IBM Spectrum Protect
Versión 8.1.0

Utilización de la interfaz de programación de aplicaciones

IBM Spectrum Protect
Versión 8.1.0

Utilización de la interfaz de programación de aplicaciones

Nota:
Antes de utilizar esta información y el producto al que da soporte, lea la información del apartado “Avisos” en la página 219.

Esta edición se aplica a la versión 8, release 1, modificación 0 de IBM Spectrum Protect (números de producto 5725-W98, 5725-W99 y 5725-X15) y a todos los releases y modificaciones posteriores mientras no se indique lo contrario en nuevas ediciones.

Contenido

Acerca de esta publicación v
Acerca de esta publicación. v
Publicaciones v
Convenios utilizados en esta publicación v

Novedades de la versión 8.1.0 vii

Capítulo 1. Visión general de la API . . . 1
Conceptos básicos de los archivos de opciones y configuración 1
Configuración del entorno de la API 3

Capítulo 2. Crear y ejecutar la aplicación de API de ejemplo 5
Archivos de origen de aplicación de ejemplo UNIX o Linux. 5
Compilación de la aplicación de ejemplo de UNIX o Linux 6
Aplicación de ejemplo de 64 bits de Windows . . . 7

Capítulo 3. Consideraciones para el diseño de una aplicación 11
Determinar los límites de tamaño 14
Mantenimiento del control de la versión de la API 14
Utilizar varias hebras 16
Señales y manejadores de señal. 16
Iniciar o finalizar una sesión. 17
Seguridad de sesión 18
Configuración de la opción `passwordaccess` a `generate` sin TCA 21
Crear un usuario administrativo con autorización de propietario de cliente 22
ID y nombres de objeto 23
Nombre de espacio de archivo 24
Nombres de alto y bajo rango 24
Tipo de objeto 25
Acceder a objetos como propietario de sesión . . . 25
Acceder a objetos de nodos y propietarios 26
Gestionar espacios de archivos 26
Asociar objetos con clases de gestión 29
Clases de gestión de consulta 30
Retener y liberar la caducidad/eliminación 31
Protección de retención de datos archivados . . 32
Consultas para el sistema IBM Spectrum Protect . . 34
Ejemplo de consultas del sistema 35
Eficiencia del servidor 36
Enviar datos a un servidor 37
El modelo de transacción 37
Agregación de archivo. 38
Transferencia de datos sin LAN. 38
Operaciones de grabación simultánea. 39
Mejora del rendimiento de la API 39
Configuración de la API para enviar datos de rendimiento al supervisor de rendimiento de cliente. 40
Configuración de las opciones del supervisor de rendimiento de cliente. 40
Enviar objetos al servidor. 42
Conceptos básicos sobre archivar y realizar copias de seguridad de los objetos. 43
Compresión 44
Eliminación de copia de almacenamiento intermedio. 46
Cifrado de la API 49
Desduplicación de datos 52
Eliminación de datos duplicados del lado del cliente de la API. 54
Eliminación de datos duplicados del lado del servidor 58
Migración tras error de la aplicación 59
Información de estado de migración tras error. . 59
Diagramas de ejemplo de copia de seguridad/archivado 62
Ejemplo de código de funciones de la API que envían datos al almacenamiento de IBM Spectrum Protect storage 64
Agrupación de archivos 66
Recibir datos de un servidor. 68
Restauración o recuperación de objeto parcial . . 69
Restaurar o recuperar datos 69
Diagramas de ejemplo de restauración y recuperación 74
Ejemplo de código de recepción de datos de un servidor 75
Actualizar y eliminar objetos en el servidor. . . . 76
Eliminar objetos del servidor 77
Registro de eventos. 77
Resumen del diagrama de estado para la API de IBM Spectrum Protect 78

Capítulo 4. Conceptos básicos sobre la interoperatividad 81
Interoperatividad del cliente de archivado/copia de seguridad 81
Asignar nombres a los objetos de la API. . . . 81
Mandatos del cliente de copia de seguridad/archivado que se pueden utilizar con la API 83
Interoperatividad del sistema operativo 84
Soporte de copia de seguridad de varios nodos con el proxy de nodo de cliente 85

Capítulo 5. Utilización de la API con Unicode 87
Cuándo se utiliza Unicode 87
Configuración de Unicode 87

Capítulo 6. Llamadas a función de la API 91
dsmBeginGetData 94
dsmBeginQuery 95
dsmBeginTxn 100
dsmBindMC 101
dsmChangePW 102
dsmCleanUp 103
dsmDeleteAccess 104
dsmDeleteFS 104
dsmDeleteObj 105
dsmEndGetData 107
dsmEndGetDataEx 107
dsmEndGetObj 108
dsmEndQuery 108
dsmEndSendObj 109
dsmEndSendObjEx 109
dsmEndTxn 110
dsmEndTxnEx 111
dsmGetData 113
dsmGetBufferData 114
dsmGetNextQObj 115
dsmGetObj 118
dsmGroupHandler 119
dsmInit 120
`dsmInitEx` 124
dsmLogEvent 128
dsmLogEventEx 129
dsmQueryAccess 130
dsmQueryApiVersion 131
dsmQueryApiVersionEx 131
dsmQueryCliOptions 132
dsmQuerySessInfo 133
dsmQuerySessOptions 134
dsmRCMsg 135
dsmRegisterFS 136
dsmReleaseBuffer 137
dsmRenameObj 137
dsmRequestBuffer 139
dsmRetentionEvent 140
dsmSendBufferData 141
dsmSendData 142
dsmSendObj 143
dsmSetAccess 147
dsmSetUp 148
dsmTerminate 150
dsmUpdateFS 150
dsmUpdateObj 151
dsmUpdateObjEx 152

Apéndice A. Archivo de origen de códigos de retorno de la API: `dsmrc.h` . 155

Apéndice B. Archivos de origen de definiciones de tipo de API 165

Apéndice C. Archivo de origen de las definiciones de la función de la API. . 207

Apéndice D. Funciones de accesibilidad para la familia de productos IBM Spectrum Protect . . . 217

Avisos 219

Glosario 223

Índice. 225

Acerca de esta publicación

Esta publicación proporciona información que le ayudará a realizar las tareas siguientes:

- Agregar IBM Spectrum Protect llamadas de interfaz de programa de aplicación a una aplicación existente
- Grabar programas con interfaces de programa de uso general que obtienen los servicios de IBM Spectrum Protect.

Además de la interfaz de programas de aplicación (API), se incluyen los siguientes programas en varios sistemas operativos:

- El programa cliente de copia de seguridad/archivado permite a los usuarios realizar copias de seguridad de archivos o archivarlos desde las estaciones de trabajo o servidores de archivos y restaurar y recuperar versiones de copia de seguridad y copias archivadas de archivos en las estaciones de trabajo locales.
- Un cliente de copia de seguridad/archivado que permite que un administrador autorizado, el personal del Help Desk o un usuario final realicen copias de seguridad, restauren, archiven y recuperen servicios mediante un navegador Web en un sistema remoto.
- Un programa cliente de administración al que puede acceder desde un navegador Web o la línea de mandatos. Un administrador controla y supervisa las actividades del servidor, define políticas de gestión de almacenamiento para copias de seguridad, de archivado y gestión de espacio, y establece planificaciones para llevar a cabo estos servicios en intervalos regulares.

Acerca de esta publicación

Esta publicación proporciona instrucciones para añadir llamadas de la API a una aplicación existente. Debería familiarizarse con el lenguaje de programación C y las funciones de IBM Spectrum Protect.

Publicaciones

La familia de productos IBM Spectrum Protect incluye IBM Spectrum Protect Snapshot, IBM Spectrum Protect for Space Management, IBM Spectrum Protect for Databases y otros productos de gestión de almacenamiento de IBM®.

Para ver la documentación de producto de IBM, consulte IBM Knowledge Center.

Convenios utilizados en esta publicación

Esta publicación utiliza los convenios tipográficos siguientes:

Ejemplo	Descripción
autoexec.ncf hsmgui.exe	Cadena de letras minúsculas con una extensión que indica nombres de archivos de programa.
DSMI_DIR	Cadena de letras mayúsculas que indica códigos de retorno y otros valores.
dsmQuerySessInfo	La letra negrita indica un comando que se especifica en una línea de comandos, el nombre de una llamada a función, el nombre de una estructura, un campo dentro de una estructura o un parámetro.

Ejemplo	Descripción
timeformat	El tipo de letra cursiva y negrita indica una opción de cliente de copia de seguridad/archivado. El tipo de letra negrita se utiliza para especificar la opción o bien en un ejemplo.
dateformat	El tipo de letra cursiva y negrita indica una opción, el valor de una opción, un nuevo término, una información que el usuario proporciona o para hacer hincapié en determinadas palabras del texto.
`maxcmdretries`	El tipo de letra monoespaciado indica fragmentos de un programa o información tal como se visualizaría en pantalla, como un ejemplo de comando.
signo más (+)	Un signo más entre dos teclas indica que se pulsan ambas teclas al mismo tiempo.

Novedades de la versión 8.1.0

IBM Spectrum Protect versión 8.1.0 presenta introduces nuevas características y actualizaciones.

Si desea una lista de características nuevas y actualizaciones en este release, consulte Actualizaciones de API.

Capítulo 1. Visión general de la API

La interfaz de programa de aplicación (API) de IBM Spectrum Protect permite a un cliente de aplicación utilizar las funciones de gestión de almacenamiento.

La API incluye llamadas de función que pueden utilizarse en una aplicación para realizar las operaciones siguientes:
- Iniciar o finalizar una sesión
- Asignar clases de gestión a objetos antes de almacenarse en un servidor
- Realizar copias de seguridad o archivado de objetos en un servidor
- Restaurar o recuperar objetos de un servidor
- Consultar al servidor acerca de información sobre objetos almacenados
- Gestionar espacios de archivos
- Enviar eventos de retención

Cuando un desarrollador de aplicación instala la API, recibe los archivos que necesita un usuario final de una aplicación:
- La biblioteca compartida de la API.
- El archivo de mensajes.
- Los archivos de opciones de cliente de ejemplo.
- El código de origen de los archivos del encabezado de la API que necesita la aplicación.
- El código de origen de una aplicación de ejemplo, y el archivo para construirlo.
- El archivo dsmtca (sólo UNIX y Linux) .

Para restaurar el registro de Las compilaciones de todas las aplicaciones de 64 bits deben ejecutarse utilizando opciones de compilador compatibles con 64 bits. Por ejemplo, debe utilizarse '-q64' cuando se construyen aplicaciones API en AIX, y '-m64' en Linux. Consulte los archivos de construcción de ejemplo para obtener más información.

Importante: Cuando instala la API, asegúrese de que todos los archivos están en el mismo nivel.

Para obtener información acerca de cómo instalar la API, consulte Instalación de los clientes de copia de seguridad y archivado de IBM Spectrum Protect.

Las referencias a UNIX y Linux incluyen sistemas operativos AIX, HP-UX, Linux, Mac OS X y Oracle Solaris.

Conceptos básicos de los archivos de opciones y configuración

Los archivos de opciones y configuración definen las condiciones y límites bajo las cuales se ejecuta la sesión.

Un administrador o un usuario final pueden configurar los valores de las opciones para:
- Configurar la conexión a un servidor
- Controlar qué objetos se envían al servidor y con qué clase de gestión se asocian

Las opciones se definen en uno o dos archivos cuando instala la API en la estación de trabajo.

En sistemas operativos UNIX y Linux, las opciones residen en dos archivos de opciones:
- dsm.opt - el archivo de opciones del cliente
- dsm.sys - el archivo de opciones de sistema del cliente

En otros sistemas operativos, el archivo de opciones del cliente (dsm.opt) contiene todas las opciones.

Restricción: La API no soporta las siguientes opciones de cliente de archivado y copia de seguridad:
- `autofsrename`
- `changingretries`
- `domain`
- `eventlogging`
- `groups`
- `subdir`
- `users`
- `virtualmountpoint`

También puede especificar opciones en la llamada de función **dsmInitEx**. Utilice el parámetro de la cadena de opciones o el parámetro del archivo de configuración de la API.

La misma opción puede derivar de más de un origen de configuración. Cuando esto ocurre, el origen con la prioridad más alta tiene preferencia. Tabla 1 lista la secuencia de prioridad.

Tabla 1. Configurar orígenes en orden de prioridad descendente

Prioridad	UNIX y Linux	Windows	Descripción
1	archivo dsm.sys (opciones del sistema del cliente)	no aplicable	Este archivo contiene opciones que un administrador del sistema solo establece para UNIX y Linux. **Consejo:** Si el archivo dsm.sys contiene stanzas de servidor, asegúrese de que la opción `passwordaccess` especifica el mismo valor (`prompt` o `generate`) en cada una de las stanzas.
2	Serie de opciones (opciones de cliente)	Serie de opciones (todas las opciones)	Una de estas opciones surte efecto cuando pasa como un parámetro a una llamada **dsmInitEx**. La lista puede contener opciones como, por ejemplo, `compressalways`, `servername` (solo UNIX y Linux), o `tcpserveraddr` (no UNIX). Con la cadena de opción de la API, un cliente de la aplicación puede realizar cambios a los valores de la opción en el campo de configuración de la API y en el archivo de opciones del cliente. Por ejemplo, es posible que la aplicación consulte al usuario final si se requiere compresión. Dependiendo de las respuestas del usuario, puede crear una cadena de opciones de la API con esta opción y pasarla a la llamada **dsmInitEx**. Para obtener más información sobre el formato de la cadena de opción de la API, consulte "**dsmInitEx**" en la página 124. También puede establecer este parámetro en `NULO`. Esto indica que no existe una cadena de opciones de la API en esta sesión.

Tabla 1. Configurar orígenes en orden de prioridad descendente (continuación)

Prioridad	UNIX y Linux	Windows	Descripción
3	Archivo de configuración de la API (opciones de cliente)	Archivo de configuración de la API (todas las opciones)	Los valores que establezca en el archivo de configuración de API sustituyen a los valores que haya establecido en el archivo de opciones de cliente. Establezca las opciones en el archivo de configuración de API con valores que sean los adecuados en la sesión de IBM Spectrum Protect para el usuario. Los valores surten efecto cuando el nombre del archivo de configuración de la API pasa como un parámetro en la llamada **dsmInitEx**. También puede configurar este parámetro en NULO. Esto indica que no existe un archivo de configuración de la API en esta sesión.
4	dsm.opt, archivo (opciones de cliente)	dsm.opt, archivo (todas las opciones)	En sistemas operativos UNIX y Linux el archivo dsm.opt contiene únicamente las opciones del usuario. En otros sistemas operativos, el archivo dsm.opt contiene todas las opciones. Para omitir las opciones en estos campos, siga los métodos que se describen en esta tabla.

Conceptos relacionados:

Opciones de proceso

Configuración del entorno de la API

La API utiliza variables de entorno exclusivas para localizar archivos. Es posible utilizar diferentes archivos para las aplicaciones de la API de los que utiliza el cliente de archivado/copia de seguridad. Las aplicaciones pueden utilizar la llamada de función **dsmSetup** para modificar los valores que las variables de entorno configura.

Consejo: En Windows, el directorio de la instalación predeterminado es `%SystemDrive%\Archivos de programa\Common Files\Tivoli\TSM\api`

Tabla 2 incluye una lista de las variables del entorno de la API por sistema operativo.

Tabla 2. Variables de entorno de la API

Variables	UNIX and Linux	Windows
DSMI_CONFIG	El nombre calificado al completo para el archivo de opciones del cliente (dsm.opt).	El nombre calificado al completo para el archivo de opciones del cliente (dsm.opt).
DSMI_DIR	Apunta a la ruta que contiene el dsm.sys, dsmtca, subdirectorio en_US, y otro idioma nacional compatible (NLS). El subdirectorio en_US debe contener dsmclientV3.cat.	Apunta a la ruta que contiene dscenu.txt y cualquier archivo de mensaje NLS.

Tabla 2. Variables de entorno de la API (continuación)

Variables	UNIX and Linux	Windows
DSMI_LOG	Apunta a la ruta del archivo dsierror.log.	Apunta a la ruta del archivo dsierror.log. Si se define la opción del cliente**errorlogname**, la ubicación que especifica la opción sustituye al directorio que especifica el DSMI_LOG.

Capítulo 2. Crear y ejecutar la aplicación de API de ejemplo

El paquete de la API incluye las aplicaciones de ejemplo que demuestran el contexto de las llamadas de función de la API. Instale una aplicación de ejemplo y revise el código de origen para entender cómo utilizar las llamadas de función.

Seleccione uno de los siguientes paquetes de aplicación API de ejemplo:
- El interactivo, paquete de aplicación de hebra única (dapi*)
- El paquete de aplicación de varias hebras (callmt*)
- La aplicación de prueba de agrupación de objetos lógica (dsmgrp*)
- La aplicación de ejemplo de política de retención basada en evento (callevnt)
- La aplicación de ejemplo de retención de eliminación (callhold)
- La aplicación de ejemplo de protección de retención de datos (callret)
- El programa de ejemplo de almacenamiento intermedio de datos de IBM Spectrum Protect (callbuff)

Como ayuda para la iniciación, revise el procedimiento para crear la aplicación de ejemplo dapismp en su plataforma:
- Para aplicaciones UNIX o Linux , consulte "Archivos de origen de aplicación de ejemplo UNIX o Linux".
- Para aplicaciones Windows, consulte "Aplicación de ejemplo de 64 bits de Windows" en la página 7.

La aplicación de ejemplo dapismp crea su propia corriente de datos cuando archiva o realiza copias de seguridad de los objetos. No lee o graba objetos al sistema de archivos del disco local. El nombre del objeto no se corresponde con ningún archivo de la estación de trabajo. La "seed string" que emite genera un patrón que puede comprobarse cuando el objeto se restaura o recupera. Una vez se ha compilado la aplicación de ejemplo y se ha ejecutado **dapismp** para iniciarlas, puede seguir las instrucciones que aparecen en la pantalla.

Archivos de origen de aplicación de ejemplo UNIX o Linux

Para construir y ejecutar la aplicación de ejemplo de UNIX o Linux, debe asegurarse de que cuenta con ciertos archivos de origen. Una vez que ha construido la aplicación de ejemplo, puede compilarla y ejecutarla.

Los archivos que aparecen en una lista en Tabla 3 incluyen los archivos de origen y otros archivos que necesita para construir la aplicación de ejemplo que se incluye en el paquete de la API.

Tabla 3. Archivos que necesita para construir la aplicación de ejemplo de API para UNIX o Linux

Nombres de archivos	Descripción
README_api_enu	archivo README

Tabla 3. Archivos que necesita para construir la aplicación de ejemplo de API para UNIX o Linux (continuación)

Nombres de archivos		Descripción
dsmrc.h dsmapitd.h dsmapips.h dsmapifp.h release.h		Archivo de encabezado de los códigos de retorno Archivo de encabezado de definiciones de tipo corriente Archivo de encabezado de definiciones de tipo específico del sistema operativo Archivo de encabezado de prototipo de función Archivo de encabezado de valores de release
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	Módulos para la aplicación de ejemplo de línea de mandato
makesmp[64].*xxx*		Archivo para construir dapismp del sistema operativo. *xxx* indica el sistema operativo.
callmt1.c callmt2.c		Archivos de ejemplo de varias hebras
callmtu1.c callmtu2.c		Archivos de ejemplo Unicode de varias hebras
libApiDS.xx libApiDS64.xx, o libApiTSM64.xx		Biblioteca compartida (el sufijo depende de la plataforma)
dsmgrp.c callevnt.c callhold.c callret.c callbuff.c dpsthread.c		Agrupar archivos de ejemplo Código de origen de ejemplo de política de retención basada en evento Código de origen de ejemplo de retención de eliminación Código de origen de ejemplo de protección de retención de datos

Compilación de la aplicación de ejemplo de UNIX o Linux

Compile la aplicación de la API de ejemplo **dapismp** utilizando un compilador para el sistema operativo.

Debe instalar los siguientes compiladores para compilar la aplicación de ejemplo de API UNIX o Linux :

- IBM AIX: compilador IBM Visual Age versión 6 o posterior
- Compilador HP-IA64 - aCC A.05.50 o superior
- Compilador Linux - GCC versión 3.3.3 o posterior
- Compilador Mac OS X - GCC versión 4.0 o posterior
- Compilador Oracle Solaris - Oracle Studio C++ versión 11 o posterior

1. Para compilar los ejemplos de la API, ejecute el siguiente mandato:

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

   Donde *xxx* indica el sistema operativo.

2. Después de construir los ejemplos, configure las variables de entorno, incluyendo la variable de entorno `DSMI_DIR`, y los archivos de opciones. Para obtener más información, consulte “Conceptos básicos de los archivos de opciones y configuración” en la página 1.
3. La primera vez que inicia sesión, hágalo como usuario root para registrar la contraseña.

   Consejo: Al configurar la opción `compressalways` en `no` es posible que no se vuelva a enviar un objeto sin comprimir. Este comportamiento depende de la funcionalidad de la aplicación.

   Para especificar el método de comunicación de memoria compartida en AIX, el usuario cliente de APIIBM Spectrum Protect debe cumplir una de las siguientes condiciones:

   - Debe haber iniciado sesión como usuario root.
   - Debe tener el mismo UID que el proceso que está ejecutando el servidor de IBM Spectrum Protect.

   Esta restricción no es pertinente si la opción `passwordaccess` está establecida en `generate` en el archivo de opciones del sistema de cliente `dsm.sys` y el TCA se está utilizando o si modifica los permisos del archivo de programa de la aplicación con los siguientes mandatos:

   ```
   chown root.system your_api_program
   chown u+s your_api_program
   ```

   Para obtener más información, consulte la documentación del programa de aplicación.
4. Ejecute el mandato **dapismp** para iniciar la aplicación.
5. Seleccione de la lista de opciones que se visualizan. Asegúrese de que ejecuta la acción de inicio de sesión antes de ejecutar cualquier otra acción.

   Requisito: Siempre prefije el espacio de archivo, los nombres de alto y bajo rango con el delimitador de ruta correcto (/) cuando introduzca el nombre, por ejemplo: `/myfilespace`. Debe utilizar este prefijo incluso cuando especifique el carácter comodín asterisco (*).

Conceptos relacionados:

Variables de entorno (sistemas UNIX y Linux)

Aplicación de ejemplo de 64 bits de Windows

Para crear y ejecutar la aplicación de ejemplo para sistemas Microsoft Windows de 64 bits, debe instalar la API de IBM Spectrum Protect y asegurarse de tener determinados archivos de origen.

Restricciones:

- Para obtener mejores resultados, utilice la carga dinámica. Por ejemplo, consulte el archivo `dynaload.c` y la implementación en el código de muestra.
- Los archivos para la aplicación de ejemplo están en los siguientes directorios:

  api64\obj
  Contiene los archivos de objeto del programa de ejemplo de la API.

  api64\samprun
  Contiene el programa de ejemplo **dapismp**. El programa de ejemplo contiene el directorio de ejecución.

- El DLL tsmapi64.dll es un DLL de 64 bits.
- Utilice el Microsoft C/C++ Compiler versión 15 y el archivo make `makesmp64.mak` para compilar la aplicación de ejemplo de API **`dapismp`**. Es posible que tenga que ajustar los archivos make para su entorno, especialmente, la biblioteca o los directorios de inclusión.
- Después de compilar la aplicación, ejecute la aplicación de muestra emitiendo el mandato **`dapismp`** desde el directorio `api64\samprun`.
- Seleccione de la lista de opciones que se visualizan. Asegúrese de que ejecuta la acción de inicio de sesión antes de ejecutar cualquier otra acción.
- Siempre prefije el espacio de archivo, los nombres de alto y bajo rango con el delimitador de ruta correcto (\) cuando lo introduzca, por ejemplo:`\myfilespace`. Debe utilizar este prefijo incluso cuando especifique el carácter comodín asterisco (*).

En sistemas operativos Windows , los archivos de origen que debe tener para compilar la aplicación de ejemplo están listados en Tabla 4. La aplicación de ejemplo está incluida en el paquete de API. Para su comodidad, un archivo ejecutable precompilado (dapismp.exe) también se incluye.

Tabla 4. Archivos para compilar la aplicación de ejemplo de API de Windows 64 bits

Nombres de archivos	Descripción
`api.txt`	archivo README
`tsmapi64.dll`	DLL de la API

`dsmrc.h`
`dsmapitd.h`
`dsmapips.h`
`dsmapifp.h`
`dsmapidl.h`
`release.h`
	Archivo de encabezado de códigos de retorno
	Archivo de encabezado de definiciones de tipo corriente
	Archivo de encabezado de definiciones de tipo especifico del sistema operativo
	Archivo de encabezado de prototipo de función
	Archivo de encabezado de prototipo de función cargado de forma dinámica
	Archivo de encabezado de valores de release

`dapidata.h`
`dapint64.h`
`dapitype.h`
`dapiutil.h`
	Archivos de encabezado de Código fuente

`tsmapi64.lib`	Biblioteca implícita

`dapibkup.c`
`dapiinit.c`
`dapint64.c`
`dapipref.c`
`dapiproc.c`
`dapiproc.h`
`dapipw.c`
`dapiqry.c`
`dapirc.c`
`dapismp64.c`
`dapiutil.c`
`dynaload.c`
	Archivos de código fuente de dapismp.exe

`makesmpx64.mak` (Windows x64) `makesmp64.mak` (Windows IA64)	Archivos de construcción para la aplicación de ejemplo

Tabla 4. Archivos para compilar la aplicación de ejemplo de API de Windows 64 bits (continuación)

Nombres de archivos	Descripción

`callmt1.c`
`callmt2.c`
`callmtu164.c`
`callmtu264.c`
	Archivos de ejemplo de varias hebras

`dpsthread.c`	Código de origen de archivo de ejemplo
`callevnt.c`	Código de origen de política de retención basada en evento
`callhold.c`	Código de origen de ejemplo de retención de eliminación
`callret.c`	Código de origen de ejemplo de protección de retención de datos
`callbuff.c`	Código de origen de ejemplo de almacenamiento intermedio compartido (no de copia).

Capítulo 3. Consideraciones para el diseño de una aplicación

Cuando diseña una aplicación, tiene que tener una gran comprensión de muchos aspectos de la API.

Para logar esa comprensión de la API, revise los siguientes temas:

- "Determinar los límites de tamaño" en la página 14
- "Mantenimiento del control de la versión de la API" en la página 14
- "Utilizar varias hebras" en la página 16
- "Señales y manejadores de señal" en la página 16
- "Iniciar o finalizar una sesión" en la página 17
- "ID y nombres de objeto" en la página 23
- "Configuración de la opción `passwordaccess` a `generate` sin TCA" en la página 21
- "Acceder a objetos como propietario de sesión" en la página 25
- "Acceder a objetos de nodos y propietarios" en la página 26
- "Gestionar espacios de archivos" en la página 26
- "Asociar objetos con clases de gestión" en la página 29
- "Retener y liberar la caducidad/eliminación" en la página 31
- "Consultas para el sistema IBM Spectrum Protect" en la página 34
- "Enviar datos a un servidor" en la página 37
- "Diagramas de ejemplo de copia de seguridad/archivado" en la página 62
- "Agrupación de archivos" en la página 66
- "Resumen del diagrama de estado para la API de IBM Spectrum Protect" en la página 78

Cuando diseñe la aplicación, revise las consideraciones en Tabla 5. Las estructuras de inicio con los campos **memset** pueden cambiar en los releases posteriores. El valor `stVersion` aumenta con cada mejora del producto.

Tabla 5. Consideraciones de la API para el diseño de una aplicación

Elementos de diseño	Consideraciones
Configurar entorno local	La aplicación debe establecer el entorno local antes de llamar a la API. Para establecer el entorno local al valor predeterminado, añada el siguiente código a la aplicación: `setlocale(LC_ALL,"");` Para configurar el entorno local en otro valor, utilice la misma llamada con el entorno local apropiado en el segundo parámetro. Busque información específica en la documentación relativa al sistema operativo que utilice.

Tabla 5. Consideraciones de la API para el diseño de una aplicación (continuación)

Elementos de diseño	Consideraciones

Control de sesión	Aplique las siguientes directrices al control de sesión:
	• Asigne un nombre de nodo exclusivo para cada cliente de archivado y copia de seguridad de IBM Spectrum Protect y producto del cliente de la API de IBM Spectrum Protect que utilice. Los siguientes productos son ejemplos de estos clientes:
	– IBM Spectrum Protect for Mail
	– o IBM Spectrum Protect HSM for Windows
	• Utilice un nombre de propietario coherente en un procedimiento de restauración y copia de seguridad.
	• Utilice la opción `passwordaccess` para gestionar el acceso al archivo de contraseña protegida. Esta opción afecta al uso del proceso hijo TCA en UNIX y Linux únicamente, para el nombre de nodo, nombre de propietario de sesión y gestión de contraseña.
	• Asegúrese de que las sesiones de los movimientos de datos terminan cuando se completa la tarea para que los dispositivos del servidor se liberen y se puedan utilizar en otras sesiones.
	• Para permitir una transferencia de datos sin LAN, utilice la llamada a función **dsmSetup** con el indicador `multithread` establecido en activado.
	• En AIX, cuando utiliza aplicaciones de varias hebras o sin LAN, especialmente cuando se ejecutan máquinas con varios procesadores, configure la variable de entornoAIXTHREAD_SCOPE en S en el entorno antes de iniciar la aplicación, para que el rendimiento y la planificación sean mejores. Por ejemplo:
	`EXPORT AIXTHREAD_SCOPE=S`
	Estableciendo `AIXTHREAD_SCOPE` en `S`, las hebras de usuario creadas con los atributos predeterminados se colocan en el alcance de contención del sistema. Si una hebra de usuario se crea con un alcance de contención de sistema, se vincula a una hebra de kernel y es planificada por dicho kernel. La hebra del kernel subyacente no se comparte con otra hebra de usuario. Para obtener más información acerca de esta variable de entorno, consulte el tema siguiente:
	"Utilizar varias hebras" en la página 16
	• Asegúrese de que sólo una hebra de la sesión llama a una función API en cualquier momento. Las aplicaciones que utilizan varias hebras con el mismo manejador de sesión deben sincronizar las llamadas de la API. Por ejemplo, utilice**mutex** con el fin de sincronizar las llamadas de la API:
	getTSMMutex()
	emita la llamada de la API TSM
	releaseTSMMutex()
	Utilice este método sólo cuando las hebras comparten un manejador. Puede utilizar llamadas paralelas para las funciones de la API si las llamadas tienen diferentes manejadores de sesión.
	• Implemente un modelo de productor/consumidor con hebras para el movimiento de datos. Las llamadas de la API son síncronas y las llamadas para la **función dsmGetData** y **función dsmSendData** se bloquean hasta que terminan. Al utilizar un modelo de productor/consumidor, la aplicación puede leer el siguiente almacenamiento intermedio mientras espera a la red. También, esta disociación de lectura/escritura de datos y red, aumenta el rendimiento cuando hay un atasco en la red o retrasos. En general, se espera lo siguiente:
	`Hebra de datos <---> cola de almacenamientos intermedios compartida <---> hebra de comunicación (emiten llamadas a la API de IBM Spectrum Protect)`
	• Utilice la misma sesión para varias operaciones para evitar producir una sobrecarga. Para aplicaciones que tratan con muchos objetos pequeños, implemente la agrupación de sesión para la misma sesión se pueda utilizar en múltiples operaciones pequeñas. La sobrecarga se asocia con la apertura y cierre de una sesión para el servidor IBM Spectrum Protect. La llamada `dsmInit/dsmInitEX` está serializada así que incluso en una aplicación de múltiples hebras sólo se puede iniciar una hebra cada vez. Igualmente, durante el inicio de sesión, la API envía un número de consultas únicas al servidor.

