行改进和/或更改，而不另行通知。

本信息中对非 IBM Web 站点的任何引用都只是为了方便起见才提供的，不以任何方式充当对那些 Web 站点的保证。 那些 Web 站点中的资料不是 IBM 产品资料的一部分，使用那些 Web 站点带来的风险将由您自行承担。

IBM 可以按它认为适当的任何方式使用或分发您所提供的任何信息而无须对您承担任何责任。

本程序的被许可方如果要了解有关程序的信息以达到如下目的：（i）允许在独立创建的程序和其他程序（包括本程序）之间进行信息交换，以及（ii）允许对已经交换的信息进行相互使用，请与下列地址联系：

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
美国

只要遵守适当的条款和条件，包括某些情形下的一定数量的付费，就可获得这方面的信息。

本文档中描述的许可程序及其所有可用的许可资料均由 IBM 依据 IBM 客户协议、IBM 国际程序许可协议或任何同等协议中的条款提供。

此处讨论的性能数据是在特定运行条件下衍生出来的。 实际结果可能会有差异。

涉及非 IBM 产品的信息可从这些产品的供应商、其出版说明或其他可公开获得的资料中获取。 IBM 没有对这些产品进行测试，也无法确认其性能的精确性、兼容性或任何其他关于非 IBM 产品的声明。 有关非 IBM 产品的功能的问题应询问这些产品的供应商。

此信息包含了天常商业操作中使用的数据和报告的示例。 要尽可能完整地对它们进行说明，这些示例应包括个人、公司、品牌和产品的名称。 所有这些名称都是虚构的，如有任何与实际商业企业使用的名称和地址类似之处，则纯属巧合。

版权许可证：

本信息包含源语言形式的样本应用程序，用以阐明在不同操作平台上的编程技术。 如果是为按照在编写样本程序的操作平台上的应用程序编程接口（API）进行应用程序的开发、使用、经销或分发为目的，您可以任何形式对这些样本程序进行复制、修改、分发，而无须向 IBM 付费。 这些示例尚未在所有条件下经过全面测试。 因此，IBM 不能担保或默示这些程序的可靠性、可维护性或功能。 这些实例程序“按现状”提供，不附有任何种类的保证。 对于因使用样本程序所引起的任何损害，IBM 概不负责。

凡这些样本程序的每份拷贝或其任何部分或任何演绎作品，都必须包括如下版权声明：© （贵公司的名称）（年）。 此部分代码是根据 IBM 公司的样本程序衍生出来的。 © Copyright IBM Corp.（输入年份）.

商标

IBM、IBM 徽标和 ibm.com® 是 International Business Machines Corp., 在全球许多管辖区域的商标或注册商标。 其他产品和服务名称可能是 IBM 或其他公司的商标。 Web 页面“Copyright and trademark information”(www.ibm.com/legal/copytrade.shtml) 提供了 IBM 商标的最新列表。

Adobe 是 Adobe Systems Incorporated 在美国和/或其他国家或地区的注册商标。

Linear Tape-Open、LTO 和 Ultrium 是 HP、IBM Corp 和 Quantum 在美国和其他国家或地区的商标。

Intel 和 Itanium 是 Intel Corporation 或其子公司在美国和其他国家或地区的商标或注册商标。

注册商标 Linux 是依据 Linux Foundation 的子许可证来使用的，Linux Foundation 是此标记的全球所有者 Linus Torvalds 的独家被许可方。

Microsoft、Windows 和 Windows NT 是 Microsoft Corporation 在美国和/或其他国家或地区的商标。

Java™ 和所有基于 Java 的商标和徽标是 Oracle 和/或其子公司的商标或注册商标。

Red Hat®、OpenShift®、Ansible® 和 Ceph® 是 Red Hat, Inc. 或其子公司在美国和其他国家或地区的商标或注册商标。

UNIX 是 The Open Group 在美国和其他国家或地区的注册商标。

VMware、VMware vCenter Server 和 VMware vSphere 是 VMware, Inc. 或其子公司在美国和/或其他管辖区域的注册商标或商标。

产品文档的条款和条件

只要遵守下列条款和条件，即授予您对这些出版物的使用权限。

适用性
这些条款和条件是有关 IBM Web 站点使用的任何条款的补充。

个人使用
您可以为了个人使用而非商业性使用复制这些出版物，但前提是保留所有专有权声明。 未经 IBM 明确许可，不能分发或展示这些出版物或其中任何部分，也不能制作其衍生产品。

商业性使用
您仅可在贵公司内部复制、分发和显示这些出版物，但前提是保留所有专有权声明。 未经 IBM 的明确许可，您不得制作这些出版物的演绎作品，也不得在贵公司外部复制、分发或显示这些出版物或其部分出版物。

权利
除非本许可权中明确授予，否则不得授予对这些出版物或其中包含的任何信息、数据、软件或其他知识产权的任何许可权、许可证或权利，无论明示的还是默示的。

只要 IBM 认为这些出版物的使用会损害其利益或者 IBM 判定未正确遵守上述指示信息，IBM 将有权撤销本文授予的许可权。

您不可以下载、出口或再出口本信息，除非完全遵守所有适用的法律和法规，包括所有美国出口法律和法规。

IBM 对于这些出版物的内容不作任何保证。 这些出版物“按现状”提供，不附有任何种类的（无论是明示的还是默示的）保证，包括但不限于默示的有关适销、不侵权和适用于某种特定用途的保证。

隐私策略注意事项

包括用作服务解决方案的软件在内的 IBM 软件产品（“软件产品服务”）可能会使用 Cookie 或其他技术来收集产品使用情况信息，以帮助改善最终用户体验、定制与最终用户的交互或者用于其他目的。 在许多情况下，“软件产品”不会收集任何个人可标识信息。 我们的某些软件产品可以帮助您收集个人可标识信息。 如果此“软件产品”使用 cookie 来收集个人可标识信息，那么有关此产品使用 cookie 的特定信息将在下面予以阐述。

此软件产品不使用 cookie 或其他技术来收集个人可标识信息。

如果为此软件产品部署的配置向您（作为客户）提供了通过 cookie 和其他技术从最终用户收集个人可标识信息的能力，那么您应该向自己的法律顾问咨询适用于此类数据收集的任何法律（包括通知和许可的任何需求）。

有关将各种技术（包括 cookie）用于这些目的的更多消息，请参阅 IBM 隐私策略 (http://www.ibm.com/privacy)、IBM 在线隐私声明 (http://www.ibm.com/privacy/details) 中标题为“Cookies, Web Beacons and Other Technologies”的部分以及“ IBM Software Products and Software-as-a-Service Privacy Statement”(http://www.ibm.com/software/info/product-privacy)。

词汇表

词汇表提供 IBM Spectrum Protect 产品系列的术语和定义。

请参阅 IBM Spectrum Protect 词汇表。

索引

[G]

[H]

[J]

[K]

[L]

[M]

[Z]

[数字]

A

C

D

E

F

H

I

K

L

M

O

P

Q

R

S

T

U

W

程序号： 5725-W98
5725-W99
5725-X15