Tabla 5. Consideraciones de la API para el diseño de una aplicación (continuación)

Elementos de diseño	Consideraciones
Secuencia de operación	El servidor IBM Spectrum Protect bloquea las entradas de la base de datos del espacio de archivo durante algunas operaciones. Las siguientes reglas se aplican cuando se están diseñando aplicaciones API de IBM Spectrum Protect : • Las consultas bloquean el espacio de archivo durante toda la transacción. • El bloqueo de consulta se puede compartir con otras operaciones de consulta, con el fin de que varias operaciones de consulta en el mismo espacio de archivo se ejecuten de forma simultánea. • Las siguientes operaciones se utilizan para modificar la base de datos del servidor de IBM Spectrum Protect (**DB Chg**): enviar, obtener, cambiar el nombre, actualizar y suprimir. • La finalización de una operación **DB Chg** requiere un bloqueo de espacio de archivo durante el cambio de la base de datos al final de la transacción. • Se pueden ejecutar a la vez varias operaciones **DB Chg** en el mismo espacio de archivo. Es posible que haya un retraso mientras la secuencia espera al bloqueo al final de la transacción. • El bloqueo de la consulta no se puede compartir con las operaciones **DB Chg**. Una operación **DB Chg** retrasa el principio de una consulta en el mismo espacio de archivo, con lo que las aplicaciones deben separar y serializar consultas desde las operaciones **DB Chg** en el mismo espacio de archivo.
Denominación de objeto	Cuando pone nombre a los objetos, tenga en cuenta los siguientes factores: • Los nombres de objeto específicos son nombres de objeto de alto rango y de bajo rango. Si en el nombre se incluye un identificador único, como el indicador de fecha, entonces las copias de seguridad de los objetos están siempre activas. Los objetos caducan sólo cuando están marcados como inactivo por parte de la llamada a función **dsmDeleteObj**. • El método de restauración para objetos determina cómo dar formato al nombre para consultas fáciles. Si pretende utilizar una restauración de objeto parcial (POR), no es posible utilizar la compresión. Para suprimir la compresión, utilice la función **dsmSendObj objAttr objCompressed=bTrue**.
Agrupación de objetos	Agrupe objetos de manera lógica utilizando espacios de archivos. Un espacio de archivos es un contenedor en el servidor que proporciona una categoría de agrupación para los objetos. La API consulta todos los espacios de archivos durante el inicio de sesión y también durante las consultas, así que el número de espacios de archivos debe estar restringido. Una suposición razonable es que una aplicación establezca 20 - 100 espacios de archivo por nodo. La API puede ofrecer servicios para más espacios de archivo, pero cada espacio de archivo conlleva una sobrecarga para la sesión. Para crear una separación más granular, utilice el objeto `directory` en la aplicación.
Manejo de objeto	No almacene valores `objectID` para utilizarlos para restauraciones futuras. No se garantiza que estos valores sean persistentes durante la vida del objeto. Durante la restauración, preste especial atención al orden de restauración. Después de la consulta, ordene este valor antes de la restauración. Si está utilizando varios tipos de medios de serie, entonces acceda a ellos en sesiones separadas. Para obtener información, consulte el tema siguiente: "Seleccionar y ordenar objetos por orden de restauración" en la página 70
Clase de gestión	Tenga en cuenta cuánto control necesita tener la aplicación sobre la clase de gestión que está asociada con los objetos de aplicación. Es posible definir estancias de inclusión, o bien especificar un nombre en la llamada de función **dsmSendObj**.
Tamaño de objeto	IBM Spectrum Protect necesita saber una estimación de tamaño por cada objeto. Tenga en cuenta cómo calcula la aplicación el tamaño de un objeto. Es mejor la sobrevaloración del tamaño de objeto que la infravaloración.

Determinar los límites de tamaño

Algunas estructuras de datos o campos en la API tienen una limitación de tamaño. Estas estructuras son a menudo nombres u otros campos de texto que no pueden exceder una longitud predeterminada.

Los siguientes campos son ejemplos de estructuras de datos que tienen límites de tamaño:
- Tipo de aplicación
- Descripción de la copia archivada
- Destino de grupo de copia
- Nombre de grupo de copia
- Información de espacio de archivo
- Nombre de clase de gestión
- Nombre de propietario de objeto
- Contraseña

Estos límites están definidos como constantes dentro del archivo de cabecera `dsmapitd.h`. Las asignaciones de almacenamiento deben basarse en estas constantes en lugar de en los números que introduce. Para obtener más información, consulte el apartado Apéndice B, "Archivos de origen de definiciones de tipo de API", en la página 165.

Mantenimiento del control de la versión de la API

Todas las API tiene algún tipo de control de la versión. La versión de API que utiliza en su aplicación debe ser compatible con la versión de la biblioteca de API instalada en la estación de trabajo del usuario.

dsmQueryApiVersionEx debe ser la primera llamada de la API que introduce en el sistema cuando utiliza la API. Esta llamada realiza una de las tareas siguientes:
- Confirma que la biblioteca de la API está instalada y está disponible en el sistema del usuario final
- Devuelve el nivel de la versión de la biblioteca API a la que accede la aplicación

La API es compatible con versiones superiores. Las aplicaciones escritas con releases o versiones antiguas de la biblioteca de la API funcionan correctamente cuando se ejecuta una versión posterior.

Determinar el release de la biblioteca API es muy importante porque es posible que algunos releases tengan requisitos de memoria y definiciones de estructura de datos distintos. La compatibilidad con versiones inferiores es poco probable. Consulte Tabla 6 para obtener más información sobre la plataforma.

Tabla 6. Información sobre la compatibilidad de la plataforma

Plataforma	Descripción
Windows	Los archivos de mensajes deben estar al mismo nivel que los de la biblioteca (DLL). El módulo de Trusted Communication Agent (dsmtca) no se utiliza.
UNIX o Linux	La biblioteca de la API, el módulo de Trusted Communication Agent (dsmtca), y los archivos de mensajes deben estar al mismo nivel.

La llamada **dsmQueryApiVersionEx** devuelve la versión de la biblioteca de la API que está instalada en la estación de trabajo del usuario final. Puede entonces comparar el valor devuelto con la versión de la API que está utilizando el cliente de la aplicación.

El número de la versión de la API del cliente de la aplicación se introduce en el código de objeto compilado como un conjunto de cuatro constantes definidas en dsmapitd.h:

```
DSM_API_VERSION
DSM_API_RELEASE
DSM_API_LEVEL
DSM_API_SUB_LEVEL
```

Consulte el apartado Apéndice B, “Archivos de origen de definiciones de tipo de API”, en la página 165.

La versión de la API del cliente de la aplicación es normalmente inferior o igual que la biblioteca de la API instalada en el sistema del usuario. Tenga cuidado con cualquier otra condición. Puede introducir la llamada **dsmQueryApiVersionEx** en cualquier momento, independientemente de que se haya iniciado la sesión API.

Las estructuras de datos que la API utiliza también tienen información de control de la versión. Las estructuras tienen información de la versión en el primer campo. A medida que las estructuras mejoran, el número de versión aumenta. Cuando inicia el campo de la versión, utilice el valor de la Versión de la estructura definida en dsmapitd.h.

Figura 1 demuestra el tipo de definición de la estructura, **dsmApiVersionEx** del archivo de encabezado, dsmapitd.h. El ejemplo a continuación define una variable global que se denomina ***apiLibVer***. También demuestra cómo utilizarlo en una llamada a **dsmQueryApiVersionEx** con el fin de devolver la versión de la biblioteca de la API del usuario final. Por último, el valor que se devuelve se compara con el número de la versión de la interfaz de programación de aplicaciones (API) de la aplicación.

```
typedef struct
{
        dsUint16_t     stVersion;           /* Structure version               */
        dsUint16_t version;       /* API version                      */
        dsUint16_t release;       /* API release                      */
        dsUint16_t level;         /* API level                        */
        dsUint16_t subLevel;      /* API sub level                    */
} 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);
```

Figura 1. Un ejemplo de cómo obtener el nivel de la versión de la API

Utilizar varias hebras

La API de varias hebras permite a las aplicaciones crear varias sesiones con el servidor IBM Spectrum Protect dentro del mismo proceso. La API se puede introducir de nuevo. Las llamadas pueden ejecutarse en paralelo desde dentro de distintas hebras.

Consejo: Cuando ejecuta una aplicación con una API de varias hebras, utilice la llamada **dsmQueryAPIVersionEx**.

Para ejecutar la API en modo de varias hebras, configure el valor de *mtflag* en DSM_MULTITHREAD de la llamada **dsmSetUp**. La llamada **dsmSetUp** debe ser la primera llamada después de la llamada **dsmQueryAPIVersionEx**. Esta llamada debe volver antes de que ninguna hebra llame a **dsmInitEx**. Cuando todas las hebras han finalizado el proceso, introduzca una llamada a **dsmCleanUp**. El proceso principal no debe finalizar antes de que todas las llamadas finalicen el proceso. Consulte callmt1.c en la aplicación de ejemplo.

Restricción: De forma predeterminada la API es un modo de hebra única. Si una aplicación no llama a **dsmSetUp** con el valor *mtflag* configurado en DSM_MULTITHREAD, la API sólo permite una sesión por proceso.

Una vez que **dsmSetUp** finaliza correctamente, la aplicación puede iniciar varias hebras e introducir varias llamadas **dsmInitEx**. Cada una de las llamadas **dsmInitEx** devuelve un manejador para dicha sesión. Cualquier llamada subsiguiente en dicha hebra para esa sesión debe utilizar ese valor de manejador. Algunos valores son variables de entorno de proceso ancho (valores configurados en **dsmSetUp**). Cada una de las llamadas **dsmInitEx** analiza las opciones de nuevo. Las hebras se pueden ejecutar con opciones diferentes especificando un archivo de sobrescritura o una cadena de opciones en la llamada **dsmInitEx**. Esto capacita a distintas hebras ir a diferentes servidores o bien utilizar nombres de nodo distintos.

Recomendación: En HP, configure la pila de hebra en 64K o en un valor superior. Es posible que el valor predeterminado de la pila de hebras (32K) no sea suficiente

Para permitir que los usuarios de una aplicación tengan una sesión sin LAN, utilice **dsmSetUp** *mtFlag DSM_MULTITHREAD* en la aplicación. Esto es necesario incluso si la aplicación es de una única hebra. Este indicador activa las hebras necesarias para la interfaz IBM Spectrum Protect sin LAN.

Señales y manejadores de señal

La aplicación maneja señales del usuario o del sistema operativo. Si el usuario entra una secuencia de pulsación **CTRL+C** , la aplicación debe alcanzar la señal y enviar llamadas **dsmTerminate** para cada una de las hebras activas. A continuación, llame a **dsmCleanUp** para salir. Si las sesiones no están cerradas adecuadamente, pueden producirse resultados inesperados en el servidor.

La aplicación requiere manejadores de señal, tales como SIGPIPE y SIGUSR1, para obtener señales que hacen que la aplicación finalice. A continuación la aplicación recibe el código de retorno de la API. Por ejemplo, para ignorar SIGPIPE añada la siguiente instrucción en la aplicación: `signal(SIGPIPE, SIG_IGN)`. Una vez añadida esta información, en lugar de la aplicación existente en un conducto roto, se devuelve el código de retorno adecuado.

Puede utilizar este proceso subordinado, señales Trusted Communication Agent (TCA) si la opción `passwordaccess` está configurada en `generate`. Al utilizar el proceso TCA, IBM Spectrum Protect utiliza la señal SIGCLD. Si su aplicación utiliza la señal SIGCLD, tenga en cuenta la interferencia potencial de los procesos de IBM Spectrum Protect y cómo se utiliza SIGCLD. Para obtener más información acerca del uso de TCA, consulte "Seguridad de sesión" en la página 18.

Iniciar o finalizar una sesión

IBM Spectrum Protect es un producto basado en sesión, y todas las actividades deben realizarse dentro de una sesión IBM Spectrum Protect. Para iniciar una sesión, la aplicación inicia la llamada **dsmInitEx**. Esta llamada debe realizarse antes de cualquier otra llamada API que no sea **dsmQueryApiVersionEx**, **dsmQueryCliOptions**, o **dsmSetUp**.

Se puede llamar a la función **dsmQueryCliOptions** únicamente antes de la llamada **dsmInitEx**. La función devuelve los valores de opciones importantes, como de archivos de opciones, configuraciones de compresión y parámetros de comunicación. La función**dsmInitEx** configura una sesión con el servidor tal y como se indica en el pase de la llamada o se define en los archivos de opciones.

El nombre del nodo del cliente, el nombre del propietario y los parámetros de la contraseña pasan a la llamada **dsmInitEx**. El nombre del propietario distingue entre mayúsculas y minúsculas, pero no lo hacen el nombre del nodo y la contraseña. Los nodos del cliente de aplicación deben estar registrados en el servidor antes de que se inicie la sesión.

Cada vez que un cliente de aplicación de API inicia una sesión con el servidor, el tipo de aplicación de cliente se registra con el servidor. Especifique siempre una abreviatura del sistema operativo para el valor de tipo de aplicación porque este valor se entra en el campo de plataforma en el servidor. La longitud máxima de la cadena es DSM_MAX_PLATFORM_LENGTH.

La llamada de función **dsmInitEx** establece la sesión IBM Spectrum Protect con el archivo de configuración de la API y la lista de opciones del cliente de la aplicación. El cliente de la aplicación puede utilizar el archivo de configuración de la API y la lista de opciones para definir un número de opciones IBM Spectrum Protect. Estos valores modifican los valores definidos en los archivos de configuración del usuario durante la instalación. Los usuarios no pueden cambiar las opciones que defina el administrador. Si el cliente de la aplicación no cuenta con su propio archivo de configuración y lista de opciones, es posible configurar ambos parámetros en NULO. Para obtener más información acerca de los archivos de configuración, consulte el tema siguiente:

"Conceptos básicos de los archivos de opciones y configuración" en la página 1

La llamada a función **dsmInitEx** establece la sesión de IBM Spectrum Protect , utilizando parámetros que permiten la verificación ampliada.

Compruebe la llamada de función **dsmInitEx** y el código de retorno de la información**dsmInitExOut**. El administrador ha cancelado la última sesión si el código de retorno es ok (RC=ok) y el código de retorno de la información (infoRC) es DSM_RC_REJECT_LASTSESS_CANCELED. Para finalizar la sesión actual de inmediato, llame a **dsmTerminate**.

La llamada **dsmQuerySessOptions** devuelve los mismos campos que la llamada **dsmQueryCliOptions**. La llamada se puede establecer únicamente dentro de una sesión. Los valores reflejan las opciones de cliente que son válidas durante esa sesión, desde los archivos de opciones así como cualquier modificación de la llamada **dsmInitEx**.

Una vez que inicia una sesión, la aplicación puede enviar una llamada a **dsmQuerySessInfo** con el fin de determinar los parámetros del servidor que están configurados en dicha sesión. Con esta llamada se devuelven a la aplicación elementos como el dominio de política y los límites de transacción.

Finalizar una sesión con una llamada **dsmTerminate**. Cualquier conexión con el servidor está cerrada y todos los recursos asociados con esta sesión están liberados.

Para obtener un ejemplo del inicio y final de una sesión, consulte el tema siguiente:

Figura 2 en la página 20

El ejemplo define un número de variables globales y locales que se utilizan en llamadas a **dsmInitEx** y **dsmTerminate**. La llamada **dsmInitEx** toma un puntero a `dsmHandle` como parámetro, mientras la llamada **dsmTerminate** toma el `dsmHandle` como parámetro. El ejemplo en Figura 3 en la página 21 muestra los detalles de **rcApiOut**. La función **rcApiOut** llama a la función API **dsmRCMsg**, que traduce un código de retorno en un mensaje. La llamada **rcApiOut** imprime después el mensaje para el usuario. En la aplicación de ejemplo de la API se incluye una versión de **rcApiOut**. La función **dsmApiVersion** es una definición de tipo que se encuentra en el archivo de cabecera `dsmapitd.h`.

Seguridad de sesión

El sistema basado en la sesión de IBM Spectrum Protect tiene componentes de seguridad que permiten a las aplicaciones iniciar sesiones de forma segura. Estas medidas de seguridad prohíben el acceso no autorizado al servidor y ayudan a garantizar la integridad del sistema.

Todas las sesiones que se inician en el servidor deben completar una proceso de inicio de sesión que requiere una contraseña. Cuando la contraseña se une al nombre del nodo del cliente, se garantiza la autorización adecuada al conectarse al servidor. El cliente de la aplicación proporciona esta contraseña al API para iniciar la sesión.

Existen dos métodos de proceso de contraseña disponibles:*passwordaccess=prompt* o *passwordaccess=generate*. Si utiliza la opción *passwordaccess=prompt*, debe incluir el valor de la contraseña en cada llamada **dsmInitEx**. O bien, es posible proporcionar el nombre de propietario y nodo en la llamada **dsmInitEx**.

Las contraseñas tienen periodos de caducidad asociados. Si una llamada **dsmInitEx** falla con un código de retorno de contraseña caducada (DSM_RC_REJECT_VERIFIER_EXPIRED), el cliente de la aplicación deben introducir la llamada **dsmChangePW** utilizando el manejador que devuelve **dsmInitEx**. Esto actualiza la contraseña antes de que la sesión se establezca correctamente. El ejemplo en Figura 4 en la página 21 demuestra el procedimiento para cambiar una contraseña utilizando **dsmChangePW**. El propietario de inicio de sesión debe utilizar un ID de usuario root o un ID de usuario autorizado para cambiar la contraseña.

El segundo método, *passwordaccess=generate*, cifra y almacena el valor de la contraseña en un archivo. El nombre de nodo y el nombre de propietario no se puede proporcionar en la llamada **dsmInitEx**, y se utilizan los valores predeterminados del sistema. Esto protege la seguridad del archivo de la contraseña. Cuando la contraseña caduca, el parámetro `generate` crea una nueva y actualiza el archivo de contraseña automáticamente.

Sugerencias:

1. Si dos máquinas físicas distintas tienen el mismo nombre de nodoIBM Spectrum Protect o se definen varias rutas en un nodo utilizando varias secciones del servidor, es posible que *passwordaccess=generate* sólo funcione en la sección que se utiliza primero después de la caducidad de la contraseña. Durante el primer contacto entre el cliente y el servidor, se solicita al usuario la misma contraseña para cada sección de servidor por separado y se almacena una copia de la contraseña para cada sección. Cuando caduca la contraseña, se genera una nueva para la sección que establece el primer contacto entre el cliente y el servidor. Todos los intentos posteriores de conectar mediante otras secciones de servidor fallarán, puesto que no existe enlace lógico entre sus copias de la contraseña anterior y la copia actualizada generada por la primera sección utilizada después de que caducara la contraseña. En este caso, debe actualizar las contraseñas antes de que caduquen o después de que caduquen con el fin de subsanar la situación del siguiente modo:
   a. Ejecute **dsmadmc** y actualice la contraseña en el servidor.
   b. Ejecute **dsmc -servername=stanza1** y utilice la nueva contraseña para generar una entrada adecuada.
   c. Ejecute **dsmc -servername=stanza2** y utilice la nueva contraseña para generar la entrada adecuada.
2. Para UNIX o Linux: solo el usuario root o el usuario autorizado pueden cambiar la contraseña cuando utilizan *passwordaccess=prompt*. Solo el usuario root o un usuario autorizado pueden iniciar el archivo de contraseña utilizando *passwordaccess=generate*. Es posible utilizar el proceso secundario del agente de comunicación de confianza (TCA) para procesar la contraseña. La aplicación debe estar al corriente de esto porque se utilizan un proceso secundario y una señal SIGCLD. El agente TCA no se utiliza en estas situaciones:
   - La opción *passwordaccess* está configurada en *prompt*.
   - El usuario de inicio de sesión es root.
   - El interlocutor de la función debe ser un usuario autorizado.

   Restricción: Las opciones `usuarios` y `grupos` no se reconocen.

Una aplicación puede restringir el acceso de usuario por otros métodos, como por ejemplo configurar los filtros de acceso.

Las aplicaciones que utilizan varias conexiones IP a un único servidor IBM Spectrum Protect deben usar el mismo nombre de nodo y la contraseña de cliente de IBM Spectrum Protect para cada sesión. Siga estos pasos para habilitar esta compatibilidad:

1. Defina una sección del servidor IBM Spectrum Protect en el archivo dsm.sys.
2. En las conexiones que no utilizan la dirección IP predeterminada, especifique los valores de las opciones de la dirección *TCPserver* y *TCPport* en la llamada **dsmInitEx**.

Estos valores modifican la información de conexión del IP, pero la sesión utiliza el mismo nodo de la sección dsm.sys y la información de la contraseña.

Nota: Los nodos de un clúster comparten una única contraseña.

```
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);
}
```

Figura 2. Un ejemplo de iniciar y finalizar una sesión

```
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;
}
```

Figura 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;
```

Figura 4. An example of changing a password

Configuración de la opción `passwordaccess` a `generate` sin TCA

El agente de comunicación de confianza (TCA) es un proceso secundario que normalmente controla el acceso al archivo de la contraseña protegido. En sistemas UNIX y Linux, puede iniciar sesión como usuario autorizado y establecer la opción `passwordaccess` en `generate` sin iniciar el TCA.

Complete estos pasos cuando establezca `passwordaccess` en `generate` sin TCA:

1. Grabe la aplicación con una llamada a **`dsmSetUp`** la cual pasa *argv[0]*. *argv[0]* contiene el nombre de la aplicación que llama al API. La aplicación puede ejecutar un usuario autorizado; no obstante, el administrador debe decidir el nombre de inicio de sesión del usuario autorizado.
2. Configure el bit de ID de usuario efectivo (bit `S`) para el ejecutable de aplicación en `On`. El propietario del archivo ejecutable de la aplicación podrá entonces convertirse en usuario autorizado y crear un archivo de contraseña, actualizar contraseñas y ejecutar aplicaciones. El propietario del archivo ejecutable debe ser el mismo que el ID de usuario que ejecuta el programa. En el siguiente ejemplo, *`User`* es `user1`, el nombre del archivo ejecutable de la aplicación `applA`, y `user1` tiene permisos de lectura y grabación en el directorio`/home/user1`. El archivo ejecutable `applA` tiene los siguientes permisos:

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

3. Indique a los usuarios de la aplicación que utilicen el nombre de usuario autorizado para iniciar sesión. IBM Spectrum Protect verifica que el ID de inicio de sesión coincide con el propietario ejecutable de la aplicación antes de que permita acceder al archivo de contraseña protegido.
4. Configure la opción `passworddir` en el archivo `dsm.sys` para apuntar a un directorio en el que el usuario tenga permisos de lectura y grabación. Por ejemplo, escriba la siguiente línea en la stanza del servidor del archivo `dsm.sys`:

   ```
   passworddir /home/user1
   ```
5. Cree el archivo de contraseña y asegúrese de que el usuario autorizado sea su propietario.
6. Inicie sesión como `user1` y ejecute `app1A`.
7. Llame a **dsmSetUp** y pase a *argv*.

Crear un usuario administrativo con autorización de propietario de cliente

Un usuario administrativo con autorización de propietario de cliente puede definir los parámetros en la llamada de función **dsmInitEx** para empezar las sesiones. Este usuario funciona como un "usuario administrativo" con autorización para restaurar y realizar copias de seguridad en los nodos definidos.

Para recibir autorización de propietario de cliente, realice los pasos siguientes en el servidor:

1. Definir el usuario administrativo:

   ```
   REGister Admin admin_name password
   ```

   Donde:
   - *admin_name* es el nombre del usuario administrativo.
   - *password* es la contraseña del administrador.
2. Definir el nivel de autorización. Los usuarios con autorización de política o sistema también tienen autorización de propietario de cliente.

   ```
   Otorgar autorización admin_name classes authority node
   ```

   Donde:
   - *admin_name* es el usuario administrativo.
   - *classes* es el nodo.
   - *authority* tiene uno de estos niveles de autorización:
     - `owner`: autorización de copia de seguridad y restauración para el nodo
     - `node`: nodo único
     - `domain`: grupo de nodos
3. Definir el acceso a un nodo único.

   ```
   Register Node node_name password userid=ID de usuario
   ```

   Donde:
   - *node_name* es el nodo del usuario del cliente
   - *password* es la contraseña del nodo de usuario del cliente
   - *ID de usuario* es el nombre de usuario administrativo

Cuando la aplicación utiliza el usuario administrativo, la función **dsmInitEx** se llama con los parámetros `userName` y`userNamePswd`.

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

Puede configurar La opción `passwordaccess` a `generate` o `prompt`. Con ambos parámetros, el valor `userNamePswd` inicia la sesión. Cuando se inicia la sesión, puede ocurrir cualquier proceso de restauración o copia de seguridad para ese nodo.

ID y nombres de objeto

El servidor IBM Spectrum Protect es un servidor de almacenamiento de objeto cuya función principal es almacenar y recuperar de forma eficaz los objetos nombrados. El ID del objeto es único en cada objeto y permanece con el objeto durante toda la vida del objeto *excepto* cuando realiza exportaciones o importaciones.

Para cumplir estos requisitos IBM Spectrum Protect cuenta con dos áreas de almacenamiento principales, la base de datos y el almacén de datos.

- La base de datos contiene todos los metadatos, como el nombre o los atributos asociados con el objeto.
- El almacén de datos contiene los datos del objeto. El almacén de datos es una jerarquía de almacenamiento que define el administrador del sistema. Los datos se almacenan de forma eficaz y se administran en línea o en un medio externo, dependiendo de los costes y de las necesidades de acceso.

Cada uno de los objetos almacenado en el servidor tiene un nombre asociado. El cliente controla los siguientes componentes clave del nombre:

- Nombre de espacio de archivo
- Nombre de alto rango
- Nombre de bajo rango
- Tipo de objeto

Al tomar decisiones sobre cómo nombrar los objetos de una aplicación, es posible que necesite utilizar un nombre externo en los nombres de objeto completos para el usuario final. Concretamente, es posible que el usuario final necesite especificar el objeto en una sentencia de exclusión o inclusión cuando se ejecuta la aplicación. La sintaxis exacta del nombre del objeto en estas sentencias depende de la plataforma. En el sistema operativo de Windows, se utiliza en las sentencias de inclusión o exclusión la letra de la unidad asociada con el espacio de archivo en lugar del propio nombre del espacio del archivo.

Es posible que el valor del ID del objeto que se asignó cuando creó el objeto no sea el mismo que cuando realizó un proceso de restauración. Las aplicaciones deben guardar el nombre del objeto y a continuación realizar la consulta para obtener el ID del objeto actual antes de llevar a cabo una restauración.

Nombre de espacio de archivo

El nombre de espacio de archivo es uno de los componentes de almacenamiento más importantes. Puede ser el nombre de un sistema de archivos, unidad de disco o cualquier otro calificador de alto rango que agrupa datos relacionados.

IBM Spectrum Protect utiliza el espacio de archivo para identificar el sistema de archivo o la unidad del disco en el que se encuentran los datos. De este modo, es posible llevar a cabo acciones en todas las entidades de un espacio de objeto como consultar todos los objetos dentro de un espacio de archivo específico. Como el espacio de archivos es un componente tan importante del convenio de denominación de IBM Spectrum Protect, utilice llamadas especiales para registrar, actualizar, consultar o suprimir espacios de archivos.

El servidor también tiene mandatos administrativos para consultar los espacios de archivo en un nodo determinado en el almacenamiento IBM Spectrum Protect, y eliminarlos si es necesario. Todos los datos que almacena el cliente de la aplicación deben tener un nombre de espacio asociado. Seleccione el nombre cuidadosamente para agrupar datos similares en el sistema.

Un cliente de aplicación debe seleccionar nombres de espacio de archivos distintos de los nombres que utiliza un cliente de archivado/copia de seguridad, con el fin de evitar interferencias. El cliente de la aplicación debe publicar los nombres de espacio de archivo para que los usuarios puedan identificar los objetos en sentencias de inclusión y exclusión si fuera necesario.

Nota: En plataformas Windows, una letra de unidad está asociada con un espacio de archivo. Cuando registra o actualizar un espacio de archivo, debe proporcionar una letra de unidad. Debido a que la lista de inclusión/exclusión hace referencia a la letra de unidad, debe anotar cada letra y su espacio de archivo asociado. En el programa de ejemplo dapismp, la letra de unidad se configura en "G" de forma predeterminada.

Consulte Capítulo 2, “Crear y ejecutar la aplicación de API de ejemplo”, en la página 5 para obtener más información acerca de los programas de ejemplo.

Nombres de alto y bajo rango

Otros dos componentes del nombre del objeto son el calificador de nombre de alto rango y el calificador de nombre de bajo rango. El calificador de nombre de alto rango es la ruta del directorio al que pertenece el objeto, y el nombre del calificador de bajo rango es el nombre real del objeto en dicha ruta de directorio.

Cuando el nombre de espacio de archivo, nombre de alto rango y nombre de bajo rango están concatenados, deben formar un nombre correcto sintácticamente en el sistema operativo en el que se ejecuta el cliente. No es necesario que el nombre exista como un objeto en el sistema o que se parezca a los datos del sistema de archivos local. No obstante, el nombre debe cumplir con las normas de nomenclatura estándar para que se procese correctamente en las llamadas **dsmBindMC**. Consulte “Conceptos básicos sobre archivar y realizar copias de seguridad de los objetos” en la página 43 para consultar las normas de nomenclatura relacionadas con la gestión de políticas.

Tipo de objeto

El tipo de objeto define el objeto como un archivo o un directorio. Un archivo es un objeto que contiene tanto atributos como datos binarios, y un directorio es un objeto que contiene sólo atributos.

Tabla 7 muestra lo que el cliente de aplicación codifica para los nombres de objeto por plataforma.

Tabla 7. Ejemplos de nombre de objeto de aplicación por plataforma

Plataforma	Código de cliente por nombre de objeto
UNIX o Linux	/myfs/highlev/lowlev
Windows	"myvol\\highlev\\lowlev" **Nota:** En una plataforma Windows, una barra invertida doble se convierte en una barra invertida única porque una barra invertida es el carácter del espacio. Los nombres de espacio de archivo empiezan con una barra invertida en la plataforma UNIX o Linux, pero no empiezan con una barra invertida en la plataforma Windows.

Acceder a objetos como propietario de sesión

Cada objeto tiene un nombre de propietario asociado. Las normas que determinan a qué objetos se accede dependen de qué nombre de propietario se utilice cuando se inicia una sesión. Utilice este valor de propietario de sesión para controlar el acceso al objeto.

El propietario de la sesión se configura durante la llamada a **dsmInitEx** en el parámetro *clientOwnerNameP*. Si inicia una sesión con el nombre de propietario **dsmInitEx** de *NULL* y utiliza *passwordaccess=prompt*, dicho propietario de sesión se maneja con la autoridad de la sesión (usuario root o autorizado). Esto también es verdadero si inicia sesión con un ID de usuario root o un ID de usuario autorizado y utiliza *passwordaccess= generate*. Durante una sesión iniciada de esta manera, puede realizar cualquier acción en un objeto propiedad de este nodo independientemente del propietario real de dicho objeto.

Si una sesión se inicia con un nombre de propietario específico, la sesión sólo puede realizar acciones en objetos que tienen un nombre de propietario de objeto asociado. Los archivados o copias de seguridad en el sistema deben tener este nombre de propietario asociado con ellos. Todas las consultas realizadas devuelven sólo los valores que tienen este nombre de propietario asociado. El valor del propietario del objeto se configura durante la llamada **dsmSendObj** en el campo **Owner** de la estructura **ObjAttr**. El nombre de propietario es sensible a mayúsculas y minúsculas. Tabla 8 resume las condiciones bajo las cuales un usuario tiene acceso a un objeto.

Tabla 8. Resumen de acceso de usuario a objetos

Propiedad de sesión	Propiedad de objeto	Acceso de usuario
NULO (raíz, propietario de sistema)	“ ” (cadena vacía)	Sí
NULO	Nombre específico	Sí
Nombre específico	“ ” (cadena vacía)	No
Nombre específico	Mismo nombre	Sí
Nombre específico	Nombre diferente	No

Acceder a objetos de nodos y propietarios

Hay tres llamadas de función compatibles con distintos nodos y propietarios en la misma plataforma: **dsmSetAccess**, **dsmDeleteAccess**, y **dsmQueryAccess**. Estas funciones, junto con las opciones de cadena *-fromnode* y *-fromowner* que se pasan en **dsmInitEx**, permiten una consulta a través de distintos nodos, así como restaurar y recuperar procesos a través de la API.

Por ejemplo, Usuario A en nodo A utiliza la llamada de función **dsmSetAccess** para dar acceso a su copia de seguridad bajo el espacio de archivo `/db` al Usuario B del Nodo B. La regla de acceso aparece como:

ID	Tipo	Nodo	Usuario	Vía de acceso
1	Copia de seguridad	Nodo B	Usuario B	`/db/*/*`

Cuando el usuario B inicia sesión en Nodo B, la cadena de opción a **dsmInitEx** es:

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

Estas opciones están configuradas para esta sesión. Todas las consultas acceden a los espacios del archivo, y a los archivos del nodo A. No están permitidas las copias de seguridad y los archivados. Sólo los procesos de consulta, restauración y recuperación están permitidos en los espacios de archivo para los que el Usuario B tiene acceso. Si la aplicación intenta ejecutar cualquier operación utilizando **dsmBeginTxn** (para ejemplos, copias de seguridad o actualizaciones) mientras la sesión está iniciada con una opción *-fromnode* o *-fromowner*, entonces **dsmBeginTxn** falla con el código de retorno DSM_RC_ABORT_NODE_NOT_AUTHORIZED. Consulte las llamadas de función individuales y "**dsmInitEx**" en la página 124 para obtener más información.

Consejo: En UNIX y Linux es posible especificar *–fromowner=root* en la cadena de opción que pasa a la llamada de función **dsmInitEx**. Esto permite a los usuarios que no sean de raíz acceder a archivos propiedad de la raíz si se lleva a cabo un acceso.

Utilice la opción *asnodename* en la cadena de opción **dsmInitEx** con la función adecuada para realizar copias de seguridad, archivados, restauraciones, recuperaciones, consultas o eliminaciones de datos bajo el nombre del nodo de destino en el servidor IBM Spectrum Protect. Consulte "Soporte de copia de seguridad de varios nodos con el proxy de nodo de cliente" en la página 85 para obtener más información sobre cómo habilitar esta opción.

Gestionar espacios de archivos

Debido a que el espacio de archivos es tan importante para la operación del sistema, se utiliza un grupo de llamadas por separado para registrar, actualizar y eliminar los identificadores de espacio de archivos. Antes de almacenar cualquier objeto asociado con un espacio de archivo en el sistema, debe primero registrar el espacio de archivo con IBM Spectrum Protect.

Utilice la llamada **dsmRegisterFS** para realizar esta tarea. Para obtener más información acerca de los nombres de objeto e ID, consulte "ID y nombres de objeto" en la página 23.

El identificador del espacio de archivo es el calificador en el nivel superior de la jerarquía del nombre de tres partes. Agrupar los datos relacionados dentro de un espacio de archivo facilita la gestión de dichos datos. Por ejemplo, el cliente de la aplicación o el administrador del servidor IBM Spectrum Protect pueden eliminar un espacio de archivo y todos los objetos dentro de dicho espacio de archivo.

Los espacios de archivos también permiten que el cliente de aplicaciones proporcione información sobre el espacio de archivos al servidor que el servidor pueda luego consultar. Esta información se devuelve en la consulta en la estructura **qryRespFSData** e incluye la siguiente información de sistema de archivos:

Tipo	Definición
fstype	El tipo de espacio de archivo. Este campo es una cadena de caracteres que configura el cliente de la aplicación.
fsAttr[platform].fsInfo	Un campo de información de cliente que se utiliza para los datos específicos del cliente.
capacity	La cantidad de espacio total en el espacio del archivo.
ocupación	La cantidad de espacio que se ocupa en el espacio de archivos.
backStartDate	El indicador de hora de cuándo se inició la última copia de seguridad (configurado al enviar una llamada **dsmUpdateFS**).
backCompleteDate	El indicador de hora de cuándo se finalizó la última copia de seguridad (configurado al enviar una llamada **dsmUpdateFS**).

El uso de la capacidad y de la ocupación depende del cliente de la aplicación. Es posible que algunas aplicaciones no necesiten información acerca del tamaño del espacio de archivos, en cuyo caso estos campos pueden de forma predeterminada ser `0`. Para obtener más información sobre la consulta de espacios de archivos, consulte "Consultas para el sistema IBM Spectrum Protect" en la página 34.

Después de registrar un espacio de archivo en el sistema, es posible archivar o realizar copias de seguridad de los objetos en cualquier momento. Para actualizar los campos de capacidad y de ocupación del espacio de archivos después de una operación de archivado y copia de seguridad, llame a **dsmUpdateFS**. Esta llamada asegura que los valores para la ocupación y capacidad del sistema de archivos son actuales. También es posible actualizar los campos **fsinfo**, **backupstart**, y **backupcomplete**.

Si desea supervisar las últimas fechas de copia de seguridad, entre una llamada**dsmUpdateFS** antes de iniciar la copia de seguridad. Establezca la acción de actualización en DSM_FSUPD_BACKSTARTDATE. Esto fuerza al servidor la configuración del campo **backStartDate** del espacio de archivo con la hora actual. Tras finalizar la copia de seguridad para ese espacio de archivo, introduzca una llamada **dsmUpdateFS** con la acción de actualización configurada en DSM_FSUPD_BACKCOMPLETEDATE. Esta llamada crea una indicación de fecha y hora al final de la copia de seguridad.

Si un espacio de archivo ya no es necesario, puede borrarlos con el mandato **dsmDeleteFS**. En un sistema operativo UNIX o Linux, solo el usuario root o los usuarios autorizados pueden suprimir los espacios de archivos.

Los ejemplos en Figura 5 en la página 28 demuestran cómo utilizar las tres llamadas de espacio de archivo de UNIX o Linux. Para obtener un ejemplo de cómo utilizar las tres llamadas de espacio de archivos de Windows, consulte el

código de programa de ejemplo instalado en el sistema.

```
/*  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);
}
```

Figura 5. Un ejemplo de cómo trabajar con espacios de archivo, Parte 1

```
/* 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);
```

Figura 6. Un ejemplo de cómo trabajar con espacios de archivo, Parte 2

```
/*  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");
```

Figura 7. Un ejemplo de cómo trabajar con espacios de archivo, Parte 3

Asociar objetos con clases de gestión

Una función principal de IBM Spectrum Protect es el uso de políticas (clases de gestión) para definir cómo se almacenan y gestionan los objetos en el almacenamiento IBM Spectrum Protect. Un objeto se asocia con una clase de gestión cuando se archivo o se realiza una copia de seguridad.

Esta clase de gestión determina:
- Cuántas versiones del objeto se guardan si se hacen copias de seguridad de éste
- Cuánto tiempo se guardan las copias archivadas
- Dónde se introducen los objetos en la jerarquía de almacenamiento del servidor

Las clases de gestión consisten tanto en grupos de copias de seguridad como en grupos de copias de archivados. Un grupo de copia es un conjunto de atributos que definen las políticas de gestión de un objeto que se va a archivar o al que se le va a realizar una copia de seguridad. Si se está realizando una operación de copia de seguridad, se aplican los atributos en el grupo de la copia de seguridad. Si se está realizando un archivado, se aplican los atributos del grupo de la copia de archivado.

El grupo de la copia de archivado o copia de seguridad en una clase de gestión particular puede estar vacío o ser NULO. Si un objeto está vinculado a un grupo de copia de seguridad NULO, no es posible realizar una copia de seguridad a dicho objeto. Si un objeto está vinculado al grupo de copia de archivado NULO, no es posible archivar dicho objeto.

Como el uso de una política es un componente muy importante de IBM Spectrum Protect, la API requiere que a todos los objetos enviados al servidor se les asigne primero una clase de gestión mediante la llamada de **dsmBindMC**. Con el software de IBM Spectrum Protect, puede usar una lista de inclusión-exclusión que afecte al enlace de clase de gestión. La llamada La llamada **dsmBindMC** utiliza la lista de inclusión/exclusión para realizar vinculaciones de clase de gestión.

Las sentencias de inclusión pueden asociar una clase de gestión específica con una copia de seguridad o archivado. Las sentencias de exclusión impiden que se realicen copias de seguridad de los objetos pero no archivados.

La API requiere que se llame a **dsmBindMC** antes de realizar una copia de seguridad o archivar un objeto. La llamada **dsmBindMC** devuelve una estructura mcBindKey que contiene información sobre la clase de gestión y los grupos de copia asociados con el objeto. Compruebe el destino del grupo de copia antes de proceder con un envío. Cuando envía varios objetos en una única transacción, deben tener el mismo destino de grupo de copia. La llamada de función **dsmBindMC** devuelve la siguiente información:

Tabla 9. Información devuelta en la llamada dsmBindMC

Información	Descripción
Clase de gestión	El nombre de la clase de gestión que se vinculó al objeto. El cliente de la aplicación puede enviar la llamada **dsmBeginQuery** con el fin de determinar todos los atributos de esta clase de gestión.
Grupo de copia de seguridad	Informa si existe un grupo de copia de seguridad para esta clase de gestión. Si se realiza una operación de copia de seguridad y no existe un grupo de copias de seguridad, este objeto no se podrá enviar a almacenamiento. Recibe un código de error si intenta enviarlo con la llamada **dsmSendObj**.
Destino de copia de seguridad	Este campo identifica la agrupación de almacenamiento a la que se envían los datos. Si está realizando varias transacciones de copia de seguridad de un objeto, todos los destinos de la copia dentro de dicha transacción deben ser los mismos. Si un objeto tiene un destino de copia distinto que los objetos anteriores en la transacción, finalice la transacción actual y empiece una transacción nueva antes de enviar el objeto. Recibe un código de error si intenta enviar objetos a distintos destinos de copia dentro de la misma transacción.
Grupo de copias archivadas	Le informa si un grupo de copias archivadas existe para esta clase de gestión. Si se está realizando una operación de archivado y no existe un grupo de copias de archivadas, este objeto no podrá enviarse a almacenamiento. Recibe un código de error si intenta enviarlo utilizando la llamada **dsmSendObj**.
Destino de copias archivadas	Este campo identifica la agrupación de almacenamiento a la que se envían los datos. Si está realizando varias transacciones de objetos archivados, todos los destinos de la copia dentro de dicha transacción deben ser los mismos. Si un objeto tiene un destino de copia distinto que los objetos anteriores en la transacción, finalice la transacción actual y empiece una transacción nueva antes de enviar el objeto. Recibe un código de error si intenta enviar objetos a distintos destinos de copia dentro de la misma transacción.

Las copias de seguridad de un objeto se pueden revincular a distintas clases de gestión si se realiza una copia de seguridad subsiguiente con el mismo el mismo nombre de objeto que utiliza una clase de gestión distinta a la del original. Por ejemplo, si realiza una copia de seguridad del Objeto A y lo vincula a Mgmtclass1, y más tarde hace una copia de seguridad del Objeto A y lo vincula a Mgmtclass2, la copia de seguridad más actual revincula las copias inactivas a Mgmtclass2. Los parámetros definidos en Mgmtclass2 controlan ahora todas las copias. No obstante, los datos no se mueven si el destino es distinto.

También puede revincular las copias de seguridad a distintas clases de gestión utilizando la llamada **dsmUpdateObj** o **dsmUpdateObjEx** con la acción DSM_BACKUPD_MC.

Referencia relacionada:

Opción Deduplication

Clases de gestión de consulta

Las aplicaciones pueden consultar las clases de gestión con el fin de determinar qué clases de gestión son posibles en un nodo determinado y qué atributos se encuentran en la clase de gestión.

Sólo puede vincular objetos a las clases de gestión utilizando la llamada **dsmBindMC**. Es posible que desee que sus aplicaciones consulten los atributos de la clase de gestión y las muestren a los usuarios finales. Para obtener más información, consulte el apartado "Consultas para el sistema IBM Spectrum Protect" en la página 34.

En el ejemplo Figura 8, se utiliza una sentencia de cambio para distinguir entre las operaciones de archivado y de copia de seguridad al llamar a **dsmBindMC**. La información que devuelve esta llamada se almacena en la estructura **MCBindKey**.

```
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;
}
```

Figura 8. Un ejemplo de asociación de una clase de gestión con un objeto

Retener y liberar la caducidad/eliminación

Puede retener la eliminación y la caducidad de objetos de archivado en respuesta a una acción pendiente o en curso que requiere que se retengan los datos particulares. En caso de que se inicie una acción que requiera el acceso a los datos, dichos datos deben estar disponibles hasta que la acción haya concluido y ya no sea necesario el acceso a los datos como parte del proceso. Después de determinar que la suspensión ya no se requiere, se reinicia la eliminación y caducidad normal de acuerdo al periodo de retención original.

Verifique el servidor tiene licencia emitiendo una llamada de**dsmRetentionEvent** de prueba:

1. Consulte un objeto que desee retener y obtenga el ID.
2. Emita el **dsmBeginTxn**, **dsmRetentionEvent** con `Hold`, y **dsmEndTxn**.
3. Si el servidor no tiene licencia, recibe un voto de cancelación con el código de razón de `DSM_RC_ABORT_LICENSE_VIOLATION`.

Restricciones:

1. No es posible emitir más de una llamada **dsmRetentionEvent** en una única transacción.
2. No es posible emitir una retención en un objeto que ya está retenido.

1. Para retener los objetos, siga estos pasos:
   a. Consulte al servidor todos los objetos que desea retener. Obtenga el ID de objeto de cada uno de ellos.
   b. Emita una llamada **dsmBeginTxn**, a continuación emita una llamada **dsmRetentionEvent** con la lista de objetos, seguido de una llamada

dsmEventType: eventHoldObj. Si el número de objetos excede el valor de maxObjPerTxn, utilice varias transacciones.

c. Utilice qryRespArchiveData en la llamada de función **dsmGetNextQObj** para confirmar que los objetos están retenidos. Compruebe el valor de objHeld en qryRespArchiveData.

2. Para liberar los objetos de la retención, siga estos pasos:
   a. Consulte al servidor todos los objetos que desea liberar. Obtenga el ID de objeto de cada uno de ellos.
   b. Emita una llamada **dsmBeginTxn**, a continuación emita una llamada **dsmRetentionEvent** con la lista de objetos, seguido de una llamada **dsmEventType**: eventReleaseObj. Si el número de objetos excede el valor de maxObjPerTxn, utilice varias transacciones.
   c. Utilice la respuesta **qryRespArchiveData** en la llamada de función dsmGetNextQObj para confirmar si los objetos se han liberado de la retención. Compruebe el valor de objHeld en **qryRespArchiveData**.

Protección de retención de datos archivados

Los datos que están bajo el control de IBM Spectrum Protect no se pueden modificar mediante agentes no autorizados como, por ejemplo, una persona o un programa. Esta protección se amplía para evitar la supresión de datos, como objetos de archivado, por cualquier agente antes de la caducidad del período de retención.

La protección de la retención de archivado ayuda a garantizar que ningún programa ni ninguna persona pueda de forma malintencionada o accidental suprimir los datos que estén bajo el control de IBM Spectrum Protect. Un objeto de archivado que se envía a un servidor de protección de la retención de supresiones accidentales y tiene un periodo de retención implementado. La protección de retención de archivado tiene las siguientes limitaciones:

- Sólo se permiten operaciones de archivado en un servidor de protección de retención.
- Cualquier objeto que no esté vinculado de forma explícita a una clase de gestión a través de un valor en la llamada de función **dsmBindMc** o a través de sentencias de inclusión/exclusión, está vinculado a un nombre explícito de la clase de gestión predeterminada. Por ejemplo, si la clase de gestión predeterminada de la política del nodo es MC1, el objeto se vincula de forma explícita a MC1 en vez de a DEFAULT. En una respuesta a la consulta, el objeto aparece vinculado a MC1.
- Después de habilitar la protección de retención de los datos de archivado, cualquier intento de suprimir un objeto antes de que caduque el periodo de retención devuelve el código DSM_RC_ABORT_DELETE_NOT_ALLOWED en la transacción final.

Consulte la documentación del servidor de IBM Spectrum Protect para obtener instrucciones para configurar la protección de retención en un objeto de archivado.

Para configurar la protección de retención de datos archivados, lleve a cabo los pasos siguientes:

1. En una instalación de servidor nueva sin datos previos, ejecute el mandato **SET ARCHIVERETENTIONPROTECTION ON**.
2. En la opción de la string de la API en la llamada de función **dsmInit** o **dsmInitEx**, escriba la siguientes instrucciones:

```
-ENABLEARCHIVERETENTIONPROTECTION=yes
```

También puede definir la opción enablearchiveretentionprotection en el archivo dsm.opt en sistemas que no sean UNIX, o en el archivo dsm.sys en sistemas UNIX:

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

Para obtener más información sobre esta opción, consulte el apartado "La opción enablearchiveretentionprotection".

3. Emita una consulta al servidor para confirmar que el servidor IBM Spectrum Protect está capacitado para la protección de retención de copias archivadas. Compruebe el valor del campo archiveRetentionProtection en la estructura dsmQuerySessInfo.

La opción enablearchiveretentionprotection

La opción enablearchiveretentionprotection especifica si habilitar la protección de retención de datos para archivar objetos en el servidor IBM Spectrum Protect dedicado a esta finalidad. Su administrador de servidor debe activar la protección de retención de datos en un nuevo servidor que todavía no tenga objetos almacenados (copia de seguridad, archivado, espacio gestionado). Si la aplicación API intenta almacenar una copia de seguridad o un objeto de espacio gestionando en el servidor, se activa un mensaje de error.

La nota en Capítulo 3, "Consideraciones para el diseño de una aplicación", en la página 11 dice: "no almacene los valores de objectID para utilizar las futuras restauraciones. No se garantiza que sea persistente durante la vida del objeto." es positiva para las aplicaciones de gestión de archivado ya que el servidor de gestión de archivado no soporta exportaciones o importaciones. Las aplicaciones de gestión de archivado pueden guardar y utilizar el ID de objeto para mejorar el rendimiento durante la restauración del objeto.

Si el servidor emite el mandato **SET ARCHIVERETENTIONPROTECTION ON**, no podrá suprimir un objeto archivado del servidor usando el mandato **delete filespace** hasta que se satisfagan los parámetros de política del grupo de copia de archivado. Consulte la documentación del servidor correspondiente para obtener información sobre cómo configurar una clase de gestión.

Política de retención basada en eventos

En una política de retención basada en eventos, el tiempo de retención de un objeto de archivado no iniciará un evento de negocio, como por ejemplo el cierre de una cuenta bancaria. La retención basada en eventos alinea la política de retención de datos de IBM Spectrum Protect con los requisitos empresariales sobre los datos. Cuando el suceso ocurre, la aplicación envía un suceso **eventRetentionActivate** para el objeto del servidor que va a iniciar la retención.

Para utilizar una política de retención basada en un evento, realice los pasos siguientes:

1. En el servidor, cree una clase de gestión con un archivo **copygroup** de tipoEVENT. Para obtener más información, consulte la documentación del servidor de IBM Spectrum Protect.
2. Consulte la clase de gestión para confirmar que la clase está basada en sucesos. Si la clase de gestión está basada en sucesos, el campo **retainInit** en la estructura**archDetailCG** es ARCH_RETINIT_EVENT.

3. Vincule los objetos a la clase de gestión basada en evento utilizando la inclusión de **archmc**, o explícitamente a través del atributo **mcNameP** en la estructura **ObjAttr** de la llamada de función **dsmSendObj**.
4. Cuando desee iniciar la retención para el objeto, consulte el servidor para todos los objetos que se ven afectados. Compruebe si están en un estado PENDING, y consiga el ID de objeto. En un estado de espera, el campo **retentionInitiated** de la estructura**qryRespArchiveData** indica DSM_ARCH_RETINIT_PENDING.
5. Emita una llamada **dsmBeginTxn** y a continuación emita una llamada **dsmRetentionEvent** con la lista de objetos, seguida de una llamada **dsmEventType**:eventRetentionActivate. Si el número de objetos excede el valor de maxObjPerTxn, utilice varias transacciones.

   Restricción: Puede emitir solo una llamada de **dsmRetentionEvent** por transacción.
6. Consulte los objetos para confirmar que la retención está activada. Si la retención se ha iniciado, el campo **retentionInitiated** de la estructura **qryRespArchiveData** tiene un valor de I.

Consultas para el sistema IBM Spectrum Protect

La API cuenta con varias consultas, como la consulta de clase de gestión, que las aplicaciones pueden utilizar.

Todas las consultas que se utilizan la llamada **dsmBeginQuery** siguen estos pasos:

1. Envíe la llamada **dsmBeginQuery** con el tipo de consulta apropiado:
   - Copia de seguridad
   - Archivado
   - Objetos de copia de seguridad activos
   - Espacio de archivos
   - Clase de gestión

   La llamada **dsmBeginQuery** informa a la API del formato de los datos que se devuelve desde el servidor. Los campos apropiados se pueden colocar en la estructura de datos que pasan por las llamadas **dsmGetNextQObj**. La llamada de inicio de consulta también permite al cliente de la aplicación configurar el alcance de la consulta al especificar adecuadamente los parámetros en la llamada de inicio de consulta.

   Restricción: En sistemas de UNIX o Linux, solo el usuario raíz puede consultar los objetos de archivado y copia de seguridad. Este tipo de consulta se conoce como "fast path".
2. Introduzca la llamada **dsmGetNextQObj** para obtener cada uno de los registros de la consulta. Esta llamada pasa un almacenamiento intermedio lo suficientemente grande para retener los datos que devuelve la consulta. Cada tipo de consulta tiene una estructura de datos correspondiente para los datos que se devuelven. Por ejemplo, el tipo de consulta de copia de seguridad tiene una estructura**qryRespBackupData** asociada que se completa cuando se envía la llamada**dsmGetNextQObj**.
3. La llamada **dsmGetNextQObj** normalmente devuelve uno de los siguientes códigos:
   - DSM_RC_MORE_DATA: vuelva a enviar la llamada **dsmGetNextQObj** de nuevo.
   - DSM_RC_FINISHED: no hay más datos. Envíe la llamada **dsmEndQuery**.

4. Envíe la llamada **dsmEndQuery**. Cuando se recuperan todos los datos o más datos que no son necesarios, escriba la llamada de **dsmEndQuery** para finalizar el proceso de consulta. La API vacía los datos que quedan de la cadena de consulta y libera los recursos que se utilizaron para la consulta.

Figura 9 muestra el diagrama de estado para las operaciones de consulta.

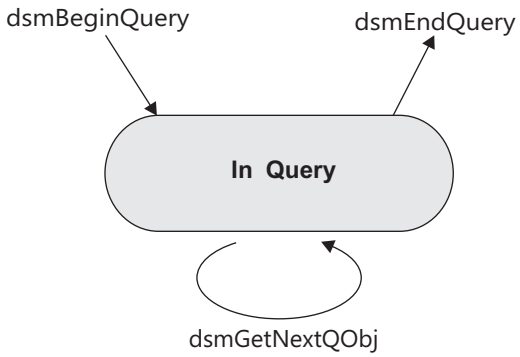


Figura 9. Diagrama de estado para consultas generales

Figura 10 muestra el diagrama de flujo de las operaciones de consulta.

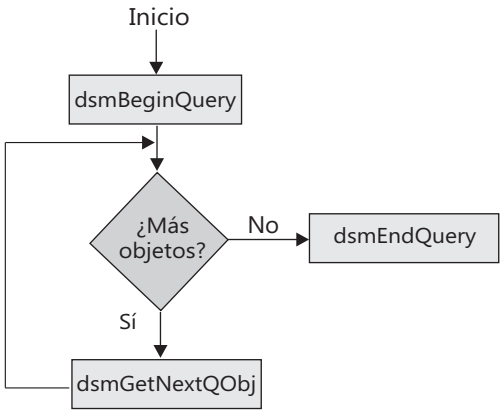


Figura 10. Diagrama de flujo de las consultas generales

Ejemplo de consultas del sistema

En este ejemplo una consulta de clase de gestión copia los valores de todos los campos en los grupos de copia de seguridad y copia archivada para una clase de gestión específica.

```
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:
            .
            .  /* other fields of backup and archive copy groups */
            .
         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);
}
```

Figura 11. Un ejemplo de cómo realizar una consulta de sistema

Eficiencia del servidor

Utilice estas directrices cuando recupere o envíe objetos al servidor de IBM Spectrum Protect.

- Cuando recupere objetos del servidor de IBM Spectrum Protect, siga estas directrices:

- Recupere los dos datos en el orden de restauración que proporciona el servidor de IBM Spectrum Protect. El orden de restauración es especialmente importante para los dispositivos de cinta, porque la recuperación de datos sin ordenar puede provocar rebobinados y montados en la cinta.
- Incluso cuando los datos se almacenan en un dispositivo de disco, puede ahorrar tiempo cuando las recuperaciones están ordenadas.
- Realice todo el trabajo que sea posible en una sola sesión del servidor de IBM Spectrum Protect.
- No inicie y detenga sesiones múltiples.

• Cuando envíe objetos al servidor de IBM Spectrum Protect, siga estas directrices:
- Envíe múltiples objetos en una sola transacción.
- Evite enviar un objeto por transacción, especialmente cuando los datos se envíen directamente a un dispositivo de cinta. Parte de la transacción del dispositivo de cinta consiste en asegurar que los datos de los almacenamientos intermedios de la memoria de acceso aleatorio (RAM) se graben en el soporte.

Conceptos relacionados:

"Seleccionar y ordenar objetos por orden de restauración" en la página 70

Información relacionada:

"Iniciar o finalizar una sesión" en la página 17

Enviar datos a un servidor

La API permite que los clientes de aplicación envíen datos, objetos nombrados y sus datos asociados al almacén del servidor IBM Spectrum Protect.

Consejo: Es posible hacer una copia de seguridad o archivar los datos. Realice todas las operaciones de envía dentro de una misma transacción.

El modelo de transacción

Todos los datos enviados al almacenamiento de IBM Spectrum Protect durante una operación de archivado o copia de seguridad se realiza en una transacción. Un modelo de transacción proporciona un nivel alto de integridad de los datos, pero impone algunas restricciones que un cliente de aplicación debe tener en cuenta.

Iniciar una transacción con una llamada a **dsmBeginTxn** o finalizar una transacción con una llamada a **dsmEndTxn**. Una transacción única es una acción atómica. Los datos que se envían dentro de los límites de una transacción están comprometidos al sistema al final de la transacción o se devuelven si la transacción finaliza de forma prematura.

Las transacciones pueden consistir de envíos únicos de objetos o de múltiples envíos de objetos. Para mejorar el rendimiento del sistema al disminuir la actividad del sistema, envíe objetos pequeños en una transacción múltiple de objetos. El cliente de la aplicación determina si las transacciones únicas o múltiples son adecuadas.

Envíe todos los objetos dentro de una transacción múltiple de objetos al mismo destino de la copia. Si necesita enviar un objeto a distintas ubicaciones que el objeto anterior, finalice la transacción actual e inicie una nueva. Dentro de la transacción nueva, puede enviar el objeto al destino de copia nuevo.

Nota: No se comprueba la coherencia del destino de la copia a los objetos que no contienen ningún bit de datos (*sizeEstimate=0*).

IBM Spectrum Protect limita el número de objetos que se pueden enviar a una transacción de objeto múltiple. Para encontrar este límite, llame a **dsmQuerySessInfo** y examine el campo **maxObjPerTxn**. Este campo muestra el valor de la opción *TXNGroupmax* que está configurada en el servidor.

El cliente de la aplicación debe hacer un seguimiento de los objetos enviados dentro de una transacción para llevar a cabo reintentos de proceso o procesos de error si la transacción finaliza de forma prematura. Tanto el servidor como el cliente pueden detener una transacción en cualquier momento. El cliente de la aplicación debe estar preparado para gestionar finalizaciones de transacción inesperadas que no se iniciaron.

Agregación de archivo

los servidores IBM Spectrum Protect utilizan una función llamada agregación de archivo. Con la agregación de archivo, todos los objetos que se envían en una misma transacción se almacenan juntos, lo cual ahorra espacio y mejora el rendimiento. Aún así es posible consultar y restaurar los objetos de forma separada.

Para utilizar esta función todos los objetos de una transacción deben tener el mismo nombre de espacio de archivo. Si el nombre de espacio de archivo cambia dentro de una transacción, el servidor cierra el objeto agregado existente y empieza uno nuevo.

Transferencia de datos sin LAN

La API puede sacar provecho de la transferencia de datos sin LAN si la opción **dsmSetUp** para las lecturas múltiples es ON. La API devuelve la existencia de un destino sin LAN en la estructura de respuesta **Query Mgmt Class archDetailCG** o en el campo **backupDetailCG bLanFreeDest**.

Puede utilizar operaciones sin LAN en plataformas que están soportadas por el agente de almacenamiento. Se excluye la plataforma Macintosh.

En las siguientes estructuras de salida se proporciona información sin LAN. La estructura (**dsmEndGetDataExOut_t**) de **dsmEndGetData** incluye el campo **totalLFBytesRecv**. Este es el número total de bytes sin LAN que se reciben. La estructura de salida (**dsmEndSendObjExOut_t**) de **dsmEndSendObjEx** incluye el campo **totalLFBytesSent**. Este es el número total de bytes sin LAN que se enviaron.

Información relacionada:

Movimiento de datos sin LAN: descripción general del agente de almacenamiento

Operaciones de grabación simultánea

Puede configurar las agrupaciones del almacenamiento de servidor IBM Spectrum Protect para grabar simultáneamente en una agrupación de almacenamiento primario y copiar la agrupación o agrupaciones de almacenamiento de copias durante una copia de seguridad o archivado. Utilice esta configuración para crear múltiples copias del objeto.

Si una operación de grabación simultánea falla, el código de retorno en la función **dsmEndTxn** puede ser DSM_RC_ABORT_STGPOOL_COPY_CONT_NO, lo cual indica que la grabación a una de las agrupaciones de almacenamiento ha fallado, y la opción de agrupación de almacenamiento COPYCONTINUE de IBM Spectrum Protect se ha establecido en NO. La aplicación termina y el administrador del servidor de IBM Spectrum Protect debe resolver el problema.

Para obtener más información sobre cómo configurar las operaciones de grabación simultánea, consulte la documentación del servidor de IBM Spectrum Protect.

Mejora del rendimiento de la API

Es posible utilizar las opciones de cliente tcpbuffsize y tcpnodelay y el parámetro API **DataBlk** con el fin de mejorar el rendimiento API.

Tabla 10 describe las acciones que puede realizar para mejorar el rendimiento de la API.

Tabla 10. Opciones de archivado/copia de seguridad y el parámetro API que mejora el rendimiento

Opciones de cliente de archivado/copia de seguridad	**Descripción**
tcpbuffsize	Especifica el tamaño de almacenamiento intermedio de TCP. El valor predeterminado es 31 KB. Para mejorar el rendimiento, establezca el valor en 32 KB.
tcpnodelay	Especifica si se deben enviar buffers pequeños al servidor en lugar de retenerlos. Para mejorar el rendimiento, establezca esta opción en *yes* para todas las plataformas. Esta opción es válida sólo para Windows y AIX.
parámetro API	**Descripción**
DataBlk	Este parámetro se utiliza con la llamada de función **dsmSendData** con el fin de determinar el tamaño de almacenamiento intermedio de la aplicación. Para obtener mejores resultados, establezca el parámetro como un valor múltiplo del valor *tcpbuffsize* que se especifica con tcpbuffsize menos 4 bytes. Por ejemplo, establezca un valor de 28 para este parámetro cuando el valor de tcpbuffsize se establece en 32 KB.

Cada llamada **dsmSendData** es síncrona y no regresa hasta que los datos transferidos a la API en **dataBlkPtr** se vacían a la red. La API añade una actividad de 4 bytes a cada almacenamiento intermedio de transacción que se coloca en la red.

Por ejemplo, cuando el tamaño del almacenamiento intermedio de la transacción es de 32 KB y el tamaño del almacenamiento intermedio **DataBlk** de la aplicación es de 31 KB, cada almacenamiento intermedio**DataBlk** de la aplicación cabe en un almacenamiento intermedio de comunicaciones y se vacía de inmediato. No

obstante, si el almacenamiento intermedio **DataBlk** de la aplicación es exactamente 32 KB, y dado que la API añade una actividad de 4 bytes por almacenamiento intermedio de transacción, hay dos vaciados: uno de 32 KB y otro de 4 bytes. Además, si establece la opción `tcpnodelay` en `no`, el vaciado de 4 puede llevar hasta 200 milisegundos.

Configuración de la API para enviar datos de rendimiento al supervisor de rendimiento de cliente

El supervisor de rendimiento de cliente es un componente del Centro de administración de Tivoli Storage Manager que se utiliza para mostrar datos de rendimiento recopilados por la API. El supervisor de rendimiento de cliente registra y muestra los datos de rendimiento para las operaciones de restauración, archivado y copia de seguridad de cliente.

Con la supervisión del rendimiento habilitada, puede mostrar los datos de rendimiento que recopila la API utilizando la supervisión del rendimiento; la supervisión del rendimiento está disponible en el Centro de administración deTivoli Storage Manager. A partir de la versión 7.1, el componente del Centro de administración ya no se distribuye con Tivoli Storage Manager o IBM Spectrum Protect. Si tiene un Centro de administración que se ha instalado con una versión anterior del servidor, puede seguir utilizándola para mostrar los datos de rendimiento. Si no tiene instalado un Centro de administración, puede descargar la versión que ha publicado anteriormente desdeftp://public.dhe.ibm.com/storage/tivoli-storage-management/maintenance/admincenter/v6r3/. Para obtener información sobre cómo utilizar la supervisión de rendimiento, consulte la documentación del servidor de Tivoli Storage Manager versión 6.3.

Configuración de las opciones del supervisor de rendimiento de cliente

Habilite los clientes de IBM Spectrum Protect para utilizar la supervisión del rendimiento especificando los parámetros en el archivo de opciones del cliente. Especifique estas opciones para cada cliente que desee supervisar.

Cuando supervise el rendimiento en equipos con UNIX yLinux, ajuste el límite del descriptor del archivo a al menos 1024, utilizando el siguiente mandato:

```
ulimit -n 1024
```

Para configurar las opciones de supervisión del rendimiento, siga estos pasos:

1. Abra el archivo de opciones del cliente para cada cliente que esté supervisando. Dependiendo de la configuración, las opciones del cliente se encuentran en uno de los siguientes archivos:
   - `dsm.opt`
   - `dsm.sys`
2. Añada las siguientes opciones al archivo de opciones del cliente:

   ```
   PERFMONTCPSERVERADDRESS
   PERFMONTCPPORT
   PERFMONCOMMTIMEOUT
   ```

PERFMONTCPSERVERADDRESS

La opción PERFMONTCPSERVERADDRESS especifica el nombre de host o dirección IP del sistema donde está instalado el supervisor de rendimiento de cliente.

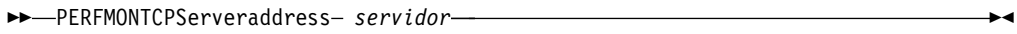
Clientes soportados

Esta opción es independiente de la plataforma y es compatible con todos los clientes.

Archivo de opciones

Establezca esta opción en el archivo de opciones del cliente (`dsm.opt` o `dsm.sys`).

Sintaxis



Parámetros

servidor
El nombre de host del servidor o dirección IP del sistema que tiene instalado el supervisor de rendimiento de cliente (es el mismo servidor que ejecuta el Centro de administración).

Ejemplos

Archivo de opciones:

```
PERFMONTCPSERVERADDRESS 131.222.10.5
```

Línea de mandatos:

Esta opción no se puede establecer mediante la línea de mandatos.

PERFMONTCPPORT

El número de puerto en que el supervisor de rendimiento de cliente está a la escucha de datos de rendimiento de los clientes.

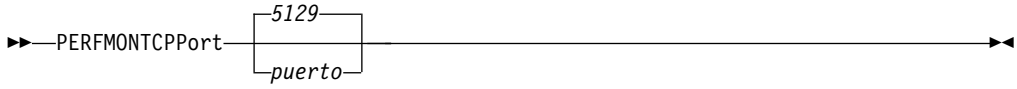
Clientes soportados

Esta opción es independiente de la plataforma y es compatible con todos los clientes.

Archivo de opciones

Establezca esta opción en el archivo de opciones del cliente (`dsm.opt` o `dsm.sys`).

Sintaxis



Parámetros

puerto
El puerto que se supervisa para los datos de rendimiento de cliente. El puerto 5129 es el puerto predeterminado.

Ejemplos

Archivo de opciones:

```
PERFMONTCPPPORT 5000
```

Línea de mandatos:
Esta opción no se puede establecer mediante la línea de mandatos.

PERFMONCOMMTIMEOUT

Especifica el tiempo máximo, en segundos, que la llamada dsmTerminate espera a que lleguen datos de rendimiento antes de finalizar una sesión.

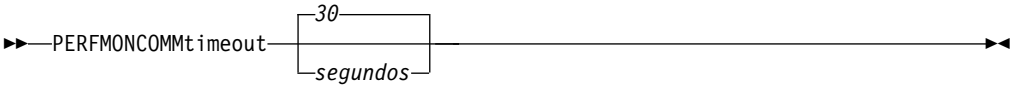
Clientes soportados

Esta opción es independiente de la plataforma y es compatible con todos los clientes.

Archivo de opciones

Establezca esta opción en el archivo de opciones del cliente (`dsm.opt` o `dsm.sys`).

Sintaxis



Parámetros

segundos
El tiempo que se debe esperar a que lleguen los datos de rendimiento restantes, antes de finalizar la sesión.

Ejemplos

Archivo de opciones:

```
PERFMONCOMMTIMEOUT 60
```

Línea de mandatos:
Esta opción no se puede establecer mediante la línea de mandatos.

Enviar objetos al servidor

Los clientes de aplicaciones pueden enviar datos o objetos nombrados y sus datos asociados a un almacenamiento IBM Spectrum Protect utilizando funciones de archivado y de copia de seguridad de API. Los componentes de copia de seguridad y archivado del sistema permiten el uso de procedimientos de gestión diferentes para los datos que se envían a almacenamiento.

El atributo de estimación de tamaño es una estimación del tamaño total del objeto de datos que se envían al servidor. Si la aplicación no conoce el tamaño del objeto exacto, configure *sizeEstimate* en una estimación más alta. Si la estimación es más

pequeña que el tamaño real, el servidor IBM Spectrum Protect utiliza recursos extra para gestionar las asignaciones de espacio extra.

Sugerencias:

- Sea tan preciso como sea posible cuando realice esta estimación de tamaño. El usuario utiliza este atributo para la asignación eficaz de espacio y la colocación de objetos dentro de sus recursos de almacenamiento.
- Si el cálculo es menor al tamaño actual, un servidor con almacenamiento en memoria caché no asigna espacio adicional y detiene el envío.

Es posible que tenga problemas si *sizeEstimate* es excesivo. Es posible que el servidor no disponga de suficiente espacio para el tamaño estimado pero tenga espacio para el tamaño real; o es posible que el servidor utilice dispositivos más lentos.

Puede realizar copias de seguridad o archivar objetos que son más grandes que dos gigabytes de tamaño. Los objetos pueden estar comprimidos o descomprimidos.

Para iniciar una operación de envío, llame a **dsmSendObj**. Si tiene más datos de los que pueda enviar en una única operación, puede repetir las llamadas a **dsmSendData** con el fin de transferir el resto de la información. Llame a **dsmEndSendObj** para finalizar la operación de envío.

Conceptos básicos sobre archivar y realizar copias de seguridad de los objetos

El componente de respaldo del sistema IBM Spectrum Protect es compatible con varias versiones de objetos nombrados que se almacenan en el servidor.

Cualquier objeto respaldado en el servidor que tiene el mismo nombre que un objeto que ya está almacenado en el servidor del mismo cliente está sujeto a un control de versión. Los objetos pueden tener un estado activo o inactivo en el servidor. La última copia de un objeto en el servidor que no ha sido desactivada se encuentra en estado activo. Cualquier otro objeto con el mismo nombre, ya sea una versión anterior o una copia desactivada, se considera inactiva. Las construcciones de clase de gestión definen los distintos criterios de gestión. Se asignan a objetos activos e inactivos en el servidor.

Tabla 11 muestra una lista de los campos del grupo de copia que se aplican a estados activos e inactivos:

Tabla 11. Campos de grupo de copia de seguridad

Campo	Descripción
VEREXISTS	El número de versiones inactivas si existen versiones activas.
VERDELETED	El número de versiones inactivas si no existen versiones activas.
RETEXTRA	El número de días que se conservan las versiones inactivas.
RETONLY	El número de días que se conservan las últimas versiones inactivas si no existen versiones activas.

Si cada una de las versiones de copia de seguridad tienen un nombre único, como por ejemplo un sello de hora en el nombre, entonces la asignación de versión no ocurre de forma automática: todos los objetos están activos. Los objetos activos nunca caducan, con lo que una aplicación es responsable de desactivarlos con la

llamada **dsmDeleteObj**. En este caso, la aplicación necesita que los objetos desactivados caduquen lo antes posible. El usuario define un grupo de copia de seguridad con VERDELETED=0 y RETONLY=0.

El componente de archivado del sistema IBM Spectrum Protect permite que los objetos estén almacenados en el servidor con controles de caducidad o retención en lugar de con controles de versión. Cada uno de los objetos almacenados son únicos, aunque el nombre de cualquiera de ellos sea el mismo que el de un objeto que ya está archivado. Los objetos archivados tienen un campo de descripción asociado con los metadatos que se pueden utilizar durante una consulta con el fin de identificar un objeto específico.

A cada objeto del servidor IBM Spectrum Protect se le asigna un ID de objeto exclusivo. La persistencia del valor original no está garantizado durante el ciclo de vida de un objeto (especialmente, después de una exportación o importación). Por lo tanto, una aplicación no debe consultar y guardar el ID del objeto original para utilizarlo en restauraciones posteriores. En su lugar, una aplicación debe guardar el nombre del objeto e introducir la fecha. Puede utilizar esta información durante una restauración para consultar objetos y verificar la fecha de introducción. A continuación, el ID de objeto actual se puede utilizar para restaurar el objeto.

Compresión

Las opciones de configuración en un nodo determinado y la opción **dsmSendObj** `objCompressed` , determinan si IBM Spectrum Protect comprime el objeto durante un envío. También, los objetos con un `sizeEstimate` inferior a `DSM_MIN_COMPRESS_SIZE` nunca se comprimen.

Si el objeto ya está comprimido (`objCompressed=bTrue`), no se comprime de nuevo. Si no está comprimido, IBM Spectrum Protect decide si comprimir el objeto, en función de los valores de la opción de compresión establecida por el administrador y que esté establecida en los orígenes de configuración de la API.

El administrador puede cambiar los umbrales de compresión en el servidor utilizando el mandato register node (`compression=yes`, `no` o `client-determined`). Si esto está `determinado por el cliente`, entonces el comportamiento de la compresión está determinado por el valor de opción de compresión en las fuentes de configuración.

Algunos tipos de datos, como los datos que ya están comprimidos, pueden crecen cuando se procesan con el algoritmo de compresión. Cuando esto ocurre, se genera el código de retorno DSM_RC_COMPRESS_GREW. Si se da cuenta de que esto ocurre, pero desea que la operación de envío continúe de todas formas, comuníqueselo a los usuarios finales con el fin de especificar la siguiente opción en los archivos de opciones:

```
COMPRESSAlways Yes
```

S, durante una función **dsmSendData** , con la compresión habilitada, obtiene el código de retorno DSM_RC_COMPRESS_GREW, es posible que desee iniciar y enviar el objeto de nuevo sin compresión. Para implementarlo, configure **dsmSendObj** `ObjAttr.objCompressed` en `bTrue`.

La información sobre el comportamiento de compresión durante **dsmSendObj** se devuelve por la llamada **dsmEndSendObjEx**. `objCompressed` especifica si se realizó la compresión. `totalBytesSent` es el número de bytes enviados por la aplicación.

totalCompressedSize es el número de bytes después de la compresión. La llamada **dsmEndSendObjEx** también tiene un campo totalLFBytesSent que contiene el total de bytes enviados sin LAN.

Atención: Si la aplicación va a utilizar restauraciones o recuperaciones parciales de objetos, no es posible comprimir los datos mientras los envía. Para implementarlo, configure **dsmSendObj** ObjAttr.objCompressed en bTrue.

Tipo de compresión

El tipo de compresión que el cliente utiliza se determina mediante la combinación de compresión y la deduplicación de datos del lado del cliente que se utiliza durante el proceso de copia de seguridad y archivado.

El algoritmo de compresión que utiliza el cliente lo notifica la API en un nuevo campo en las estructuras **qryRespArchiveData** y **qryRespBackupData**:

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

Se notifican los siguientes tipos de compresión:

LZ4 Un método de compresión más rápido y eficaz que utiliza el cliente cuando se envía un objeto deduplicado por el cliente a una agrupación de almacenamiento de contenedores compatible con LZ4 en el servidor IBM Spectrum Protect. El servidor debe tener la versión 7.1.5 o posterior, y debe utilizar agrupaciones de almacenamiento de contenedores. La compresión LZ4 del lado del cliente se utiliza solo cuando está habilitada la deduplicación de datos del lado del cliente.

LZW Un tipo tradicional de compresión que el cliente utiliza en cualquiera de las siguientes situaciones:

- Los objetos deduplicados por el cliente se envían a agrupaciones de almacenamiento tradicionales (no de contenedores) del servidor.
- El objeto de cliente no experimenta deduplicación de datos del lado del cliente.
- El objeto de cliente experimenta solo deduplicación de datos tradicional del lado del servidor.

Campo en blanco
El objeto no está comprimido por el cliente. El objeto no está comprimido porque la opción compression está establecida en *no*, o no se ha especificado la opción durante el proceso de copia de seguridad o archivado. Aunque el objeto no está comprimido por el cliente, puede que el servidor lo comprima.

El tipo de compresión no es configurable. Viene determinada por el cliente de archivado y copia de seguridad en el momento del proceso de copia de seguridad o archivado.

Ejemplo

El siguiente ejemplo muestra el campo Tipo de compresión de la salida de las consultas de copia de seguridad y archivado de una aplicación de ejemplo de 64 bits **dapismp**:

```
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
```

Eliminación de copia de almacenamiento intermedio

La función de eliminación de copia de almacenamiento intermedio elimina la copia de los almacenamientos intermedios entre una aplicación y el servidor IBM Spectrum Protect, lo que da como resultado una mejor utilización del procesador. Para un efecto máximo, utilice este método en un entorno sin LAN.

Los almacenamientos intermedio para el movimiento de datos los asignaIBM Spectrum Protect y se pasa un puntero de vuelta a la aplicación. La aplicación coloca los datos en el almacenamiento intermedio proporcionado, y dicho almacenamiento intermedio pasa a través de las capas de comunicación al agente de almacenamiento (utilizando la memoria compartida). A continuación los datos pasan al dispositivo de cinta, el cual elimina las copias de los datos. Esta función se puede utilizar con operaciones de copia de seguridad o de archivado.

Atención: Cuando utilice este método, preste especial atención a la gestión adecuada del almacenamiento intermedio y a los tamaños de estos. Los buffers se comparten entre los componentes y cualquier sobrescritura de memoria que es el resultado de un error de programación se convierte en un error importante.

La secuencia general de llamadas para archivados/copias de seguridad es la siguiente:

```
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
```

La función **dsmRequestBuffer** se puede llamar varias veces, hasta el valor que especifica la opción numTsmBuffers. Una aplicación puede tener dos hebras: una hebra de producción que rellena los buffers con datos y una hebra de consumo que envía dichos almacenamientos intermedios a IBM Spectrum Protect con la llamada **dsmSendBufferData**. Cuando se emite una llamada **dsmRequestBuffer** y se ha alcanzado **numTsmBuffers**, la llamada **dsmRequestBuffer** se bloquea hasta que se libera un almacenamiento intermedio. La liberación del almacenamiento intermedio se puede producir llamando a **dsmSendBufferData**, que envía y libera un almacenamiento intermedio o llamando a **dsmReleaseBuffer**. Para obtener más información, consulte callbuff.c en el directorio de muestra de la API.

Si en algún momento hay un fallo en el envío, la aplicación debe liberar todos los buffers retenidos y terminar la sesión. Por ejemplo:

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

Si una aplicación llama a **dsmTerminate** y aún está retenido el almacenamiento intermedio, la API no existe. Se devuelve el siguiente código: DSM_RC_CANNOT_EXIT_MUST_RELEASE_BUFFER. Si la aplicación no puede liberar el almacenamiento intermedio, debe salir del proceso para forzar una limpieza.

Eliminación de la copia de almacenamiento intermedio y restauración y recuperación

El servidor IBM Spectrum Protect controla la cantidad de datos que se colocan en almacenamiento intermedio basándose en la optimización del acceso de cintas con restauración y recuperación. Este método no es beneficioso para la aplicación como método normal de obtención de datos. Durante la creación de un prototipo, compruebe el rendimiento del método de eliminación de copia de almacenamiento intermedio y utilice este método sólo si ve una mejora que merece la pena.

La cantidad máxima de datos en un almacenamiento intermedio único que devuelve el servidor IBM Spectrum Protect es (256K bytes – sobrecarga de cabecera). Como consecuencia, sólo las aplicaciones que tratan con pequeñas grabaciones de almacenamiento intermedio se benefician de este mecanismo de recuperación de datos. La aplicación debe prestar una atención especial al número de bytes en el almacenamiento intermedio proporcionado, dependiendo del tamaño del objeto, la red y de otras condiciones de limitación. En algunos casos, el uso de la eliminación de copia de almacenamiento intermedio puede realizarse peor que una restauración normal. La API normalmente capta los datos y devuelve una longitud fija a la aplicación. La aplicación puede controlar el número de grabaciones de datos devueltos al disco.

Si utiliza la eliminación de copia de seguridad de almacenamiento intermedio, cree un mecanismo de captación de datos para almacenamientos intermedios que tienen un tamaño menor al tamaño del almacenamiento intermedio de la grabación preferida. Por ejemplo, si una aplicación escribe bloques de datos de 64 K en disco, la aplicación debe adoptar estas acciones:

1. Llamar a **dsmGetBufferData**.
2. Grabar bloques de 64K.

3. En el bloque final, copie el resto a un **tempBuff**, emita otra llamada a **dsmGetBufferData** y rellene el **tempBuff** con el resto de los datos.
4. Continúe grabando bloques de 64K:

```
dsmGetBufferData #1 get 226K    dsmGetBufferData #2 get 240K
Block1 64K - grabar en disco          Block1 30K - copiar a tempbuff-grabar en disco
Block2 64K - grabar en disco          Block2 64K - grabar en disco
Block3 64K - grabar en disco          Block3 64K - grabar en disco
Block4 34K - copiar a tempbuff        Block4 64K - grabar en disco
Block5 18K - grabar en tempbuff       etc
```

En este ejemplo, seis grabaciones de disco son directas y 1 está almacenada en memoria.

La secuencia general de llamadas para restauración y recuperación es de la siguiente manera:

dsmInitEx (UseTsmBuffers = True numTsmBuffers = la cantidad de almacenamientos intermedios que la aplicación desea asignar).

```
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
```

Por cada llamada **dsmGetBufferData** implemente una llamada **dsmReleaseBuffer**. **dsmGetBufferData** y su correspondiente **dsmReleaseBuffer** no necesitan ser consecutivos. Una aplicación puede emitir varias llamadas**dsmGetBufferData** primero para obtener varios almacenamientos intermedios, y a continuación emitir las llamadas correspondientes **dsmReleaseBuffer** más tarde. Para obtener el código de ejemplo que utiliza esta función, consulte `callbuff.c` en el directorio de muestra de la API.

Restricción: Puesto que la API proporciona el almacenamiento intermedio y el objetivo es minimizar la utilización del procesador, no se permite más procesamiento de datos en el almacenamiento intermedio. La aplicación no puede utilizar el cifrado y la compresión con la eliminación de copia de almacenamiento intermedio porque ambas operaciones requieren el procesamiento y copias de datos.

Implemente tanto la ruta de movimiento de datos regular como la eliminación de copia de almacenamiento intermedio para habilitar al usuario el cambio entre ambas rutas, de acuerdo a sus necesidades. Si el usuario tiene que comprimir o cifrar los datos, utilice el mecanismo existente. Si hay una restricción de procesador, utilice el nuevo mecanismo. Ambos mecanismos son complementarios y no se sustituyen uno a otro completamente.

Cifrado de la API

Existen dos métodos de cifrado disponibles para el cifrado de datos: el cifrado gestionado por la aplicación y el cifrado del cliente de IBM Spectrum Protect.

Seleccione y utilice solo uno de estos métodos para cifrar los datos. Los métodos se excluyen mutuamente y si cifra los datos mediante ambos métodos no podrá restaurar o recuperar algunos de los datos. Por ejemplo, presuponga que una aplicación utiliza el cifrado gestionado por aplicación para cifrar el objeto A y, a continuación, utiliza el cifrado del cliente de IBM Spectrum Protect para cifrar el objeto B. Durante la operación de restauración, si la aplicación establece la opción para utilizar el cifrado del cliente de IBM Spectrum Protect e intenta restaurar ambos objetos, solo se puede restaurar el objeto B; el objeto A no se puede restaurar porque lo ha cifrado la aplicación y no el cliente.

Independientemente del método de cifrado que se utilice, elIBM Spectrum Protect debe habilitar el cifrado de la contraseña. De forma predeterminada, el servidor utiliza`SET AUTHENTICATION ON`.

La API utiliza el cifrado AES de 128 bits o AES de 256 bits. El cifrado AES de 256 bits proporciona un cifrado de datos más robusto que el cifrado de datos AES de 128 bits. Los archivos de los que se realiza una copia de seguridad utilizando el cifrado AES de 256 bits no se pueden restaurar con un cliente más antiguo. El cifrado se puede activar con o sin compresión. Si utiliza el cifrado, no se pueden utilizar las funciones de restauración y recuperación de objeto parcial y eliminación de copia de almacenamiento intermedio.

Cifrado gestionado por la aplicación

Con el cifrado gestionado por la aplicación, la aplicación proporciona la contraseña clave al API (utilizando la clave DSM_ENCRYPT_USER) y es responsabilidad de la aplicación gestionar la contraseña clave.

Atención: Si no se ha guardado la clave de cifrado, y ha olvidado la clave, no se podrán recuperar los datos.

La aplicación proporciona la contraseña clave en la llamada **`dsmInitEx`** y debe proporcionar la contraseña adecuada cuando llegue el momento de la restauración.

Atención: Si la contraseña clave se pierde, no hay forma de restaurar los datos.

Debe usarse la misma contraseña de clave para las operaciones de copia de seguridad y restauración (o archivado y recuperación) del mismo objeto. Este método no depende del nivel del servidor IBM Spectrum Protect. Para configurar este método, la aplicación necesita realizar lo siguiente:

1. Configure la variable `bEncryptKeyEnabled` en `bTrue` en la llamada a **`dsmInitEx`**, y configure la variable`encryptionPasswordP` para que apunte a una cadena con la contraseña clave de cifrado.
2. Configure `include.encrypt` para que los objetos `encrypt`. Por ejemplo, para cifrar todos los datos, configure:

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

   e

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

   Para cifrar el objeto `/FS1/DB2/FULL`, defina:

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

3. Configure ENCRYPTKEY=PROMPT|SAVE en la cadena de opción que pasa al API en la llamada**dsmInitEx** en Windows. Esta opción también se puede establecer en dsm.opt (Windows) o dsm.sys (UNIX o Linux).

De forma predeterminada, la opción encryptkey se configura en prompt. Este valor garantiza que la clave no se almacena automáticamente. Si se especifica guardar conencryptkey, IBM Spectrum Protect almacena la clave en la máquina local, pero en dicho caso sólo una clave es válida para todas las operaciones IBM Spectrum Protect con el mismo nombre de nodo.

Después del envío de un objeto, **dsmEndSendObjEx** especifica si un objeto ha sido cifrado y qué método se ha utilizado. Los valores posibles en el campo son *encryptionType*:

- DSM_ENCRYPT_NO
- DSM_ENCRYPT_USER
- DSM_ENCRYPT_CLIENTENCRKEY

En la siguiente tabla se incluyen los tipos de cifrado de la API, requisitos previos y las funciones disponibles.

Tabla 12. Tipos de cifrado de API, requisitos previos y funciones disponibles

Tipo	Requisito previo	Función disponible
ENCRYPTIONTYPE	Ninguno	Configure ENCRYPTIONTYPE en la cadena de opción que pasa al API en la llamada **dsmInitEx** en Windows. ENCRYPTIONTYPE es AES128 de forma predetermina da.
EncryptKey=save	Ninguno	API y copias de seguridad/archivados
EncryptKey=prompt	Ninguno	API y copias de seguridad/archivados
EncryptKey=generate	Ninguno	API y copias de seguridad/archivados
EnableClientEncryptKey	Ninguno	Sólo API

Nota: Se recomienda que la autenticación del servidor esté en ENCENDIDO. Si la autenticación no está activada OFF, la clave no se cifra, pero los datos siguen cifrados. No obstante, esto no se recomienda.

Tabla 13 muestra cómo tanto los usuarios autorizados como los no autorizados pueden cifrar y descifrar datos durante una operación de restauración o copia de seguridad, dependiendo del valor especificado en la opción passwordaccess. El archivo TSM.PWD debe existir para realizar las siguientes operaciones de usuarios autorizados y no autorizados. El usuario autorizado crea el archivo TSM.PWD y configura la opción encryptkey en guardar y la opción passwordaccess en generar.

Tabla 13. Cifrar o descifrar datos con la clave gestionada por la aplicación en UNIX o Linux

Operación	opción passwordaccess	opción encryptkey	Resultado
Copia de seguridad de usuario autorizado	generate	save	Datos cifrados.
	generate	prompt	Datos cifrados si encryptionPasswordP contiene una contraseña cifrada.
	prompt	save	Datos cifrados si encryptionPasswordP contiene una contraseña cifrada.
	prompt	prompt	Datos cifrados si encryptionPasswordP contiene una contraseña cifrada.

Tabla 13. Cifrar o descifrar datos con la clave gestionada por la aplicación en UNIX o Linux (continuación)

Operación	opción passwordaccess	opción encryptkey	Resultado
Restauración de usuario autorizado	generate	save	Datos cifrados.
	generate	prompt	Datos cifrados si encryptionPasswordP contiene una contraseña cifrada.
	prompt	save	Datos cifrados si encryptionPasswordP contiene una contraseña cifrada.
	prompt	prompt	Datos cifrados si encryptionPasswordP contiene una contraseña cifrada.
Copia de seguridad de usuario no autorizado	generate	save	Datos cifrados.
	generate	prompt	Datos cifrados si encryptionPasswordP contiene una contraseña cifrada.
	prompt	save	Datos cifrados si encryptionPasswordP contiene una contraseña cifrada.
	prompt	prompt	Datos cifrados si encryptionPasswordP contiene una contraseña cifrada.
Restauración de usuario no autorizado	generate	save	Datos cifrados.
	generate	prompt	Datos cifrados si encryptionPasswordP contiene una contraseña cifrada.
	prompt	save	datos cifrados si encryptionPasswordP contiene una contraseña cifrada
	prompt	prompt	Datos cifrados si encryptionPasswordP contiene una contraseña cifrada.

Cifrado del cliente de IBM Spectrum Protect

IBM Spectrum Protect el cifrado de cliente utiliza la clave gestionada por el valor DSM_ENCRYPT_CLIENTENCRKEY para proteger los datos. El cifrado de cliente es transparente para la aplicación que utiliza la API, con excepción de que las operaciones de restauración y recuperación de objetos parciales no son viables para objetos cifrados o comprimidos.

Tanto para el cifrado de cliente de IBM Spectrum Protect como para el cifrado gestionado mediante la aplicación, la contraseña de cifrado hace referencia a un valor de serie que se utiliza para generar la clave de cifrado actual. El valor para la opción de contraseña de cifrado es de 1 a 63 caracteres de longitud, pero la clave generada siempre es de 8 bytes para 56 DES, de 16 bytes para 128 AES y de 32 bytes para 256 AES.

Atención: Si la clave de cifrado no está disponible, los datos no se pueden restaurar o recuperar. Cuando utiliza ENABLECLIENTENCRYPTKEY para el cifrado, la clave de cifrado se guarda en la base de datos del servidor. Para los objetos que utilizan este método, la base de datos del servidor debe existir y contar con los valores adecuados para los objetos para una restauración adecuada. Asegúrese de que realiza copias de seguridad de la base de datos del servidor con frecuencia para evitar que se pierdan los datos.

Este es el método más simple de implementar, cuando una clave de cifrado aleatoria se genera por sesión y se almacena en el servidor de IBM Spectrum Protect con el objeto en la base de datos del servidor. Durante la restauración, la clave almacenada se utiliza para el descifrado. Con este método IBM Spectrum Protect es responsable de la gestión de la clave, y la aplicación no tiene que ocuparse de la clave. Como la clave se almacena en la base de datos del servidor, debe tener una base de datos de IBM Spectrum Protect válida para una operación

de restauración de un objeto cifrado. Cuando la clave se transmite entre la API y el servidor, también está cifrada. La transmisión de la clave es segura, y cuando la clave se almacena en la base de datos del servidor de IBM Spectrum Protect está cifrada. La única vez que la clave se destapa con la corriente de datos de exportación es cuando los datos del nodo se exportan entre servidores.

Para habilitar el cifrado de cliente de IBM Spectrum Protect, lleve a cabo los siguientes pasos:

1. Especifique `-ENABLECLIENTENCRYPTKEY=YES` en la cadena de la opción que se pasa a la API en la llamada dsmInitEx o que define el archivo de opciones del sistema `dsm.opt` (Windows) o `dsm.sys` (UNIX o Linux).
2. Configure `include.encrypt` en los objetos que va a cifrar. Por ejemplo, para cifrar todos los datos, configure:

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

   e

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

   Para cifrar el objeto `/FS1/DB2/FULL`, defina:

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

Desduplicación de datos

La desduplicación de datos es un método para reducir las necesidades de almacenamiento eliminando datos redundantes.

Visión general

En IBM Spectrum Protect hay disponibles dos tipos de eliminación de datos duplicados: *eliminación de datos duplicados del lado del cliente* y *eliminación de datos duplicados del lado del servidor*.

Optimización de almacenamiento de datos del lado de cliente es una técnica de optimización de almacenamiento de datos que se utiliza en el cliente de archivado y copia de seguridad para eliminar datos redundantes durante el proceso de copia de seguridad y archivado antes de que los datos se transfieran al servidor de IBM Spectrum Protect. El uso de la eliminación de datos duplicados del lado del cliente puede reducir la cantidad de datos que va a enviarse a través de una red de área local.

La eliminación de datos duplicados del lado del servidor es una técnica de eliminación de datos duplicados realizada por el servidor. El administrador de IBM Spectrum Protect puede especificar la ubicación de la eliminación de datos duplicados (cliente o servidor) a utilizar con el parámetro **DEDUP** en el mandato del servidor **REGISTER NODE** o **UPDATE NODE**.

Mejoras

Con la optimización de almacenamiento de datos del lado del cliente, puede:

- Excluir archivos específicos en un cliente de una eliminación de duplicados de datos.
- Habilitar una memoria caché de eliminación de duplicados de datos que reduzca el tráfico de la red entre el cliente y el servidor. La memoria caché contiene las extensiones que se enviaron al servidor en operaciones anteriores de copia de seguridad incremental. En lugar de consultar al servidor la existencia de una extensión, el cliente consulta su memoria caché.

Especificar un tamaño y ubicación para una memoria caché de cliente. Si se detecta una incoherencia entre el servidor y la memoria caché local, la memoria caché local se elimina y se vuelve a rellenar.

Nota: Para las aplicaciones que utilizan la API de IBM Spectrum Protect, la memoria caché de eliminación de datos duplicados no se debe utilizar debido a la posibilidad de que se produzcan errores en las copias de seguridad causados por la falta de sincronización de la memoria caché con el servidor de IBM Spectrum Protect. Si hay configuradas varias sesiones de cliente de archivado y copia de seguridad simultáneas, debe haber una caché independiente configurada para cada sesión.

- Habilitar la eliminación de duplicados de datos y la compresión del lado del cliente para reducir la cantidad de datos almacenados por el servidor. Todas las extensiones se comprimen antes de ser enviadas al servidor. Se establece un compromiso entre el ahorro de almacenamiento y la potencia de procesamiento necesaria para comprimir los datos de cliente. En general, si se comprime y optimiza el almacenamiento de datos en el sistema cliente, se utiliza aproximadamente el doble de potencia de procesamiento que la optimización de almacenamiento de datos por sí sola.

  El servidor también puede funcionar con datos comprimidos o a los que se ha aplicado la optimización de almacenamiento. Además, los clientes de copia de seguridad y archivado anteriores a V6.2 pueden restaurar los datos comprimidos y con optimización de almacenamiento.

La eliminación de datos duplicados del lado del cliente utiliza el siguiente proceso:

- El cliente crea extensiones. Las *extensiones* son partes de archivos que se comparan con otras extensiones de archivos para identificar duplicados.
- El cliente y el servidor colaboran para identificar las extensiones duplicadas. El cliente envía extensiones no duplicadas al servidor.
- Las operaciones subsiguientes de eliminación de duplicados de datos del cliente crean nuevas extensiones. Es posible que algunas o todas las extensiones coincidan con las extensiones que se crearon en operaciones anteriores de optimización de almacenamiento de datos y que se enviaron al servidor. Las extensiones coincidentes no vuelven a enviarse al servidor.

Beneficios

La optimización de almacenamiento de datos del lado del cliente ofrece varias ventajas:

- Puede reducir la cantidad de datos enviados a la red de área local (LAN).
- La potencia de procesamiento necesaria para identificar datos duplicados se descarga desde el servidor a los nodos cliente. La optimización de almacenamiento de datos del lado del servidor siempre está habilitada para las agrupaciones de almacenamiento preparadas para optimización de almacenamiento. Sin embargo, los archivos que están en agrupaciones de almacenamiento preparadas para optimización de almacenamiento y en las que el cliente haya realizado la optimización de almacenamiento no necesitan procesamiento adicional.
- Se elimina la potencia de procesamiento necesaria para suprimir los datos duplicados en el servidor, lo que produce un ahorro de espacio en el servidor inmediato.

La eliminación de duplicados de datos del lado del cliente tiene una desventaja posible. El servidor no tiene copias completas de los archivos del cliente *hasta* que

se realiza una copia de seguridad de las agrupaciones de almacenamiento primarias que contienen extensiones de cliente en una agrupación de almacenamiento de copia sin optimización de almacenamiento. (Las *extensiones* son partes de un archivo que se crean durante el proceso de optimización de almacenamiento de datos). Durante la copia de seguridad de la agrupación de almacenamiento en una agrupación de almacenamiento sin eliminación de datos duplicados, las extensiones del cliente se vuelven a ensamblar en archivos contiguos.

De forma predeterminada, debe realizarse una copia de seguridad de las agrupaciones de almacenamiento primarias de acceso secuencial que están configuradas para optimización de almacenamiento de datos en agrupaciones de almacenamiento de copia sin optimización de almacenamiento antes de poder recuperarlas y antes de poder eliminar datos duplicados. La opción predeterminada garantiza que el servidor tenga copias de archivos completos en todo momento, en una agrupación de almacenamiento primaria o en una agrupación de almacenamiento de copia.

Importante: Para conseguir una mayor reducción de datos, puedes habilitar la eliminación de datos duplicados del lado del cliente y la compresión a la vez. Todas las extensiones se comprimen antes de ser enviadas al servidor. La compresión ahorra espacio, pero aumenta el tiempo de procesamiento en la estación de trabajo del cliente.

Las siguientes opciones están relacionadas con la eliminación de duplicados de datos:

- Eliminación de duplicados
- Dedupcachepath
- Dedupcachesize
- Enablededupcache
- Exclude.dedup
- Include.dedup

Eliminación de datos duplicados del lado del cliente de la API

La API utiliza *la eliminación de datos duplicados del lado del cliente* en el cliente de archivado y copia de seguridad para eliminar datos redundantes durante el proceso de copia de seguridad y archivado antes de que los datos se transfieran al servidor IBM Spectrum Protect.

La API utiliza la eliminación de datos duplicados del lado del cliente para eliminar los datos redundantes durante el proceso de copia de seguridad y archivado antes de que se transfieran los datos al servidor de IBM Spectrum Protect. El uso de la eliminación de datos duplicados del lado del cliente puede reducir la cantidad de datos que va a enviarse a través de una red de área local. El uso de la eliminación de datos duplicados del lado del cliente también puede reducir el espacio del almacenamiento del servidor IBM Spectrum Protect.

Cuando el cliente está habilitado para la eliminación de datos suplicados del lado del cliente, y se realiza una operación de copia de seguridad o archivado, los datos se envían al servidor en forma de extensiones. La siguiente vez que se realiza una operación de copia de seguridad o archivado, el cliente y el servidor identifican qué extensiones de datos ya se han archivado o de cuáles ya se ha hecho una copia de seguridad y solo se envían al servidor las extensiones de datos únicas.

Para la deduplicación de datos del lado del cliente, el servidor y la API tienen que estar en la versión 6.2 o posterior.

Antes de utilizar la eliminación de duplicado de datos para realizar copias de seguridad o archivar archivos, el sistema tienen que cumplir los siguientes requisitos:

- El cliente tiene que tener la opción `deduplication` habilitada.
- El servidor tiene que habilitar el cliente para la eliminación de datos duplicados del lado del cliente con el parámetro **DEDUP=CLIENTORSERVER** en el mandato **REGISTER NODE** o **UPDATE NODE**.
- El destino de la agrupación de almacenamiento de los datos tiene que ser una agrupación de almacenamiento con eliminación de duplicados de datos habilitada. La agrupación de almacenamiento con eliminación de duplicados de datos habilitada es solo de tipo de dispositivo de archivo.
- Asegúrese de que los archivos están vinculados a la clase de gestión correcta.
- Un archivo puede excluirse del proceso de eliminación de datos suplicados del lado del cliente. De manera predeterminada, se incluyen todos los archivos.
- Los archivos tienen que superar los 2 KB.
- El servidor puede limitar el tamaño de transacción máximo para la eliminación de datos duplicados definiendo la opción `CLIENTDEDUPTXNLIMIT` en el servidor. Consulte la información disponible en la documentación del servidor sobre esta opción.

Si no se cumple alguno de estos requisitos, los datos se procesan de forma normal, sin que se lleve a cabo la eliminación de datos duplicados del lado del cliente.

A continuación se detallan algunas restricciones de la eliminación de datos duplicados:

- El movimiento de datos sin LAN y la eliminación de datos duplicados del lado del cliente se excluyen mutuamente. Si ha habilitado el movimiento de datos sin LAN y la eliminación de datos duplicados del lado del cliente, se completarán las operaciones de movimiento de datos sin LAN y se ignorará la eliminación de datos duplicados del lado del cliente.
- El cifrado y la eliminación de datos duplicados del lado del cliente se excluyen mutuamente. Si ha habilitado el cifrado y la eliminación de datos duplicados del lado del cliente, se completarán las operaciones de cifrado y se ignorará la eliminación de datos duplicados del lado del cliente. Los archivos cifrados y los archivos aptos para la eliminación de datos duplicados del lado del cliente pueden procesarse en la misma operación, pero se llevan a cabo en transacciones distintas.

  Requisitos:

  1. En cualquier transacción, todos los archivos tienen que incluirse o excluirse en la eliminación de datos duplicados. Si la transacción incluye archivos mixtos, no se realizará correctamente y la interfaz de programación de aplicaciones (API) devolverá un código de `DSM_RC_NEEDTO_ENDTXN`.
  2. Utilice el cifrado del dispositivo de almacenamiento junto con la eliminación de duplicados de datos del lado del cliente. Como la capa de sockets seguros (SSL) se utiliza junto a la optimización de almacenamiento del lado del cliente, no es necesario el cifrado del cliente.
- Las siguientes funciones no están disponibles para la eliminación de datos duplicados del lado del cliente:
  – Cliente de IBM Spectrum Protect for Space Management (HSM)

  - Almacenamiento intermedio compartido con la interfaz de programación de aplicaciones (API)
  - NAS
  - Copia de seguridad de subarchivos
- La eliminación de copias del almacenamiento intermedio no puede utilizarse con transformaciones de datos tales como compresión, cifrado y eliminación de duplicados de datos.
- Si utiliza una optimización de almacenamiento del lado de cliente, la API detecta y no puede realizar (con RC=254) las copias de seguridad de las extensiones de archivo marcadas como caducadas en el servidor durante el envío de datos al servidor. Si desea volver a entrar la operación, debe incluir esa programación en la aplicación de llamada.
- Las operaciones de grabación simultánea que tienen lugar en el servidor tienen preferencia sobre la eliminación de datos duplicados del lado del cliente. Si las operaciones de grabación simultánea están habilitadas, no se produce la eliminación de datos duplicados del lado del cliente.

  Restricción: Cuando se habilita la eliminación de datos del lado del cliente, la interfaz de programación de aplicaciones (API) no puede recuperarse de un estado en el que el servidor se ha quedado sin almacenamiento en la agrupación de destino, ni siquiera si se ha definido la siguiente agrupación. Se detiene un código de razón de detención DSM_RS_ABORT_DESTINATION_POOL_CHANGED y la operación falla. Existen dos formas de recuperarse de esta situación:

  1. Solicite al administrador que añada más volúmenes reutilizables a la agrupación de archivos original.
  2. Volver a intentar realizar la operación con la eliminación de datos duplicados deshabilitada.

Para conseguir mayores ahorros en el ancho de banda, puede habilitar una caché local para la eliminación de datos duplicados. La caché local impide que las consultas vayan al servidor de IBM Spectrum Protect. El valor predeterminado de ENABLEDEDUPCACHE es NO, lo que significa que la memoria caché no está fuera de sincronía con el servidor. Si la memoria caché está fuera de sincronía con el servidor, la aplicación vuelve a enviar todos los datos. Si la aplicación puede reintentar una transacción fallida, y desea utilizar la memoria caché local, establezca la opción ENABLEDEDUPCACHE en YES en el archivo dsm.opt (Windows) o dsm.sys (UNIX).

Al final de una restauración, si *todos* los datos se restauraron a través de la interfaz de programación de aplicaciones (API) y el cliente eliminó los datos duplicados del cliente, se calcula un resumen de punto a punto y se compara con el valor calculado en el momento de realizar la copia de seguridad. Si los valores no coincide, se devuelve el error DSM_RC_DIGEST_VALIDATION_ERROR. Si una aplicación recibe este error, los datos están corrompidos. Este error también puede ser resultado de un error pasajero en la red, así que vuelva a intentar llevar a cabo la restauración o recuperación.

Aquí tiene un ejemplo del mandato de sesión de consulta que muestra la información sobre la eliminación de datos duplicados:

```
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:
```

Aquí tiene un ejemplo del mandato de clase de gestión de consulta que muestra la información sobre la eliminación de datos duplicados:

```
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
```

Referencia relacionada:

Opción Deduplication

Excluir archivos de la eliminación de datos duplicados

Puede excluir archivos de copia de seguridad o archivado de la eliminación de datos duplicados.

Para excluir archivos del proceso de eliminación de datos duplicados, siga estos pasos:

1. Configure la opción `exclude.dedup` para los objetos que quiere excluir.

   Por ejemplo, para excluir todos los datos de eliminación de duplicados para sistemas UNIX, establezca:

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

2. Para excluir todos los datos de la eliminación de datos duplicados de los sistemas Windows, defina:

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

Importante: Si se envía un objeto a la agrupación de eliminación de datos duplicados, la eliminación de duplicados se produce en el servidor, incluso si el objeto se excluye de la eliminación de datos duplicados del lado del cliente.

Incluir archivos para la desduplicación de datos

Puede incluir archivos de copia de seguridad o archivado en la eliminación de datos duplicados.

Para refinar la lista de archivos que van a incluirse, puede utilizarse la opción `include.dedup` en combinación con la opción `exclude.dedup`.

De forma predeterminada, todos los objetos aptos, se incluyen en la eliminación de datos duplicados.

Aquí tiene algunos ejemplos de UNIX y Linux:

```
exclude.dedup /FS1/.../*

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

Aquí tiene algunos ejemplos de Windows:

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

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

Eliminación de datos duplicados del lado del servidor

La optimización de almacenamiento de datos del lado del servidor es la optimización de almacenamiento de datos realizada por el servidor.

El administrador de IBM Spectrum Protect puede especificar la ubicación de la eliminación de datos duplicados (cliente o servidor) a utilizar con el parámetro **DEDUP** en el mandato del servidor **REGISTER NODE** o **UPDATE NODE**.

En una agrupación de almacenamiento con eliminación de datos duplicados habilitada (agrupación de archivos), solo se retiene una instancia de la extensión de datos. Otras instancias de la misma extensión de datos se sustituyen por un puntero que señala hacia la instancia retenida.

Para obtener más información sobre la eliminación de datos duplicados del lado del servidor, consulte la documentación del servidor de IBM Spectrum Protect.

Migración tras error de la aplicación

Cuando el servidor de IBM Spectrum Protect no está disponible debido a una interrupción, las aplicaciones que utilizan la API pueden utilizar la migración tras error automáticamente a un servidor secundario para la recuperación de datos.

El servidor de IBM Spectrum Protect al que se conectan el cliente y la API durante los procesos de producción normales se denomina *servidor primario*. Cuando el servidor primario está configurado para la réplica del nodo, el servidor también se conoce como *servidor de réplica de origen*. Los datos del nodo del cliente en el servidor de réplica se pueden replicar en el *servidor de réplica de destino*. Este servidor también se conoce como *servidor secundario* y es el servidor al que pasa el cliente de forma automática cuando el servidor primario falla.

El cliente y la API deben estar configurados para la migración tras error automática del cliente,y deben conectarse al servidor de la versión 7.1 (o posterior) que replica los datos de nodo del cliente. La configuración para la API es la misma que la configuración para el cliente de archivado y copia de seguridad.

Durante las operaciones normales, la información de conexión para el servidor secundario se envía automáticamente desde el servidor primario durante el proceso de inicio de sesión. La información del servidor secundario se guardará automáticamente en el archivo de opciones del cliente.

Cada vez que la aplicación del cliente inicia sesión en el servidor deIBM Spectrum Protect, intenta contactar con el servidor primario. Si el servidor primario no está disponible, la aplicación se migra automáticamente al servidor secundario utilizando la información del servidor secundario del archivo de opciones del cliente. En la modalidad de migración tras error, la aplicación puede consultar al servidor secundario y restaurar o recuperar los datos replicados.

Debe realizar la copia de seguridad de la aplicación al menos una vez en el servidor primario. La API puede migrarse al servidor secundario para recuperar los datos solo si los datos del nodo del cliente se replicaron desde el servidor primario al servidor secundario.

Conceptos relacionados:

Configuración y uso de la migración tras error automática del cliente

Información de estado de migración tras error

La interfaz de programación de aplicaciones proporciona información de estado que puede utilizarse para determinar el estado de la migración tras error y el estado de los datos de réplica del cliente en el servidor secundario.

El estado indica si la réplica de copia de seguridad más reciente se ha replicado en el servidor secundario. Si la indicación de fecha y hora de la operación de copia de seguridad más reciente de la API coincide con la indicación de fecha y hora de la copia de seguridad del servidor secundario, el estado de la réplica es el actual. Si las indicaciones de fecha y hora no coinciden, el estado de réplica no es el actual y es posible que esté desfasado.

La siguiente información de estado de réplica se devuelve en la respuesta de **query filespace** con la llamada de función **dsmGetNextQObj** en la estructura **qryRespFSData**:

Tabla 14. La información sobre el estado de la réplica se indica en la API

Información de estado	Tipo	Definición
Inicio de última réplica	**lastReplStartDate**	La última vez que se inició una réplica.
Fin de la última réplica	**lastReplCmpltDate**	La última vez que se completó una réplica, incluso si se produjo una anomalía.
Fecha de la última copia de seguridad almacenada (Servidor)	**lastBackOpDateFromServer**	La indicación de fecha y hora de la última copia de seguridad que se almacenó en el servidor.
Fecha de la última copia de seguridad (Local)	**lastBackOpDateFromLocal**	La indicación de fecha y hora de la última copia de seguridad que se almacenó el cliente.

El estado de migración tras error se indica mediante el campo **bIsFailOverMode** en la estructura **dsmInitExOut_t**.

Consulte Apéndice B, "Archivos de origen de definiciones de tipo de API", en la página 165 para la estructura y el tipo de definiciones de la API.

El código de retorno de DSM_RC_SIGNON_FAILOVER_MODE indica que el cliente y la API han fallado en el servidor secundario y se están ejecutando en modo de migración tras error.

Ejemplo de inicio de sesión durante una migración tras error

El siguiente ejemplo muestra la salida de un inicio de sesión en el servidor durante una migración tras error:

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

Ejemplo de un mandato de sesión de consulta

El siguiente ejemplo muestra la salida de un mandato **query session** que muestra la información del servidor secundario (réplica):

```
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
```

Ejemplo de un mandato de espacio de archivos de consulta

El siguiente ejemplo es la salida de un mandato **query filespace** que muestra el estado de réplica de un espacio de archivos en el servidor secundario:

```
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
```

Referencia relacionada:

"dsmGetNextQObj" en la página 115

Diagramas de ejemplo de copia de seguridad/archivado

La API está diseñada para flujos lógicos sencillos y transiciones claras entre los distintos estados de la aplicación del cliente. Esta transición de estado capta fallos lógicos y errores de programa muy temprano en el ciclo de desarrollo, mejorando en gran medida la calidad y fiabilidad del sistema.

Por ejemplo, no es posible realizar una llamada **dsmSendObj** a no ser que se inicie una transacción y se haya realizado anteriormente una llamada **dsmBindMC** para el objeto al que se le está realizando una copia de seguridad.

Figura 12 muestra el diagrama de estado para realizar operaciones de copia de seguridad o archivado dentro de una transacción. La flecha que va de "In Send Object" a **dsmEndTxn** indica que una llamada **dsmEndTxn** se puede iniciar después de una llamada a **dsmSendObj** o **dsmSendData**. Es posible que desee realizar esta operación si ocurre una condición de error durante el envío de un objeto y desea detener toda la operación. En este caso, debe utilizar un voto de DSM_VOTE_ABORT. En circunstancias normales, no obstante, llame a **dsmEndSendObj** antes de finalizar la transacción.

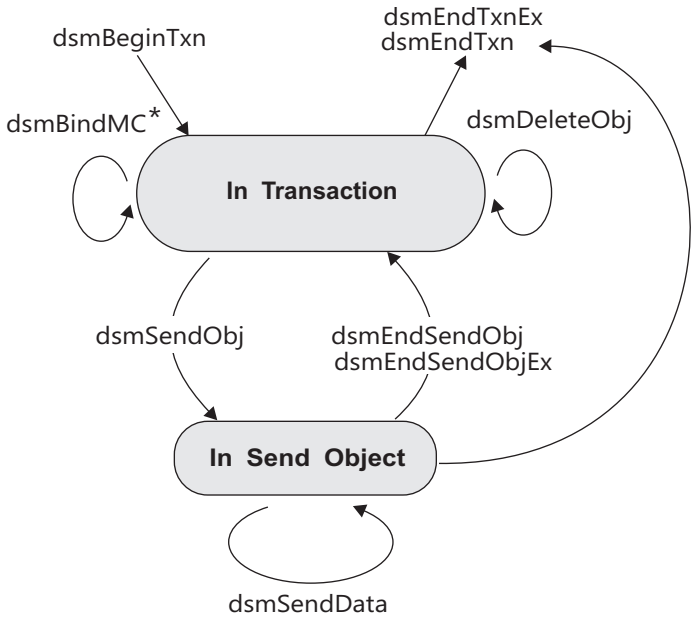


Figura 12. Diagrama de estado para operaciones de copia de seguridad y archivado

Figura 13 en la página 63 muestra el organigrama para realizar operaciones de copia de seguridad y archivado dentro de una transacción.

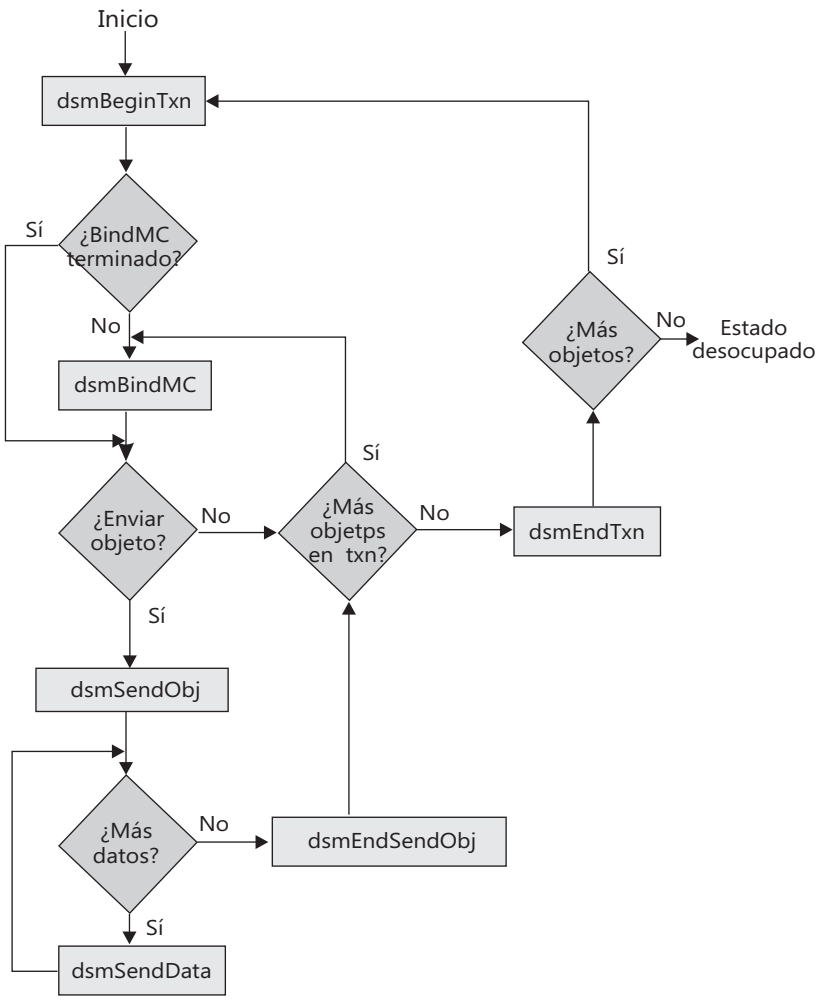


Figura 13. Organigrama de las operaciones de archivado y copia de seguridad

La función principal en estos dos diagramas es el bucle entre las siguientes llamadas de la API desde una transacción:

- **dsmBindMC**
- **dsmSendObj**
- **dsmSendData**
- **dsmEndSendObj**

La llamada **dsmBindMC** es única porque se puede iniciar desde dentro o fuera de los límites de una transacción. También puede iniciarla desde una transacción distinta si fuera necesario. El único requisito de la llamada **dsmBindMC** es que se realice antes de realizar una copia o archivar un objeto. Si el objeto que se está archivando o al que se le está realizando una copia de seguridad no está asociado con una clase de gestión, **dsmSendObj** devuelve un código de error. En este caso, la transacción finaliza con una llamada a **dsmEndTxn** (esta condición de error no aparece en el diagrama).

El diagrama ilustra cómo una aplicación utiliza varias transacciones de objeto. Muestra dónde se pueden colocar los puntos de decisión para determinar si el objeto que se envía encaja dentro de la transacción o si se debe iniciar una

transacción nueva.

Ejemplo de código de funciones de la API que envían datos al almacenamiento de IBM Spectrum Protect storage

Este ejemplo demuestra el uso de las funciones API que envían datos al almacenamiento de IBM Spectrum Protect. La llamada **dsmSendObj** aparece dentro de una sentencia de cambio, con lo que es posible llamar parámetros distintos dependiendo de si se está realizando una operación de copia de seguridad o de archivado.

La llamada **dsmSendData** se realiza desde dentro de un bucle que envía de forma recurrente datos hasta que se configura un indicador que permite la ejecución del programa para salir del bucle. Toda la operación de envío se lleva a cabo desde dentro de la transacción.

El tercer parámetro de la llamada **dsmSendObj** es un almacenamiento intermedio que contiene la descripción del archivado. Debido a que los objetos a los que se les realiza una copia de seguridad no tienen una descripción, este parámetro es NULO cuando se lleva a cabo una copia de seguridad de un objeto.

Figura 8 en la página 31 muestra un ejemplo que ilustra el uso de la llamada de función **dsmBindMC**.

```
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 =
}
```

Figura 14. Un ejemplo de envío de datos a un servidor

Agrupación de archivos

La API de IBM Spectrum Protect tiene un protocolo de agrupación de archivos que relaciona a varios objetos separados. Es posible mencionar y gestionar estos archivos como un grupo lógico en el servidor. Un grupo lógico requiere que todos los miembros del grupos y el líder pertenezcan al mismo nodo y espacio de archivo en el servidor.

Cada grupo lógico cuenta con un líder de grupo. Si el líder de grupo se elimina, se elimina el grupo. No puede suprimir un miembro que es parte de un grupo. La caducidad de todos los miembros en un grupo depende del líder del grupo. Por ejemplo, si un miembro está marcado para su caducidad, el miembro no caduca a menos que caduque el líder del grupo. No obstante, si un miembro no está marcado para su caducidad y el líder del grupo está caducado, entonces todos los miembros caducan.

Los grupos de archivos contienen únicamente datos de seguridad y no pueden contener datos de archivado. Los objetos archivados pueden utilizar el campo **Archive Description** para facilitar un tipo de grupo si la aplicación lo requiere.

La llamada **dsmGroupHandler** agrupa las operaciones. La función **dsmGroupHandler** debe llamarse desde dentro de una transacción. La mayoría de las condiciones de error se detectan en las llamadas **dsmEndTxn**l o **dsmEndTxnEx**.

La estructura out en **dsmEndTxnEx** incluye un campo nuevo, **groupLeaderObjId**. Este campo contiene el ID de objeto del líder del grupo si un grupo fue abierto en esa transacción. Es posible crear un grupo en más de una transacción. Un grupo no se compromete o guarda en el servidor hasta que se realiza un cierre. El **dsmGroupHandler** es una interfaz que acepta cinco operaciones distintas. Incluyen:

- DSM_GROUP_ACTION_OPEN
- DSM_GROUP_ACTION_CLOSE
- DSM_GROUP_ACTION_ADD
- DSM_GROUP_ACTION_ASSIGNTO
- DSM_GROUP_ACTION_REMOVE

Tabla 15 incluye en una lista las acciones de la llamada de función **dsmGroupHandler**:

Tabla 15. funciones dsmGroupHanlder

Acción	Descripción
OPEN	La acción OPEN crea un grupo. El siguiente objeto que se envía se convierte en líder de grupo. El líder de grupo no puede tener contenido. Todos los objetos después del primer objeto se convierten en miembros que se añaden al grupo. Para crear un grupo, abra un grupo y pase una cadena única para identificarlo. Este identificador único permite que se abran varios grupos con el mismo nombre. Después de que el grupo se haya abierto, el siguiente objeto que se envía es el líder del grupo. Todos los demás objetos que se envían son miembros del grupo.

Tabla 15. funciones dsmGroupHanlder (continuación)

Acción	Descripción
CLOSE	La acción CLOSE compromete y guarda un grupo abierto. Para cerrar el grupo, pase el nombre del objeto y la cadena única que se utiliza en la operación de apertura. La aplicación debe comprobar que existen grupos abiertos, y si fuera necesario, cerrarlos o eliminarlos. Un grupo no se compromete o guarda hasta que se realice un cierre del grupo. Una acción CLOSE falla en las siguientes condiciones: • El grupo que está intentando cerrar tiene el mismo nombre que un grupo abierto existente. • Existe incompatibilidad de clase de gestión entre el grupo cerrado actual y el grupo nuevo del mismo nombre que se va a cerrar. En este caso, complete los pasos siguientes: 1. Consulte el grupo cerrado anterior. 2. Si la clase de gestión del grupo cerrado existente es diferente de la clase de gestión asociada con el grupo abierto actual, emita un **dsmUpdateObject** con tipo DSM_BACKUPD_MC. Este mandato actualiza el grupo existente a la nueva clase de gestión. 3. Emita la acción CLOSE.
ADD	La acción ADD adjunta un objeto a un grupo. Todos los objetos que se envían después de la acción ADD se asignan al grupo.
ASSIGNTO	La acción ASSIGNTO permite al cliente asignar objetos que existen en el servidor al grupo igual declarado. Esta transacción configura la relación del grupo igual. La acción ASSIGNTO es parecida a la acción ADD, con las siguientes excepciones: • La acción ADD se aplica a objetos dentro de una transacción en proceso. • La acción ASSIGNTO se aplica a un objeto que está en el servidor.
REMOVE	La acción REMOVE elimina un miembro, o una lista de miembros, de un grupo. No es posible eliminar del grupo al líder del grupo. Un miembro de grupo debe eliminarse antes de que puede borrarse.

Utilice los siguientes tipos de consulta para el soporte de grupo:
- **qtBackupGroups**
- **qtOpenGroups**

qtBackupGroups consulta a grupos que están cerrados mientras que **qtOpenGroups** consulta a grupos que están abiertos. El almacenamiento intermedio de consulta para los nuevos tipos tiene campos para **groupLeaderObjId** y **objType**. La consulta se realiza de forma distinta dependiendo de los valores de estos dos campos. La siguiente tabla incluye algunas posibilidades de consulta:

Tabla 16. Ejemplo de consultas

groupLeaderObjId.hi	groupLeaderObjId.lo	objType	Resultado
0	0	NULO	Devuelve una lista de todos los líderes de grupo
grpLdrObjId.hi	grpLdrObjId.lo	0	Devuelve una lista de todos los miembros de grupo asignados al líder de grupo específico (**grpLdrObjId**).
grpLdrObjId.hi	grpLdrObjId.lo	objType	Devuelve una lista utilizando **BackQryRespEnhanced3**, para cada miembro de grupo que se asigna al líder de grupo específico (**grpLdrObjId**), y coincide con el tipo de objeto (**objType**).

La estructura de la respuesta (**qryRespBackupData**) de **dsmGetNextQObj** incluye dos campos para el soporte de grupo:

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

Estos campos son distintivos booleanos. El siguiente ejemplo muestra la creación del grupo, la adición de miembros al grupo y el cierre del grupo para confirmar el grupo en el servidor de IBM Spectrum Protect.

```
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
```

Figura 15. Ejemplo del pseudo-código que se utiliza para crear un grupo

Para un ejemplo de código, consulte el programa de grupo de muestra `dsmgrp.c` que está incluido en el directorio de la API `sampsrc`.

Recibir datos de un servidor

Los clientes de aplicaciones pueden recibir datos u objetos con nombre y sus datos asociados del almacenamiento de IBM Spectrum Protect utilizando las funciones de restauración y recuperación. La función de restauración accede a los objetos de los que se hizo una copia de seguridad previamente, y la función de recuperación accede a los objetos que han sido archivados.

Restricción: El API sólo puede restaurar o recuperar objetos a los que se les ha hecho una copia de seguridad o que han sido archivados con llamadas API.

Ambas funciones de recuperación y restauración inician una operación de consulta. La consulta devuelve información distinta dependiendo de si los datos fueron archivados o se les realizó una copia de seguridad anteriormente. Por ejemplo, una consulta de objetos a los que se les ha realizado una copia de seguridad devuelve información sobre si los objetos están activos o inactivos, mientras que una consulta sobre objetos archivados devuelve información como descripciones de objeto. Ambas consultas devuelven ID de objeto que se utilizan para identificar de forma exclusiva el objeto en el servidor.

Restauración o recuperación de objeto parcial

El cliente de la aplicación puede recibir únicamente una parte del objeto. Esto se denomina una restauración de objeto parcial o una recuperación de objeto parcial.

Atención: La recuperación o restauración parcial de objetos cifrados o comprimidos produce resultados imprevisibles.

Nota: Si codifica la aplicación para que utilice una recuperación o restauración parcial de un objeto no es posible comprimir los datos mientras los envía. Para implementar esto, configure *ObjAttr.objCompressed* en *bTrue*.

Para realizar una restauración o recuperación de objeto parcial, asocie los siguientes campos de datos con cada entrada **GetList** de objeto:

desplazamiento
El desplazamiento de byte dentro del objeto en donde empezar a devolver datos.

longitud
El número de bytes de objeto para devolver.

Utilice DSM_MAX_PARTIAL_GET_OBJ con el fin de determinar el número máximo de objetos que pueden realizar una recuperación o restauración parcial de una lista **dsmBeginGetData** específica.

Los siguientes campos de datos, utilizados en la llamada **dsmBeginGetData**, determinan qué parte del objeto se restaura o recupera:

- Si tanto el desplazamiento como la longitud son cero, se restaura o recupera todo el objeto del almacenamiento IBM Spectrum Protect.
- Si el desplazamiento es mayor que cero, pero la longitud es cero, el objeto se restaura o recupera del desplazamiento hasta el final.
- Si la longitud es mayor que cero, sólo se recupera o restaura la parte del objeto desde el desplazamiento con la longitud especificada.

Restaurar o recuperar datos

Después de realizar una consulta y de establecer una sesión con el servidor IBM Spectrum Protect, es posible ejecutar un procedimiento para restaurar o recuperar los datos.

Para restaurar o recuperar datos, complete los pasos siguientes:

1. Consulte al servidor IBM Spectrum Protect para archivar o realizar una copia de seguridad de los datos.
2. Determine los objetos que se van a restaurar o recuperar del servidor.
3. Ordene los objetos en el campo **Restore Order**.
4. Envíe la llamada **dsmBeginGetData** con la lista de objetos a los que desea acceder.
5. Envíe la llamada **dsmGetObj** para obtener cada objeto para el sistema. Es posible que se necesiten varias llamadas **dsmGetData** para cada objeto con el fin de obtener todos los datos del objeto asociado. Envíe la llamada **dsmEndGetObj** después de obtener todos los datos de un objeto.
6. Envíe la llamada **dsmEndGetData** después de que todos los datos para todos los objetos se reciban o para e enviar la operación de recepción.

Realizar consultas al servidor

Antes de empezar ninguna operación de restauración o recuperación, consulte primero el servidor IBM Spectrum Protect para determinar qué objetos puede recibir del almacenamiento.

Para enviar la consulta la aplicación debe introducir las estructuras y listas de parámetros adecuadas para la llamada **dsmBeginQuery**. La estructura debe incluir el espacio de archivos que la consulta examina y entradas que se ajustan a un patrón en los campos de nombre de alto y bajo rango. Si la sesión se inició con un nombre de propietario NULO, no es necesario especificar el campo del propietario. No obstante, si la sesión se inició con un nombre de propietario explícito, sólo se devuelven los objetos asociados con ese nombre de propietario.

La consulta **BackupQuery** de punto en el tiempo proporciona una instantánea del sistema en una hora determinada. Al especificar una fecha válida, es posible consultar todos los archivos a los que se les realizó una copia de seguridad a esa hora. Incluso si un objeto tiene una copia de seguridad activa de una fecha posterior, el punto en el tiempo modifica el estado de un objeto con el fin de que se devuelva la copia inactiva anterior. Para obtener más información, consulte el siguiente ejemplo: pitDate.

Una consulta devuelve toda la información que se almacenó con el objeto, además de la información en la tabla siguiente.

Tabla 17. Consultar la información devuelta por el servidor

Campo	Descripción
`copyId`	Los valores `copyIdHi` y `copyIdLo` proporcionan un número de 8 bytes que identifica de manera exclusiva este objeto para el nodo del almacenamiento de IBM Spectrum Protect. Utilice este ID para solicitar un objeto específico del almacenamiento de los procesos de restauración o recuperación.
`restoreOrderExt`	El valor `restoreOrderExt` proporciona un mecanismo para recibir objetos del almacenamiento de IBM Spectrum Protect de la manera más eficaz posible. Ordene los objetos para restaurar este valor y asegurarse de que las cintas sólo se montan una vez y se lee de adelante a atrás.

Debe guardar toda o parte de la información de la consulta para los procesos posteriores. No modifique los campos `copyId` y `restoreOrderExt` ya que se necesitan en la operación de restauración real. También es necesario conservarla información necesaria para abrir un archivo de datos o identificar un destino.

Llame a **dsmEndQuery** para finalizar la operación de consulta.

Seleccionar y ordenar objetos por orden de restauración

Una vez que se ha realizado la consulta de copia de seguridad o archivado, el cliente de la aplicación debe determinar qué objetos, si es que hay alguno, se van a restaurar o recuperar.

Después se ordenan los objetos en orden ascendente (de bajo a alto). Este orden es muy importante para el rendimiento de la operación de restauración. Al ordenar los objetos en los campo **restoreOrderExt** se garantiza que los datos se leen desde el servidor en el orden más eficaz.

Primero se restauran todos los datos del disco, seguido de los datos en clases de medios que requieren montajes de volumen (como una cinta). El campo **restoreOrderExt** también garantiza que los datos de la cinta se lean en orden con el proceso empezando al principio de la cinta y progresando hacia el final.

Un orden adecuado en el campo **restoreOrderExt** significa que no se realizan montajes de cintas duplicados ni rebobinados de cintas innecesarios.

Un valor que no es cero en el campo **restoreOrderExt**.top se correlaciona con un dispositivo de acceso de serie único en el servidor IBM Spectrum Protect. Debido a que un dispositivo de acceso de serie sólo se puede utilizar en un punto de sesión/montaje a la vez, la aplicación debe asegurarse de que si utiliza varias sesiones no existen restauraciones simultaneas con el mismo valor **restoreOrderExt**.top. De lo contrario la primera sesión es capaz de acceder a los objetos, pero las otras sesiones esperan hasta que la primera sesión termina y el dispositivo esté disponible.

El siguiente ejemplo ilustra cómo ordenar objetos utilizando los campos **Restaurar orden**.

Figura 16. Ordenación de objetos con los campos Restaurar orden

```
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);
}
```

Iniciar la llamada dsmBeginGetData

Después de seleccionar y ordenar los objetos que recibir, envíelos a IBM Spectrum Protect para una operación de restauración o de recuperación. La llamada **dsmBeginGetData** inicia una operación de restauración o recuperación. Los objetos se devuelven al cliente de la aplicación en el orden solicitado.

Rellene la información de estos dos parámetros en estas llamadas:

mountWait
: Este parámetro indica al servidor si el cliente de la aplicación espera a que se monte un medio fuera de línea para obtener los datos de un objeto o si dicho objeto debe ignorarse durante el proceso de restauración o recuperación.

dsmGetObjListP
: Este parámetro es una estructura de datos que contiene el campo**objId** el cual es una lista de todos los ID de objeto que se restauran o recuperan. Cada **objId** está asociado con una estructura **partialObjData** que describe si todo el **objId** o únicamente una sección particular del objeto va a recuperarse.

  Cada **objId** tiene ocho bytes de longitud, con lo que una única solicitud de recuperación o restauración puede contener miles de objetos. El número de objetos que se solicitan en una única llamada está limitada a DSM_MAX_GET_OBJ o DSM_MAX_PARTIAL_GET_OBJ.

Recibir todos los objetos para restaurar o recuperar

Una vez que la llamada **dsmBeginGetData** se envía, es posible realizar un procedimiento para recibir cada uno de los objetos que se envían al servidor.

El código de retorno DSM_RC_MORE_DATA significa que se ha devuelto un almacenamiento intermedio y debe llamar a **dsmGetData** de nuevo. Compruebe el **DataBlk.numBytes** para saber el número real de bytes retornados.

Cuando obtiene todos los datos de un objeto, debe enviar una llamada **dsmEndGetObj**. Si se reciben más objetos, envíe **dsmGetObj** de nuevo.

Si desea detener el proceso, por ejemplo, descartar cualquier dato que quede en la cadena de restauración para todos los objetos que no se han recibido todavía envíe la llamada **dsmEndGetData**. Esta llamada deshecha los datos del servidor al cliente. No obstante, es posible que este método tarde en finalizarse. Si desea finalizar una operación de restauración, utilice **dsmTerminate** para cerrar la sesión.

1. Envíe la llamada **dsmGetObj** para identificar el objeto que solicitó de la cadena de datos y para obtener el primer bloque de datos que está asociado con el objeto.
2. Envíe más llamadas **dsmGetData**, tantas como sean necesarias, para obtener los datos de objeto restantes.

Diagramas de ejemplo de restauración y recuperación

Se pueden utilizar un diagrama de estado y un diagrama de flujo para ilustrar cómo realizar las operaciones de recuperación o restauración.

La flecha que va desde "In Get Object" a **dsmEndGetData** indica que puede enviar una llamada **dsmEndGetData** después de una llamada a **dsmGetObj** o **dsmGetData**. Es posible que tenga que hacer esto si ocurre una condición de error mientras obtiene un objeto del almacenamiento IBM Spectrum Protect y desea detener la operación. En circunstancias normales, no obstante, llame primero a **dsmEndGetObj**.

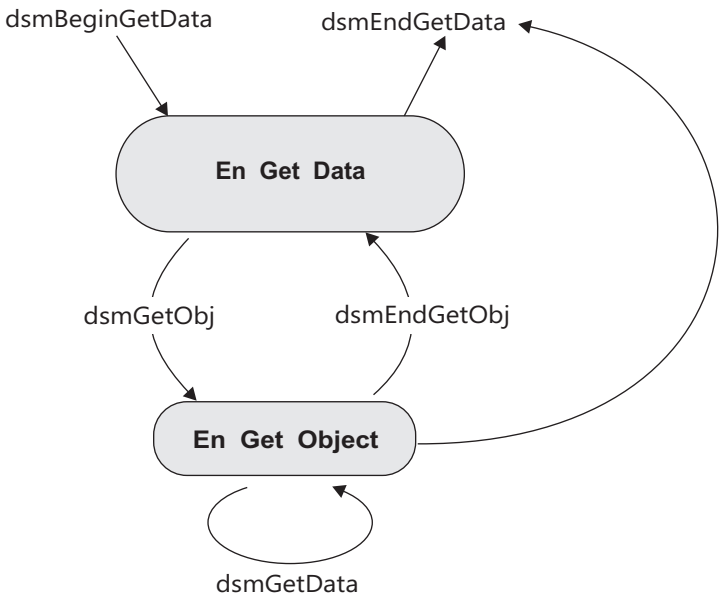


Figura 17. Diagrama de estado para operaciones de restauración y recuperación

Figura 18 en la página 75 muestra el organigrama para llevar a cabo operaciones de recuperación o restauración.

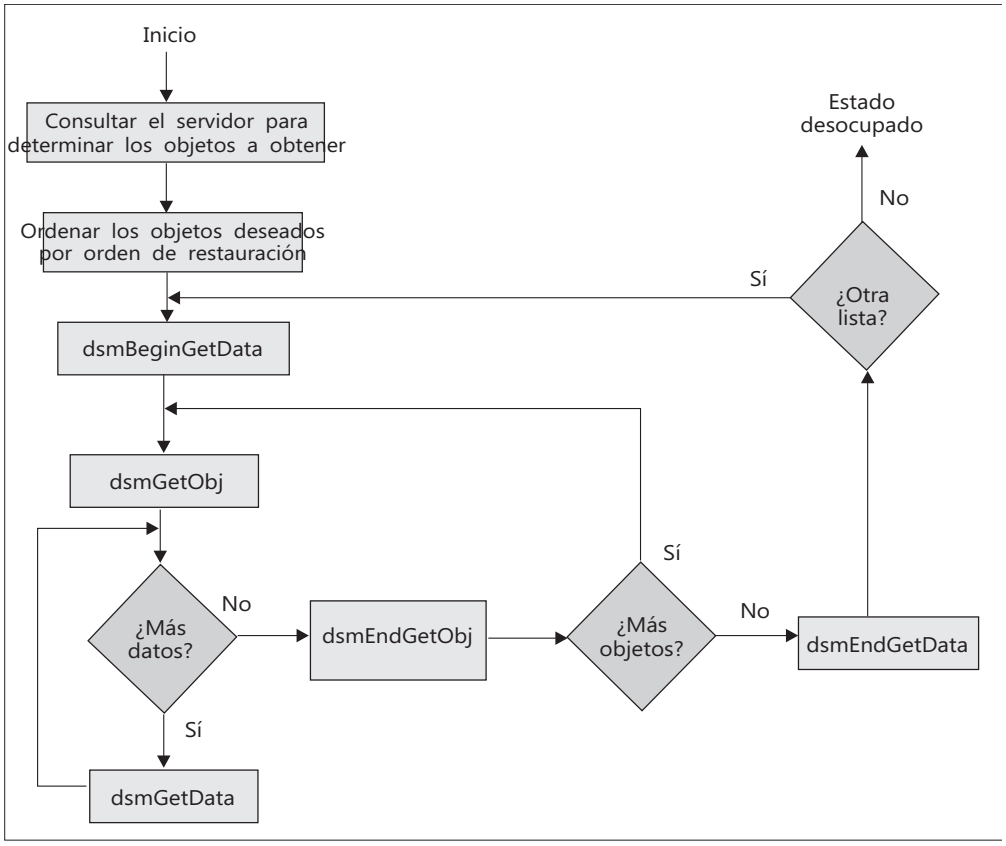


Figura 18. Organigrama para operaciones de restauración y recuperación

Ejemplo de código de recepción de datos de un servidor

Este ejemplo ilustra el uso de las funciones de API para recuperar datos del almacenamiento IBM Spectrum Protect.

La llamada de función **dsmBeginGetData** aparece dentro de una sentencia de cambio, con lo que es posible llamar a distintos parámetros dependiendo de si se trata de una operación de restauración o recuperación. La llamada de función **dsmGetData** se llama de dentro de un bucle que de forma repetida obtiene datos de un servidor hasta que se configura un indicador que permite la ejecución del programa para salir del bucle.

Figura 19. Un ejemplo de recibir datos de un servidor

```
/* 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;
```

Actualizar y eliminar objetos en el servidor

Las aplicaciones API pueden utilizar la llamada de función **dsmUpdateObj** o **dsmUpdateObjEx** para actualizar objetos que fueron archivados o a los que se le realizó una copia de seguridad. Utilice cualquiera de las llamadas en el estado de sesión únicamente, y actualice los objetos de uno en uno. Utilice **dsmUpdateObjEx** para actualizar cualquiera de los objetos que contienen el mismo nombre.

Para seleccionar un objeto archivado, configure la llamada de función **dsmSendType** en **stArchive**.

- Con **dsmUpdateObj**, sólo se actualiza el último objeto archivado con el nombre asignado.
- Con **dsmUpdateObjEx**, es posible actualizar cualquier objeto al especificar el ID del objeto correcto.

En un objeto archivado, la aplicación puede actualizar los siguientes campos:

- Descripción
- Información de objeto
- Propietario

Para seleccionar un objeto al que se le ha realizado una copia de seguridad, configure **dsmSendType** en **stBackup**. En objetos en los que se les ha realizado una copia de seguridad, sólo se actualiza la copia activa.

En un objeto al que se ha realizado una copia de seguridad, la aplicación puede actualizar los siguientes campos:

- Clase de gestión
- Información de objeto
- Propietario

Eliminar objetos del servidor

Las aplicaciones API pueden realizar llamadas para eliminar objetos que fueron archivados o desactivar objetos a los que se les ha realizado una copia de seguridad. La supresión de objetos archivados depende de la autorización de nodo que se haya asignado al registrar el administrador el nodo. Los administradores pueden especificar que los nodos puedan eliminar objetos archivados.

Utilice la llamada de función **dsmDeleteObj** con el fin de eliminar objetos archivados y desactivar objetos a los que se les ha realizado una copia de seguridad. Al utilizar **delType** se elimina la copia de seguridad del objeto del servidor. Esto se basa en **objID**, elimina un objeto de la base de datos del servidor. Sólo un propietario del objeto puede eliminarlo. Es posible eliminar cualquier versión (activa o inactiva) de un objeto. El servidor reconcilia las versiones. Si elimina una versión activa de un objeto, la primera versión inactiva se transforma en activa. Si elimina una versión inactiva de un objeto, todas las versiones antiguas avanzan. El nodo debe estar registrado con el permiso **backDel**.

Un objeto archivado se marca para su eliminación en el almacenamiento cuando el sistema realiza el siguiente ciclo de caducidad del objeto. Una vez eliminado un objeto archivado del servidor, no se podrá recuperar.

Cuando desactiva una copia de seguridad de un objeto en el servidor, el objeto se mueve de un estado activo a un estado inactivo. Estos estados tienen distintas políticas de retención asociadas que se basan en la clase de gestión asignada.

Al igual que la llamada **dsmSendObj**, se envía una llamada **dsmDeleteObj** dentro de los límites de una transacción. El diagrama de estado en Figura 12 en la página 62 muestra cómo una llamada a **dsmDeleteObj** está precedida por una llamada a **dsmBeginTxn** y seguida de una llamada a **dsmEndTxn**.

Registro de eventos

Una aplicación de la API puede registrar mensajes de eventos en ubicaciones centrales. La aplicación puede dirigir el registro al servidor IBM Spectrum Protect , la máquina local o ambos. La llamada a función **dsmLogEventEx** se lleva a cabo dentro de una sesión. Para ver los mensajes registrados en el servidor, utilice el mandato de consulta **actlog** a través del cliente administrativo.

Utilice la opción de cliente de IBM Spectrum Protect , `errorlogretention`, para podar el archivo de registro de errores de cliente si la aplicación graba varios mensajes de cliente al registro de cliente `dsmLogType`, `logLocal` o `logBoth`.

Para obtener más información sobre registros de IBM Spectrum Protect, consulte la documentación del servidor de IBM Spectrum Protect.

Resumen del diagrama de estado para la API de IBM Spectrum Protect

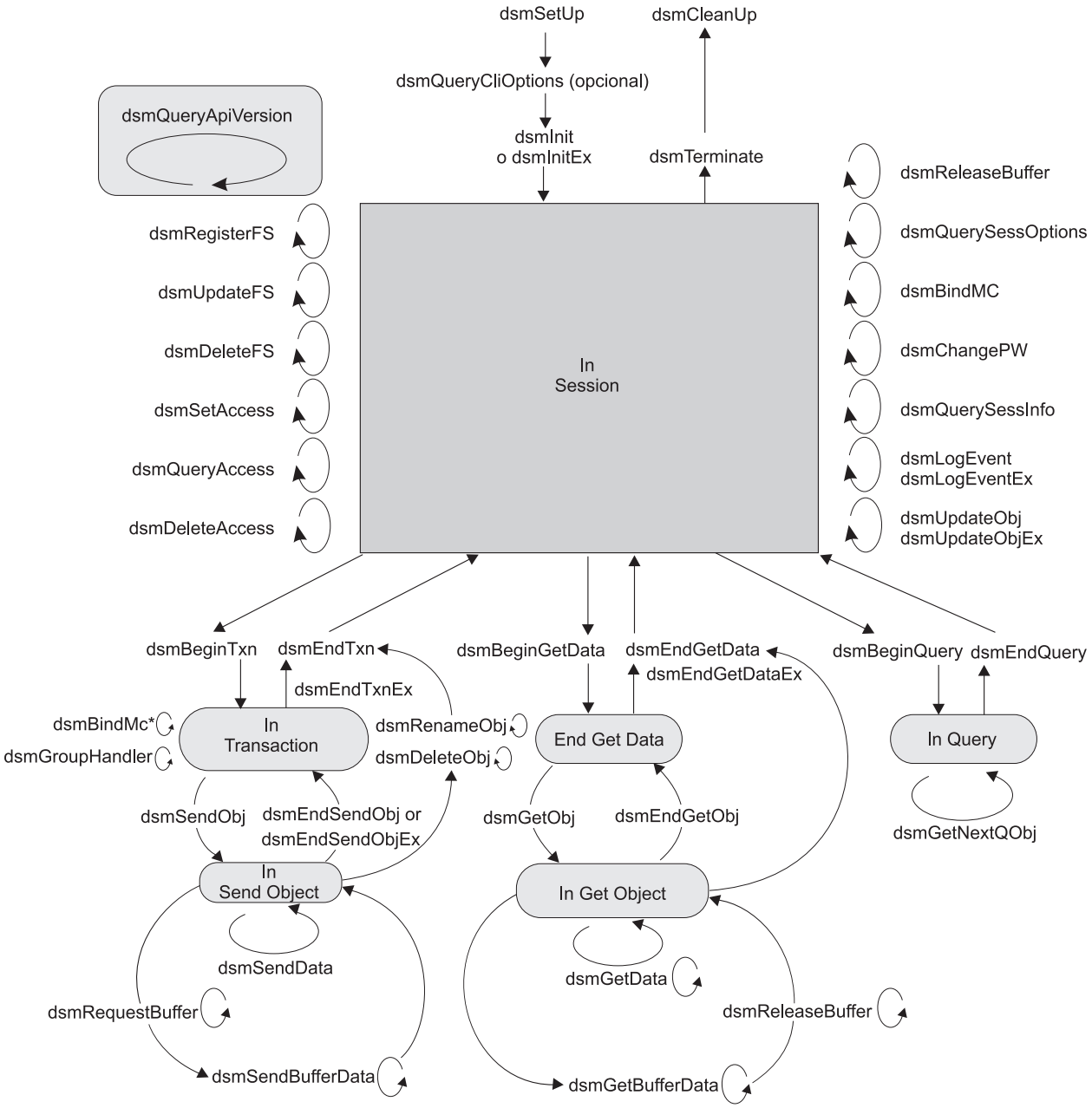
Una vez que haya leído todas las consideraciones para crear su propia aplicación con la API de IBM Spectrum Protect, revise este resumen de diagrama de estado en toda la aplicación.

Figura 20 en la página 79 contiene el diagrama de estado de la API. Contiene todos los diagramas de estado mostrados anteriormente además de varias llamadas que no se han mostrado.

Los puntos en este diagrama incluyen:

- Llamar a **dsmQueryApiVersionEx** en cualquier momento. No tiene un estado asociado. ConsulteFigura 1 en la página 15 para obtener un ejemplo.
- Llamar a **dsmQueryCliOptions** antes de una llamada **dsmInitEx** únicamente.
- Utilizar **dsmRegisterFS**, **dsmUpdateFS**, y **dsmDeleteFS** para gestionar espacios de archivo. Estas llamadas se realizan desde dentro de un estado de sesión inactivo. Utilice la llamada **dsmBeginQuery** con el fin de consultar espacios de archivo. Para obtener más información acerca de las llamadas de espacios de archivo, consulte “Gestionar espacios de archivos” en la página 26.
- Envíe la llamada **dsmBindMC** desde una sesión inactiva o desde un estado de transacción de objeto de envío. Consulte el ejemplo en Figura 8 en la página 31.
- Envíe la llamada **dsmChangePW** desde un estado de sesión inactivo.

  Nota: Si la llamada **dsmInitEx** devuelve un código de retorno de contraseña caducada, la llamada **dsmChangePW** debe realizarse antes de iniciar una sesión válida. Consulte Figura 4 en la página 21 para obtener un ejemplo que utiliza **dsmChangePW**.
- Si una llamada devuelve un error, el estado se queda como estaba. Por ejemplo, si **dsmGetObj** se devuelve con un error, el estado permanece In Get Data y una llamada a **dsmEndGetObj** es un error de secuencia de llamada.



* Puede estar dentro o fuera de una transacción

Figura 20. Resumen del diagrama de estado de la API

Capítulo 4. Conceptos básicos sobre la interoperatividad

API tiene dos tipos de interoperatividad: entre el cliente de copia de seguridad y archivado y las aplicaciones API y entre sistemas operativos diferentes.

Interoperatividad del cliente de archivado/copia de seguridad

La línea de mandato de archivado/copia de seguridad puede acceder a los objetos API para proporcionar una interoperatividad limitada. Los objetos de la API sólo pueden visualizarse y accederse desde un cliente de línea de mandato de archivado/copia de seguridad y no pueden verse o accederse desde ninguna de las interfaces gráficas. El cliente de línea de mandato de archivado/copia de seguridad sólo puede restaurar contenido del archivo y nada más, con lo que sólo debe utilizarse para un tipo de operación de salvamento.

Se proporcionan las siguientes acciones de línea de mandato:
- Eliminar archivado
- Eliminar espacio de archivo
- Consulta
- Restaurar
- Recuperar
- Configurar acceso

La información de ruta contiene los directorios de los objetos de cliente de archivado/copia de seguridad. Por el contrario, es posible que la información de ruta de objeto API no tenga ninguna relación con los directorios existentes: es posible que la ruta esté totalmente inventada. La interoperatividad no cambia este aspecto de los tipos de objeto. Para utilizar esta función correctamente, siga las reglas y restricciones.

Notas:

1. No existe interoperatividad entre el cliente de archivado/copia de seguridad y los objetos API almacenados en un servidor de protección de retención.
2. No es posible utilizar los GUI de cliente de archivado/copia de seguridad para acceder a archivos que se han almacenado utilizando el cliente API. Sólo es posible utilizar la línea de mandato para acceder a estos archivos.

Asignar nombres a los objetos de la API

Establece una convención de nomenclatura para los nombres de objeto de API. La convención de nomenclatura debe ofrecer servicios para el nombre de espacio de archivos, el calificador de alto nivel y el calificador de bajo nivel. El nombre de espacio de archivos y los calificadores de alto nivel pueden hacer referencia a nombres de directorio reales. Cada nombre de objeto puede consistir en más de un nombre de directorio que se aplica al calificador de bajo rango.

Por comodidad, utilice el nombre del objeto que no tiene el prefijo con la información de directorio como calificador de nivel bajo. Para obtener más información, consulte el apartado “ID y nombres de objeto” en la página 23.

Los nombres del espacio de archivos debe estar totalmente calificados cuando se hace referencia a ellos desde la API o la línea de mandatos de la copia de

seguridad/archivado. Por ejemplo, en un sistema operativo UNIX o Linux , registra los siguientes espacios de archivo:

- `/a`
- `/a/b`

Cuando hace referencia a `/a`, se muestran los objetos que están relacionados sólo al espacio de archivo `/a`. Para ver objetos que están relacionados con `/a/b`, debe especificar `/a/b` como nombre del espacio de archivos.

Después de registrar ambos espacios de archivo, si hace una copia de seguridad del objeto `b` en el espacio de archivos `/a`, la consulta para `/a/b` continúa mostrando objetos que están relacionados únicamente a espacio de archivos `/a/b`.

La excepción a esta restricción ocurre en las referencias de espacio de archivo cuando se intenta consultar o eliminar espacios de archivo con la API. En ambos casos, los nombres de espacio de archivos no tiene que estar completos si utiliza un carácter comodín. Por ejemplo, `/a*` hace referencia a `/a` y `/a/b`.

Consejo: Si le importa la interoperatividad, evite los nombres de espacio de archivos que se solapen.

En los sistemas Windows, introduzca los nombres de espacio de archivos entre corchetes { } para los objetos de la API cuando acceda a los objetos desde la interfaz de línea de mandatos de archivado y copia de seguridad. Los sistemas operativos Windows colocan automáticamente nombres de espacio de archivos en letras en mayúsculas cuando registra o hace referencia a los nombres. Sin embargo, esta función automática no ocurre para el resto de la especificación de nombre de objeto. Si desea una interoperatividad total, introduzca el calificador de alto rango y el de bajo rango en mayúsculas en la aplicación cuando se realizan copias de seguridad de los objetos API. Si la aplicación no tiene calificadores de alto nivel en mayúsculas (nombres de directorio) ni calificadores de bajo nivel (nombres de archivo) antes de enviar los objetos al servidor, no podrá acceder a los objetos directamente por nombre a través del cliente de archivado y copia de seguridad.

Por ejemplo, si hay un objeto almacenado en el servidor como`{"FileSpacename"}\TEST\MYDIRNAME\file.txt`, no puede restaurarlo directamente o consultar al objeto `file.txt` porque la aplicación no puso en mayúsculas el nombre del archivo antes de que se copiase al servidor. La única forma de manipular estos objetos es utilizar los caracteres comodines. Por ejemplo, para consultar `\TEST\MYDIRNAME\file.txt`, un cliente de archivado y copia de seguridad debe utilizar los caracteres comodín para todas las partes del nombre del objeto que no se pusieron en mayúscula antes de enviarse al servidor. El siguiente mandato debe utilizarse para consultar este archivo `file.txt`:

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

Si cualquiera de los otros calificadores también se han guardado en minúsculas, esos calificadores también deben consultarse utilizando comodines. Por ejemplo, para consultar un objeto que se guarda como`{"FileSpaceName"}\TEST\mydirname\file.txt`, utilice el siguiente mandato:

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

El siguiente ejemplo demuestra estos conceptos. En Windows y UNIX o en entornos de Linux , no tiene que especificar los calificadores de alto nivel o bajo nivel completos. Sin embargo, si no especifica el calificador completo, debe utilizar el carácter comodín.

Plataforma	Ejemplo
Windows	Para consultar todos los archivos a los que se ha realizado copia de seguridad en el espacio de archivos `MYFS`, entre la siguiente serie: `dsmc q ba "{MYFS}\*\*"` Debe utilizar al menos un asterisco (*) para cada uno de los calificadores de alto y bajo rango.
UNIX o Linux	Para consultar todos los archivos a los que se ha realizado copia de seguridad en el espacio de archivos `/A`, entre la siguiente serie: `dsmc q ba "/A/*/*"` Debe utilizar al menos un asterisco (*) para cada uno de los calificadores de alto y bajo rango.

Mandatos del cliente de copia de seguridad/archivado que se pueden utilizar con la API

Es posible utilizar un subconjunto de mandatos de cliente de copia de seguridad/archivado dentro de una aplicación. Por ejemplo, puede ver y gestionar objetos que son propiedad de otros usuarios ya sea en el mismo nodo o en nodos distintos.

Para ver y gestionar objetos propiedad de otros usuarios ya sea en el mismo nodo o en nodos distintos, lleve a cabo estos pasos:

1. De acceso con el mandato **set access**.
2. Especifique el propietario y el nodo. Utilice las opciones *fromowner* y *fromnode* de la línea de mandato de archivado/copia de seguridad con el fin de especificar el propietario y el nodo. Por ejemplo:

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

Tabla 18 describe los mandatos que se pueden utilizar con los objetos API.

Tabla 18. Mandatos de cliente de archivado/copia de seguridad que puede utilizar con los objetos API

Mandato	Descripción
Delete Archive	Los archivos archivados propiedad del usuario actual se pueden eliminar. La configuración del mandato set access no afecta a este mandato.
Delete Filespace	El mandato **delete filespace** afecta a los objetos de la API.
Consulta	En la línea de mandato de archivado y copia de seguridad es posible consultar los objetos API archivados o que se les ha realizado una copia de seguridad y objetos propiedad de otros usuarios, o que residen en otros nodos. Consulte “Asignar nombres a los objetos de la API” en la página 81 para obtener más información sobre consultar objetos API. Utilizar la opción existente *–fromowner* con el fin de consultar objetos propiedad de un usuario distinto para los que se han dado los permisos de acceso. Utilizar la opción existente *–fromnode* con el fin de consultar objetos que existen en otro nodo par a el que se definen los permisos de acceso que se han concedido. Para obtener más información, consulte “**dsmInitEx**” en la página 124.

Tabla 18. Mandatos de cliente de archivado/copia de seguridad que puede utilizar con los objetos API (continuación)

Mandato	Descripción
Restaurar **Recuperar**	**Nota:** Utilice estos mandatos solo para las situaciones de excepción. Los objetos API cifrados con la clave gestionada por la aplicación se pueden recuperar o restaurar si la clave de cifrado se conoce o guarda en el archivo de la contraseña o en el registro. Los objetos de API se cifran mediante el cifrado transparente que no se puede restaurar o recuperar mediante el cliente de archivado y copia de seguridad. Estos mandatos devuelven los datos como archivos de bit que se crean utilizando los atributos de archivo predeterminados. Es posible restaurar o recuperar objetos API propiedad de otros usuarios o que son de un nodo diferente. El mandato de acceso determina qué objetos son aptos.
Set Access	El mandato **set access** permite a los usuarios gestionar objetos API propiedad de otro usuario, o que residen en otro nodo.

Interoperatividad del sistema operativo

La API de IBM Spectrum Protect es compatible con la interoperatividad entre plataformas. Las aplicaciones en un sistema UNIX o Linux pueden funcionar en espacios de archivo de los que se realizan copias de seguridad desde un sistema de Windows. Del mismo modo, un sistema Windows puede operar en los espacios de archivos de copia de seguridad de un sistema UNIX o Linux.

De forma predeterminada, los objetos de un sistema UNIX son compatibles con los nombres de los objetos de otros sistemas UNIX. De forma predeterminada, los objetos de los sistemas Windows son compatibles con los nombres de los objetos de los sistemas UNIX. Varios parámetros controlan la asignación de nombres de los objetos en los espacios de archivo de IBM Spectrum Protect. Si configura una aplicación de forma adecuada, las aplicaciones que se ejecutan en sistemas tanto Windows como UNIX pueden utilizar los nombres de objetos. Utilice los mismos parámetros para la copia de seguridad y la restauración de objetos.

Restricción: Una aplicación Windows que utiliza Unicode crea un espacio de archivo que no es compatible con las aplicaciones que se ejecutan en sistemas UNIX.

Para conseguir la interoperatividad, siga estas tareas de configuración:

1. Establezca una nomenclatura coherente. Seleccione un carácter para el delimitador `dir`, tal como una barra inclinada (`/`) o una barra invertida (`\`). Coloque este carácter delimitador de directorios delante del nombre del espacio de archivos, el calificador de alto nivel y el calificador de bajo nivel.
2. Cuando llame a **dsmInitEx**, defina el valor del campo **dirDelimiter** en el carácter delimitador de directorio que ha seleccionado y establezca **bCrossPlatform** en **bTrue**.
3. Configure el indicador **useUnicode** en **bFalse** cuando utiliza la interfaz IBM Spectrum Protect. Los nombres de archivos Unicode no son compatibles con los nombres de archivo que no son de Unicode.

Soporte de copia de seguridad de varios nodos con el proxy de nodo de cliente

Las copias de seguridad de varios nodos que comparten almacenamiento se pueden consolidar en un nombre del nodo de destino común del servidor IBM Spectrum Protect. Este método es útil cuando el sistema que ejecuta la copia de seguridad puede cambiar en el tiempo, como con un clúster. Puede utilizar la opción `asnodename` para restaurar datos desde un sistema distinto al otro que ejecutó la copia de seguridad.

Utilice la opción `asnodename` en la serie de opciones **dsmInitEx** para hacer copia de seguridad, archivar, restaurar y restablecer consultar o suprimir datos bajo el nombre de nodo de destino en el servidor IBM Spectrum Protect. También puede especificar la opción `asnodename` en el archivo `dsm.opt` o `dsm.sys`.

Restricción: No utilice nodos de destino como nodos tradicionales, especialmente si cifra los archivos antes de realizar una copia de seguridad de ellos en el servidor.

Para habilitar esta opción, complete los pasos siguientes:

1. Instale el cliente de la API en todos los nodos de un entorno de datos compartido.
2. Si no está ya registrado, registre cada uno de los nodos con el servidor de IBM Spectrum Protect. Registre los nombres de nodo de "destino" comunes que van a compartir cada uno de los nodos agente utilizados en el entorno de datos compartido.
3. Registre cada uno de los nodos de agente del entorno de datos compartidos con el servidor. El nombre de nodo de agente se utiliza para la autenticación. Los datos no se almacenan utilizando el nombre de nodo del agente cuando se utiliza la opción `asnodename`.
4. Solicite al administrador que otorgue autoridad de proxy a todos los nodos del entorno compartido para acceder al nombre de nodo de destino en el servidor IBM Spectrum Protect, utilizando **grant proxynode** .
5. Utilice el mandato de cliente administrativo **query proxynode** para mostrar los nodos de cliente que tienen autoridad para realizar operaciones de cliente en nombre de otro nodo. Esta autorización la otorga el mandato **grant proxynode**. O bien utilice el mandato **dsmQuery** con el tipo de consulta **qtProxyNodeAuth** con el fin de ver los nodos para los que este nodo puede actuar como proxy.
6. Si la aplicación está utilizando el cifrado de usuario de datos, no `TSMENCRKEY`, asegúrese de que todos los nodos están utilizando la misma clave de cifrado. Debe utilizar la misma clave de cifrado en todos los archivos a los que se les ha realizado una copia de seguridad en el entorno del nodo compartido.

Tareas relacionadas:

Soporte de copia de seguridad de datos con el proxy del nodo de cliente (sistemas UNIX y Linux)

Soporte de copia de seguridad de datos con el proxy del nodo de cliente (sistemas Windows)

Capítulo 5. Utilización de la API con Unicode

La API IBM Spectrum Protect soporta Unicode UCS2, una página de códigos de doble byte y de longitud fija cuyo código apunta a todas las páginas conocidas como Japonés, Chino o Alemán. Soporta hasta 65535 códigos únicos.

Restricción: Esta función sólo está disponible en Windows.

Con Unicode, la aplicación puede realizar copias de seguridad y recuperar nombres de archivos en cualquier conjunto de caracteres de la misma máquina. Por ejemplo, en una máquina inglesa, es posible realizar una copia de seguridad y restaurar los nombres de archivo en una página de códigos de cualquier otro idioma.

Cuándo se utiliza Unicode

Es posible simplificar una aplicación que soporta varios idiomas al grabar una aplicación Unicode y sacar partido de la interfaz Unicode de IBM Spectrum Protect.

Utilice la interfaz Unicode de IBM Spectrum Protect si alguna de las siguientes condiciones es verdadera:

- Si la aplicación ya está compilada para Unicode y estaba convirtiendo a un conjunto de caracteres multibyte (mbcs) antes de llamar al API IBM Spectrum Protect.
- Si está grabando una aplicación nueva y desea capacitarlas para soportar Unicode.
- Si la aplicación utiliza una cadena que le llega de un sistema operativa o de otra aplicación que utiliza Unicode.

Si no necesita Unicode, no es necesario compilar la aplicación de nuevo.

La API continúa soportando la interfaz dsm. El SDK de la API contiene los programas de ejemplo callmtu1.c y callmtu2.c que muestran cómo utilizar la API de Unicode. Utilice **makemtu** para compilar estos programas.

Configuración de Unicode

Para configurar y utilizar Unicode debe realizar un procedimiento particular para que la API registre un espacio de archivo Unicode en el servidor y que todos los nombres de archivo en dicho espacio se conviertan en cadenas Unicode.

Restricción: No es posible almacenar nombres de archivo Unicode y no Unicode en el mismo espacio de archivo.

1. Compilar el código con el indicador `-DUNICODE`.
2. Todas las cadenas de la aplicación deben ser cadenas de caracteres comodín.
3. Siga las estructuras del archivo tsmapitd.h, y las definiciones de función del archivo tsmapifp.h en las llamadas al API.
4. Configure el indicador *useUnicode* en *bTrue* en la llamada de función **tsmInitEx**. Cualquier espacio de archivo nuevo se registra como un espacio de archivo Unicode.

Cuando envía datos a espacios de archivo no Unicode registrados previamente, la API continúa enviando nombres de archivo como no Unicode. Renombre los espacios de archivo antiguos en el servidor como fsname_old e inicie un espacio de archivo nuevo con datos nuevos. La API restaura los datos que no son Unicode de los espacios de archivo antiguos. Utilice el campo **bIsUnicode** en la estructura **tsmQryRespFSData** que se devuelve en un espacio de archivo de una consulta para determinar si un espacio de archivo es o no Unicode.

Cada llamada de función **dsmXXX** tiene una llamada de función **tsmXXX** correspondiente. La diferencia entre las dos son las estructuras que utilizan. Todas las estructuras de llamada a función **tsmXXX** tienen tipos dsChar_t types para los valores de cadena cuando se compilan con el distintivo UNICODE. dsChar_r se correlaciona con un carácter comodín. No hay otras diferencias entre estas interfaces.

Restricción: Utilice una interfaz o la otra. No mezcle las interfaces de llamada a función **dsmXXX** y de llamada a función **tsmXXX**. Asegúrese de que utiliza las estructuras de IBM Spectrum Protect y las definiciones de la versión IBM Spectrum Protect.

Algunas constantes continúan definiéndose en el archivo dsmapitd.h, con lo que necesita tanto el archivo dsmapitd.h como el archivo tsmapitd.h en la compilación.

Puede utilizar la interfaz de IBM Spectrum Protect en otros sistemas operativos, como por ejemplo UNIX o Linux, pero en estos sistemas operativos, el tipo dsChar_t se correlaciona con char porque Unicode solo recibe soporte con sistemas operativos Windows. Sólo puede grabar una variación de la aplicación y compilar en más de un sistema operativo utilizando la interfaz de IBM Spectrum Protect. Si está escribiendo una aplicación nueva, utilice la interfaz de IBM Spectrum Protect.

Si está actualizando una aplicación existente:
1. Convierta las estructuras de llamada a función **dsmXXX** y las llamadas a la interfaz de IBM Spectrum Protect.
2. Migre los espacios de archivos existentes.
3. Realice copias de seguridad de los espacios de archivo nuevo con el indicador *useUnicode* configurado en *true*.

Nota: Después de utilizar un cliente capacitado para Unicode para acceder a un nodo, no es posible conectarse al mismo nodo con una versión antigua de la API o con una API de otro sistema operativo. Si su aplicación utiliza funciones de plataforma cruzada, no utilice el indicador `Unicode`. No hay compatibilidad de plataforma cruzada entre sistemas operativos Unicode y no Unicode.

Cuando habilita el indicador *useUnicode*, todas las estructuras de cadena se consideran cadenas Unicode. En el servidor, sólo los siguientes campos son verdaderamente Unicode:
- Nombre de espacio de archivo
- Alto rango
- Bajo rango
- Descripción de la copia archivada

Todos los campos restantes convierten a mbcs en la página de código local antes de que se envíen al servidor. Los campos, como el nombre de nodo, son cadenas de caracteres comodín. Deben ser válidos en el entorno regional actual. Por ejemplo, en una máquina japonesa, es posible hacer copias de seguridad de archivos con nombres chinos, pero el nombre del nodo debe ser una cadena válida

en japonés. El archivo de opción permanece en la página de código actual. Si necesita crear una lista de inclusión/exclusión Unicode , utilice la opción *inclexcl* con un nombre de archivo y cree un archivo Unicode con patrones Unicode.

Referencia relacionada:

Opción inclexcl

Capítulo 6. Llamadas a función de la API

Tabla 19 proporciona una lista por orden alfabético de las llamadas de función de la API, una descripción breve y la ubicación de la información más detallada acerca de la llamada de función, la cual incluye:

Elemento	Descripción
Objetivo	Describe la llamada de función.
Sintaxis	Contiene el código C real de la llamada de función. Este código se copia de la versión del archivo de encabezado dsmapifp.h UNIX o Linux. Consulte Apéndice C, "Archivo de origen de las definiciones de la función de la API", en la página 207. Este archivo cambia ligeramente en otros sistemas operativos. Los programadores de aplicación de otros sistemas operativos deben comprobar su versión del archivo de encabezado, dsmapifp.h, para obtener la sintaxis exacta de las definiciones de la API.
Parámetros	Describe cada uno de los parámetros en la llamada de función, identificándola como una entrada (I) o salida (O), dependiendo de cómo se utilice. Algunos parámetros están diseñados para funcionar tanto de entrada como de salida (I/O). Los tipos de datos que se mencionan en esta sección se definen en el archivo de encabezado dsmapitd.h. Consulte el apartado Apéndice B, "Archivos de origen de definiciones de tipo de API", en la página 165.
Códigos de retorno	Contiene una lista de los códigos de retorno que son específicos a la llamada de función. Los errores del sistema generales, como los errores de comunicación, problemas del servidor o errores del usuario que puedan aparecen en una llamada no se incluyen en la lista. Los códigos de retorno se definen en el archivo de encabezado dsmrc.h. Consulte Apéndice A, "Archivo de origen de códigos de retorno de la API: `dsmrc.h`", en la página 155.

Tabla 19. Llamadas a función de la API

Llamada a función y location	Descripción
"dsmBeginGetData" en la página 94	Inicia una operación de restauración o recuperación en una lista de objetos del almacenamiento.
"dsmBeginQuery" en la página 95	Inicia una solicitud de consulta para IBM Spectrum Protect para obtener información.
"dsmBeginTxn" en la página 100	Inicia una o más transacciones que empiezan una acción completa. O bien todas las acciones tienen éxito o ninguna.
"dsmBindMC" en la página 101	Asocia o vincula una clase de gestión al objeto que pasa.
"dsmChangePW" en la página 102	Cambia una contraseña de IBM Spectrum Protect.
"dsmCleanUp" en la página 103	Esta llamada se utiliza si se ha llamado a **`dsmSetUp`**.
"dsmDeleteAccess" en la página 104	Elimina las reglas de autorización de las copias de seguridad o archivadas de los objetos.
"dsmDeleteFS" en la página 104	Elimina un espacio de archivo del almacenamiento.
"dsmDeleteObj" en la página 105	Desactiva copias de seguridad, o elimina copias archivadas del almacenamiento.

Tabla 19. Llamadas a función de la API (continuación)

Llamada a función y location	Descripción
"dsmEndGetData" en la página 107	Finaliza una sesión **dsmBeginGetData** que obtiene objetos del almacenamiento.
"dsmEndGetDataEx" en la página 107	Proporciona el total de bytes sin LAN que fueron enviados.
"dsmEndGetObj" en la página 108	Finaliza una sesión**dsmGetObj** que obtiene datos de un objeto especificado.
"dsmEndQuery" en la página 108	Significa el final de una acción **dsmBeginQuery**.
"dsmEndSendObj" en la página 109	Indica el final de los datos que se envían al almacenamiento.
"dsmEndSendObjEx" en la página 109	Proporciona información de compresión y el número de bytes que se enviaron.
"dsmEndTxn" en la página 110	Finaliza una transacción de IBM Spectrum Protect.
"dsmEndTxnEx" en la página 111	Proporciona información acerca del ID del objeto líder que se utiliza en la llamada **dsmGroupHandlerfunction**.
"dsmGetData" en la página 113	Obtiene una cadena de byte de datos de IBM Spectrum Protect y la coloca en el almacenamiento intermedio del llamador.
"dsmGetBufferData" en la página 114	Obtiene un almacenamiento intermedio asignado por IBM Spectrum Protect de datos del servidor IBM Spectrum Protect.
"dsmGetNextQObj" en la página 115	Obtiene la siguiente respuesta de una llamada previa **dsmBeginQuery** y la coloca en el almacenamiento intermedio del llamador.
"dsmGetObj" en la página 118	Obtiene los datos del objeto solicitados de la corriente de datos y los coloca en el almacenamiento intermedio del llamador.
"dsmGroupHandler" en la página 119	Realiza una acción en un grupo de archivos lógico dependiendo de la entrada dada.
"dsmInit" en la página 120	Inicia una sesión de la API y conecta al cliente con el almacenamiento.
"**dsmInitEx**" en la página 124	Inicia una sesión de la API utilizando parámetros adicionales que permiten la verificación extendida.
"dsmLogEvent" en la página 128	Registra un mensaje de usuario al archivo de registros del servidor, al registro de error local o a ambos.
"dsmLogEventEx" en la página 129	Registra un mensaje de usuario al archivo de registros del servidor, al registro de error local o a ambos.
"dsmQueryAccess" en la página 130	Realiza una consulta al servidor acerca de todas las reglas de autorización de acceso en versiones de copias archivadas o de copias de seguridad de objetos.
"dsmQueryApiVersion" en la página 131	Realiza una solicitud de consulta acerca de la versión de la biblioteca de la API a la que accede el cliente de la aplicación.
"dsmQueryApiVersionEx" en la página 131	Realiza una solicitud de consulta acerca de la versión de la biblioteca de la API a la que accede el cliente de la aplicación.
"dsmQueryCliOptions" en la página 132	Consulta valores de opción importantes en los archivos de opción del usuario.

Tabla 19. Llamadas a función de la API (continuación)

Llamada a función y location	Descripción
"dsmQuerySessInfo" en la página 133	Inicia una solicitud de consulta a IBM Spectrum Protect sobre información relacionada con la operación de la sesión especificada en **dsmHandle**.
"dsmQuerySessOptions" en la página 134	Consulta valores de opción importantes que son válidos en la sesión especificada en **dsmHandle**.
"dsmRCMsg" en la página 135	Obtiene el texto del mensaje que está asociado con el código de retorno de la API.
"dsmRegisterFS" en la página 136	Registra un espacio de archivo nuevo con el servidor.
"dsmReleaseBuffer" en la página 137	Devuelve un almacenamiento intermedio asignado por IBM Spectrum Protect.
"dsmRenameObj" en la página 137	Renombra el alto y bajo rango del objeto.
"dsmRequestBuffer" en la página 139	Obtiene un almacenamiento intermedio asignado por IBM Spectrum Protect para la eliminación de una copia de almacenamiento intermedio.
"dsmRetentionEvent" en la página 140	Envía una lista de ID de objetos al servidor, con una operación de evento de retención para que se realice en estos objetos.
"dsmSendBufferData" en la página 141	Envía datos de un almacenamiento intermedio asignado por IBM Spectrum Protect.
"dsmSendData" en la página 142	Envía una corriente de bytes de datos a IBM Spectrum Protect a través de un almacenamiento intermedio.
"dsmSendObj" en la página 143	Inicia una solicitud para enviar un objeto único al almacenamiento.
"dsmSetAccess" en la página 147	Da a otros usuarios o nodos acceso a versiones de copia de seguridad o copias archivadas de los objetos, acceso a todos los objetos o acceso a un grupo determinado.
"dsmSetUp" en la página 148	Sobrescribe los valores de variable de entorno.
"dsmTerminate" en la página 150	Finaliza una sesión con el servidor y limpia el entorno de IBM Spectrum Protect.
"dsmUpdateFS" en la página 150	Actualiza un espacio de archivo en el almacenamiento.
"dsmUpdateObj" en la página 151	Actualiza la información objInfo que está asociada con una copia de seguridad activa de un objeto que ya está en el servidor, o actualiza objetos archivados.
"dsmUpdateObjEx" en la página 152	Actualiza la información objInfo que está asociada con un objeto archivado específico incluso cuando existen varios objetos con el mismo nombre, o actualiza copias de seguridad activas.

Referencia relacionada:

Códigos de retorno de la API

dsmBeginGetData

La llamada de función **dsmBeginGetData** inicia una operación de restauración o recuperación en la lista de objetos del almacenamiento. La lista de objetos se encuentra dentro de la estructura **dsmGetList** . La aplicación crea esta lista con los valores de la consulta que precede a una llamada a **dsmBeginGetData**.

El llamador debe en primer lugar utilizar los campos del orden de restauración que se obtienen de la consulta del objeto con el fin de ordenar la lista que se incluye en esta llamada. Esto garantiza que los objetos se restauran del almacenamiento de la forma más eficaz posible sin tener que rebobinar o remontar las cintas de datos.

Al obtener objetos enteros, el *dsmGetList.numObjID* máximo es DSM_MAX_GET_OBJ. Al obtener objetos parciales, el máximo es DSM_MAX_PARTIAL_GET_OBJ.

Siga la llamada a **dsmBeginGetData** con una o más llamadas a **dsmGetObj** con el fin de obtener todos los objetos de la lista. Después de obtener todos los objetos, o que no se necesiten datos adicionales, se envía la llamada **dsmEndGetObj**.

Cuando se obtienen todos los objetos, o se cancela **dsmEndGetObj** se envía la llamada **dsmEndGetData**. A continuación puede iniciar el ciclo de nuevo.

Sintaxis

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

Parámetros

dsUint32_t dsmHandle (I)
El manejador que asocia esta llamada con una llamada **dsmInitEx**.

dsBool_t mountWait (I)
Un valor booleano verdadero o falso indica si el cliente de la aplicación espera que se monten medios desconectados si los datos que se necesitan están actualmente desconectados. Si mountWait es verdadero, la aplicación espera a que el servidor monte los medios necesarios. La aplicación espera hasta que el soporte esté montado o la solicitud cancelada.

dsmGetType getType (I)
Un tipo enumerado que consiste en **gtBackup** y **gtArchive** indica qué tipo de objeto se va a obtener.

dsmGetList *dsmGetObjListP (I)
Esta estructura contiene información acerca de los objetos o bien de los objetos parciales que se van a restaurar o recuperar. La estructura apunta a una lista de ID de objetos y, en el caso de una recuperación o restauración parcial, a una lista de desplazamientos y longitudes asociadas. Si la aplicación utiliza la función de recuperación o restauración parcial de objetos, configure el campo **dsmGetList.stVersion** en **dsmGetListPORVersion**. En una restauración o recuperación de objeto parcial, no es posible comprimir datos mientras los envía. Para implementar esto, configure **ObjAttr.objCompressed** en *bTrue*.

Consulte Figura 19 en la página 75 y Apéndice B, "Archivos de origen de definiciones de tipo de API", en la página 165 para obtener más información acerca de esta estructura.

Consulte “Restauración o recuperación de objeto parcial” en la página 69 para obtener más información acerca de restauraciones o recuperaciones de objeto parciales.

Códigos de retorno

Los números de código de retorno se proporcionan entre paréntesis ( ).

Tabla 20. Códigos de retorno de dsmBeginGetData

Código de retorno	Explicación
DSM_RC_ABORT_INVALID_OFFSET (33)	El desplazamiento que se especificó durante la recuperación de un objeto parcial es mayor que la longitud del objeto.
DSM_RC_ABORT_INVALID_LENGTH (34)	La longitud que se especificó durante la recuperación de un objeto parcial es mayor que la longitud del objeto, o el desplazamiento y la longitud se extienden más allá del final del objeto.
DSM_RC_NO_MEMORY (102)	No queda RAM para finalizar la solicitud.
DSM_RC_NUMOBJ_EXCEED (2029)	**dsmGetList.numObjId** es mayor que DSM_MAX_GET_OBJ.
DSM_RC_OBJID_NOTFOUND (2063)	No se ha encontrado el ID del objeto. No se restauró el objeto.
DSM_RC_WRONG_VERSION_PARM (2065)	La versión de la API del cliente de aplicación es distinta de la versión de la biblioteca de IBM Spectrum Protect.

dsmBeginQuery

La llamada a la función **dsmBeginQuery** inicia una solicitud de consulta al servidor sobre datos, archivos y clases de gestión.

En especial, **dsmBeginQuery** consulta:
- Datos archivados
- Datos de copia de seguridad
- Datos de copia de seguridad activos
- Espacios de archivo
- Clases de gestión

Los datos de la consultan que se extraen de la llamada se obtienen por una o más llamadas a **dsmGetNextQObj**. Cuando se ha completado la consulta, se envía la llamada**dsmEndQuery**.

Sintaxis

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

Parámetros

dsUint32_t dsmHandle (I)

El manejador que asocia esta llamada con una llamada **dsmInitEx** anterior.

dsmQueryType queryType (I)

Identifica el tipo de consulta a ejecutar. Asigne una de las siguientes opciones:

qtArchive

Consultas de objetos archivados.

qtBackup

Consultas de objetos de copias de seguridad.

qtBackupActive

Consultas de objetos de copias de seguridad activas sólo para el nombre del espacio del archivo completo que pase. Esta consulta se denomina "ruta de acceso rápida" y es una forma eficaz de consultar objetos activos del almacenamiento.

Requisito previo: Debe iniciar sesión como usuario root en un sistema operativo UNIX o Linux.

qtFilespace

Consultas de espacios de archivo registrados.

qtMC

Consultas de clases de gestión definidas.

qtBackupGroups

Consultas de grupos que están cerrados.

qtOpenGroups

Consultas de grupos que están abiertos.

qtProxyNodeAuth

Consultas de los nodos de los que puede actuar éste como proxy.

qtProxyNodePeer

Consultas de nodos peer con el mismo objetivo.

dsmQueryBuff *queryBuffer (I)

Identifica un puntero de un almacenamiento intermedio que está correlacionado con una estructura de datos concreta. Esta estructura está asociada con el tipo de consulta transmitida. Estas estructuras pueden contender el criterio de selección de cada tipo de consulta. Rellene los campos de cada estructura con el fin de especificar el ámbito de la consulta que desea ejecutar. El campo `stVersion` de cada estructura contiene el número de versión de estructura.

Las estructuras de datos y sus campos relacionados incluyen los siguientes elementos:

qryArchiveData

objName

El nombre del objeto completo. Puede utilizar un comodín, como por ejemplo un asterisco (*) o un signo de interrogación (?), en la parte de rango alto o de rango bajo del nombre. Un asterisco corresponde a cero o más caracteres, y un interrogante corresponde exactamente a un carácter. El campo `objType` de `objName` puede tener más de uno de los siguientes valores:

- `DSM_OBJ_FILE`
- `DSM_OBJ_DIRECTORY`
- `DSM_OBJ_ANY_TYPE`

Para obtener más información acerca de los nombres de rango alto y de rango bajo, consulte el tema siguiente: “Nombres de alto y bajo rango” en la página 24.

propietario

El nombre del propietario del objeto.

insDateLowerBound

El límite inferior para la fecha de inserción en la que se archivó el objeto. Para obtener el límite superior predeterminado, configure el componente del año en `DATE_MINUS_INFINITE`.

insDateUpperBound

El límite superior para la fecha de inserción en la que se archivó el objeto. Para obtener el límite superior predeterminado, configure el componente del año en `DATE_PLUS_INFINITE`.

expDateLowerBound

El límite inferior de la fecha de caducidad. Los valores predeterminados de ambos campos de fecha de caducidad son los mismos que los campos de fecha de inserción.

expDateUpperBound

El límite superior de la fecha de caducidad.

descr

La descripción del archivado. Entre un asterisco (*) para buscar todas las descripciones.

qryBackupData

objName

El nombre del objeto completo. Puede utilizar un comodín, como por ejemplo un asterisco (*) o un signo de interrogación (?), en la parte de rango alto o de rango bajo del nombre. Un asterisco corresponde a cero o más caracteres, y un interrogante corresponde exactamente a un carácter. El campo `objType` de `objName` puede tener más de uno de los siguientes valores:

- `DSM_OBJ_FILE`
- `DSM_OBJ_DIRECTORY`
- `DSM_OBJ_ANY_TYPE`

Para obtener más información acerca de los nombres de rango alto y de rango bajo, consulte el tema siguiente: “Nombres de alto y bajo rango” en la página 24.

propietario

El nombre del propietario del objeto.

objState

Puede consultar por uno de los siguientes estados de objeto:

- `DSM_ACTIVE`
- `DSM_INACTIVE`

- DSM_ANY_MATCH

pitDate

El valor punto en el tiempo. Una consulta que utiliza este campo devuelve el último objeto al que se le realizó una copia de seguridad antes de dicha fecha y hora. objState puede estar activo o inactivo. Los objetos que se eliminaron antes de pitDate no se devuelven. Por ejemplo:

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

El viernes, llame a la consulta con el punto en el tiempo en miércoles a las 12:00:00, y devuelve la siguiente información:

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

La llamada no devuelve DEF porque ese objeto se eliminó entes del valor de punto en el tiempo.

qryABackupData

objName

El nombre del objeto completo. Puede utilizar un comodín, como por ejemplo un asterisco (*) o un signo de interrogación (?), en la parte de rango alto o de rango bajo del nombre. Un asterisco corresponde a cero o más caracteres, y un interrogante corresponde exactamente a un carácter. El campo objType de objName puede tener más de uno de los siguientes valores:

- DSM_OBJ_FILE
- DSM_OBJ_DIRECTORY
- DSM_OBJ_ANY_TYPE

Para obtener más información acerca de los nombres de rango alto y de rango bajo, consulte el tema siguiente: “Nombres de alto y bajo rango” en la página 24.

qryFSData

fsName

Introduzca el nombre de un espacio de archivo específico en este campo o introduzca un asterisco (*) con el fin de extraer información sobre todos los espacios de archivo registrados.

qryMCData

mcName

Introduzca el nombre de una clase de gestión específica en este campo o introduzca una cadena vacía (“ ”) con el fin de extraer información sobre todas las clases de gestión.

Nota: No puede utilizar un asterisco (*).

mcDetail

Determina si se devuelve o no la información sobre la clase de gestión en la copia de seguridad y en los grupos de copia de archivado. Son válidos los valores siguientes:

- bTrue
- bFalse

qryBackupGroup:

groupType

El tipo de grupo es DSM_GROUPTYPE_PEER.

fsName

El nombre de espacio de archivos.

propietario

ID del propietario.

groupLeaderObjId

ID del objeto líder del grupo.

objType

El tipo de objeto.

qryProxyNodeAuth:

targetNodeName

El nombre de nodo de destino.

peerNodeName

El nombre de nodo del igual.

hlAddress

La dirección del igual del nombre de alto rango.

llAddress

La dirección del igual del nombre de rango bajo.

qryProxyNodePeer:

targetNodeName

El nombre de nodo de destino.

peerNodeName

El nombre de nodo del igual.

hlAddress

La dirección del igual del nombre de alto rango.

llAddress

La dirección del igual del nombre de rango bajo.

Códigos de retorno

La tabla siguiente describe los códigos de retorno de la llamada a función **dsmBeginQuery**.

Tabla 21. Códigos de retorno para dsmBeginQuery

Código de retorno	Número de código de retorno	Explicación
DSM_RC_NO_MEMORY	102	No hay suficiente memoria para completar la solicitud.

Tabla 21. Códigos de retorno para dsmBeginQuery (continuación)

Código de retorno	Número de código de retorno	Explicación
DSM_RC_FILE_SPACE_NOT_FOUND	124	No se ha encontrado el espacio del archivo especificado.
DSM_RC_NO_POLICY_BLK	2007	La información de la política de servidor no está disponible.
DSM_RC_INVALID_OBJTYPE	2010	Tipo de objeto no válido.
DSM_RC_INVALID_OBJOWNER	2019	Nombre de propietario de objeto no válido.
DSM_RC_INVALID_OBJSTATE	2024	Condición de objeto no válida.
DSM_RC_WRONG_VERSION_PARM	2065	La versión de la API del cliente de aplicación es distinta de la versión de la biblioteca de IBM Spectrum Protect.

dsmBeginTxn

La llamada de función **dsmBeginTxn** inicia una o más transacciones IBM Spectrum Protect que a su vez comienzan una acción completa; todas las acciones se procesan correctamente o ninguna de ellas. Una acción puede ser una llamada única o una serie de llamadas. Por ejemplo, una llamada **dsmSendObj** seguida por un número de llamadas **dsmSendData** se puede considerar una acción única. Del mismo modo, una llamada **dsmSendObj** con **dataBlkPtr** que indica el área de datos que contiene el objeto del que se realizará una copia de seguridad también se considera una acción única.

Intente agrupar más de un objeto en una transacción única en las operaciones de transferencia de datos. La agrupación de objetos resulta en mejoras de rendimiento significativas en el sistema IBM Spectrum Protect. Desde una perspectiva de cliente y servidor, esto supone una cierta carga de productividad adicional al iniciar y finalizar cada transacción.

Existen limitaciones con respecto a lo que se puede realizar dentro de una transacción única. Estas transacciones incluyen:

- Un número máximo de objetos que se pueden enviar o eliminar en una transacción única. Este límite se encuentra en los datos que **dsmQuerySessInfo** devuelve en el campo *ApiSessInfo.maxObjPerTxn*. Esto se corresponde con la opción del servidor *TxnGroupMax*.
- Todos los objetos que se envían al servidor (ya sean copias de seguridad o archivados) dentro de una transacción única deben tener el mismo destino de copia que está definido en la vinculación de la clase de gestión del objeto. Este valor se encuentra en los datos que devuelve **dsmBindMC** en los campos **mcBindKey.backup_copy_dest** o **mcBindKey.archive_copy_dest**.

Con la API, o el cliente de la aplicación puede supervisar y controlar estas restricciones, o bien la API. Si es la API la que está supervisando las restricciones, los códigos de retorno adecuados de las llamadas de la API notifican al cliente de la aplicación si se está alcanzando una o más restricciones.

Siempre coincida una llamada **dsmBeginTxn** con una llamada **dsmEndTxn** con el fin de optimizar el conjunto de acciones dentro de un par de llamadas **dsmBeginTxn** y **dsmEndTxn**.

Sintaxis

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

Parámetros

dsUint32_t dsmHandle (I)
El manejador que asocia esta llamada con una llamada **dsmInitEx** anterior.

Códigos de retorno

Los números de código de retorno se proporcionan entre paréntesis).

Tabla 22. Códigos de retorno de dsmBeginTxn

Código de retorno	Explicación
DSM_RC_ABORT_NODE_NOT_AUTHORIZED (36)	FROMNODE o FROMOWNER no está permitido en las operaciones TXN.

dsmBindMC

La llamada de función **dsmBindMC** asocia, o vincula, una clase de gestión al objeto que pasa. El objeto pasa a través de una lista de inclusión/exclusión a la que se apunta en el archivo de opciones. Si no se encuentra una coincidencia de una clase de gestión específica en la lista de inclusión, se asigna la clase de gestión predeterminada. La lista de inclusión puede impedir objetos de una copia de seguridad pero no de una copia de archivado.

El cliente de la aplicación puede utilizar los parámetros que se devuelven en la estructura mcBindKey con el fin de determinar si los objetos deben archivarse o se les debe realizar una copia de seguridad, o si debe iniciarse una transacción nueva debido a distintos destinos de copia. Para obtener más información, consulte el apartado **dsmBeginTxn**.

Llame a **dsmBindMC** antes de llamar a **dsmSendObj** porque todos los objetos deben tener una clase de gestión asociada. Esta llamada puede realizarse dentro o fuera de una transacción. Por ejemplo, dentro de una transacción de varios objetos, si **dsmBindMC** indica que el objeto tiene destinos de copia distintos al objeto anterior, la transacción debe finalizarse y debe iniciarse una nueva. En este caso, no se requiere otra **dsmBindMC** porque ya se ha realizado una en este objeto.

Sintaxis

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

Parámetros

dsUint32_t dsmHandle (I)
El manejador que asocia esta llamada con una llamada **dsmInitEx** anterior.

dsmObjName *objNameP (I)
Un puntero a la estructura que contiene el nombre del espacio del archivo, el nombre de alto y bajo rango y el tipo de objeto.

dsmSendType sendType (I)
Identifica si se realiza esta vinculación de clase de gestión en envíos de copias archivadas o de seguridad. Los valores posibles para esta llamada incluyen:

Nombre	Descripción
stBackup	Una copia de seguridad de objeto
stArchive	Una copia archivada de objeto
stBackupMountWait	Una copia de seguridad de objeto
stArchiveMountWait	Una copia archivada de objeto

En la llamada **dsmBindMC**, stBackup y stBackupMountWait son equivalentes, y stArchive y stArchiveMountWait son equivalentes.

mcBindKey *mcBindKeyP (0)
Esta es la dirección de una estructura mcBindKey donde se devuelve la información de clase de gestión. El cliente de la aplicación puede utilizar la información que se devuelve aquí para determinar si este objeto encaja dentro de una transacción de varios objetos, o para realizar una consulta de clase de gestión en la clase de gestión que está vinculada al objeto.

Códigos de retorno

Los números de código de retorno se proporcionan entre paréntesis ( ).

Tabla 23. Códigos de retorno para dsmBindMC

Código de retorno	Explicación
DSM_RC_NO_MEMORY (102)	No queda RAM para finalizar la solicitud.
DSM_RC_INVALID_PARM (109)	Uno de los parámetros que se pasó tiene un valor no válido.
DSM_RC_TL_EXCLUDED (185)	La copia de seguridad del objeto está excluida y no se puede enviar.
DSM_RC_INVALID_OBJTYPE (2010)	Tipo de objeto no válido.
DSM_RC_INVALID_SENDTYPE (2022)	Tipo de envío no válido.
DSM_RC_WRONG_VERSION_PARM (2065)	La versión de la API del cliente de aplicación es distinta de la versión de la biblioteca de IBM Spectrum Protect.

dsmChangePW

La llamada a función **dsmChangePW** cambia una contraseña de IBM Spectrum Protect. En un sistema operativo de varios usuarios como UNIX o Linux, solo el usuario raíz o el usuario autorizado pueden utilizar esta llamada.

En sistemas operativos Windows, puede especificar la contraseña en el archivo dsm.opt. En este caso, **dsmChangePW** no actualiza el archivo dsm.opt. Después de que se haya realizado la llamada a **dsmChangePW**, debe actualizar el archivo dsm.opt por separado.

Esta llamada debe procesarse correctamente si **dsmInitEx** devuelve DSM_RC_VERIFIER_EXPIRED. La sesión finaliza si la llamada **dsmChangePW** falla en este caso.

Si se llama a **dsmChangePW** por algún otro motivo, la sesión permanece abierta independientemente del código de retorno.

Sintaxis

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

Parámetros

dsUint32_t dsmHandle (I)
El manejador que asocia esta llamada con una llamada **dsmInitEx** anterior.

char *oldPW (I)
La contraseña antigua del llamador. La longitud máxima es DSM_MAX_VERIFIER_LENGTH.

char *newPW (I)
La contraseña nueva del llamador. La longitud máxima es DSM_MAX_VERIFIER_LENGTH.

Códigos de retorno

Los números de código de retorno se proporcionan entre paréntesis ( ).

Tabla 24. Códigos de retorno para dsmChangePW

Código de retorno	Explicación
DSM_RC_ABORT_BAD_VERIFIER (6)	Se ha especificado una contraseña incorrecta.
DSM_RC_AUTH_FAILURE (137)	Error de autenticación. Contraseña antigua incorrecta.
DSM_RC_NEWPW_REQD (2030)	Se debe introducir un valor para la contraseña nueva.
DSM_RC_OLDPW_REQD (2031)	Se debe especificar un valor para la contraseña antigua.
DSM_RC_PASSWD_TOOLONG (2103)	La contraseña especificada es demasiado larga.
DSM_RC_NEED_ROOT (2300)	El llamador de API debe ser un usuario root o un usuario autorizado.

dsmCleanUp

La llamada de función **dsmCleanUp** se utiliza si se ha llamado a **dsmSetUp**. La llamada de función **dsmCleanUp** debe llamarse después de **dsmTerminate**. No es posible realizar ninguna otra llamada después de llamar a **dsmCleanUp**.

No existen códigos de retorno específicos de esta llamada.

Sintaxis

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

Parámetros

dsBool_t mtFlag (I)
Este parámetro especifica que la API fue utilizado en una única hebra o en un modo de varias hebras. Dentro de los valores posibles se incluyen los siguientes:

- DSM_SINGLETHREAD
- DSM_MULTITHREAD

dsmDeleteAccess

La llamada de función **dsmDeleteAccess** elimina las reglas de autorización actuales para las versiones de las copias de seguridad o las copias de archivado de los objetos. Cuando se suprime una regla de autorización, se revoca el acceso de un usuario a los archivos que especifica la regla.

Cuando se utiliza **dsmDeleteAccess**, sólo es posible eliminar las reglas de una en una. Obtenga el ID de la regla mediante el mandato **dsmQueryAccess**.

No existen códigos de retorno específicos de esta llamada.

Sintaxis

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

Parámetros

dsUint32_t dsmHandle (I)
El manejador que asocia esta llamada con una llamada **dsmInitEx** anterior.

dsUint32_t ruleNum (I)
El ID de la regla de acceso de la regla que está eliminada. Este valor se obtiene de una función de llamada **dsmQueryAccess**.

dsmDeleteFS

La llamada de función **dsmDeleteFS** elimina un espacio de archivo del almacenamiento. Para eliminar un espacio de archivo, debe tener permisos apropiados que el administrador IBM Spectrum Protect le habrá otorgado. Para determinar si cuenta con los permisos necesarios, llame a **dsmQuerySessInfo**. Esta llamada de función devuelve una estructura de datos de tipo *ApiSessInfo*, que incluye dos campos, *archDel* y *backDel*.

Nota:

- En un sistema operativo UNIX o Linux, solo un usuario root o un usuario autorizado pueden suprimir un espacio de archivos.
- Si el espacio de archivo que necesita eliminar contiene versiones de copia de seguridad, debe tener autorización para eliminar copias de seguridad (***backDel*** = BACKDEL_YES). Si el espacio de archivo contiene copias archivadas, debe tener autorización para eliminar archivados (*archDel* = ARCHDEL_YES). Si el espacio de archivo contiene tanto copias archivadas como versiones de copia de seguridad, debe tener ambos tipos de autorización para supresión.
- Cuando se utiliza un servidor de gestión de archivado, un espacio de archivo no se puede realmente eliminar. Esta llamada de función devuelve *rc=0* a pesar de que el espacio de archivo no se eliminó. La única forma de comprobar si el espacio de archivo se ha eliminado es emitiendo una consulta de espacio de archivos al servidor.
- La función de supresión del espacio de archivos del servidor IBM Spectrum Protect es un proceso en segundo plano. Si se producen errores distintos a los detectados antes de pasar un código de retorno, se grabarán en el registro del servidor IBM Spectrum Protect.

Sintaxis

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

Parámetros

dsUint32_t dsmHandle (I)
El manejador que asocia esta llamada con una llamada **dsmInitEx** anterior.

char *fsName (I)
Un puntero al nombre del espacio de archivo que se va a eliminar. El carácter comodín no está permitido.

unsigned char repository (I)
Indica si el espacio de archivo que se va a eliminar es un repositorio de copia de seguridad, de archivado o ambos. Los valores posibles para este campo son:

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

Códigos de retorno

Los números de código de retorno se proporcionan entre paréntesis).

Tabla 25. Códigos de retorno para dsmDeleteFS

Código de retorno	Explicación
DSM_RC_ABORT_NOT_AUTHORIZED (27)	No tiene la autorización necesaria para eliminar el espacio de archivo.
DSM_RC_INVALID_REPOS (2015)	Valor de repositorio no válido.
DSM_RC_FSNAME_NOTFOUND (2060)	No se ha encontrado el nombre del espacio de archivo.
DSM_RC_NEED_ROOT (2300)	El llamador de la API no debe ser un usuario root.

dsmDeleteObj

La llamada de función **dsmDeleteObj** inactiva copias de seguridad de objetos, elimina copias de seguridad de objetos o eliminar objetos archivados en el almacenamiento. El tipo **dtBackup** inactiva únicamente la copia de seguridad activa actual. El tipo **dtBackupID** elimina del servidor el ID de objeto especificado. Invoque esta función desde una transacción.

Para obtener más información, consulte el apartado **dsmBeginTxn**.

Antes de enviar **dsmDeleteObj**, envíe la secuencia de consulta que se describen en "Consultas para el sistema IBM Spectrum Protect" en la página 34 para que obtenga la información de **delInfo**. La llamada a **dsmGetNextQObj** devuelve una estructura de datos llamada **qryRespBackupData** a consultas de copias de seguridad o **qryRespArchiveData** a consultas de copias archivadas. Estas estructuras de datos contienen información que necesita en **delInfo**.

El valor de **maxObjPerTxn** determina el número máximo de objetos que se pueden eliminar en una única transacción. Para obtener este valor, llame a **dsmQuerySessInfo**.

Consejo: Su nodo debe tener el permiso adecuado que haya establecido su administrador. Para eliminar objetos archivados, debe tener autorización para

eliminar copias archivadas. Para inactivar un objeto de copia de seguridad, no necesita autorización para suprimir copias de seguridad.

Sintaxis

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

Parámetros

dsUint32_t dsmHandle (I)
El manejador que asocia esta llamada con una llamada **dsmInitEx** anterior.

dsmDelType delType (I)
Indica qué tipo de objeto (copia de seguridad o archivada) se va a eliminar. Dentro de los valores posibles se incluyen los siguientes:

Nombre	Descripción
dtArchive	El objeto que se va a eliminar se archivó anteriormente.
dtBackup	Al objeto que se va a desactivar se le ha realizado una copia de seguridad anteriormente.
dtBackupID	Al objeto que se va a desactivar se le ha realizado una copia de seguridad anteriormente. **Restricción:** Al utilizar este **delType** con *objID* se elimina la copia de seguridad del objeto del servidor. Sólo un propietario del objeto puede eliminarlo. Es posible eliminar cualquier versión (activa o inactiva) de un objeto. El servidor reconcilia las versiones. Si elimina una versión activa de un objeto, la primera versión inactiva se transforma en activa. Si elimina una versión inactiva de un objeto, todas las versiones antiguas avanzan. El nodo debe estar registrado con el permiso **backDel**.

dsmDelInfo delInfo (I)
Una estructura cuyos campos identifican al objeto. Los campos son distintos, dependiendo de si el objeto es una copia de seguridad o una copia archivada. La estructura para desactivar una copia de seguridad de un objeto `delBack`, contiene el nombre del objeto y el grupo de la copia del objeto. La estructura de un objeto archivado, `delArch`, contiene el ID del objeto.

La estructura para eliminar una copia de seguridad de un objeto `delBackID`, contiene el ID del objeto.

Códigos de retorno

Los números de código de retorno se proporcionan entre paréntesis).

Tabla 26. Códigos de retorno para dsmDeleteObj

Código de retorno	Explicación
DSM_RC_FS_NOT_REGISTERED (2061)	El nombre del espacio de archivos no está registrado.
DSM_RC_WRONG_VERSION_PARM (2065)	La versión de la API del cliente de aplicación es distinta de la versión de la biblioteca de IBM Spectrum Protect.

dsmEndGetData

La llamada de función **dsmEndGetData** finaliza una sesión **dsmBeginGetData** que obtiene objetos de un almacenamiento.

La llamada de función **dsmEndGetData** se inicia después de que todos los objetos que desea restaurar se hayan procesado, o bien finalizar el proceso de obtención de forma prematura. Llame a **dsmEndGetData** para finalizar una sesión **dsmBeginGetData** antes de poder continuar otro proceso.

Dependiendo de cuándo se llame a **dsmEndGetData**, es posible que la API necesite finalizar el proceso parcial de la corriente de datos antes de que se pueda detener el proceso. El llamador, por lo tanto, no puede esperar un retorno inmediato de esta llamada. Utilice **dsmTerminate** si la aplicación necesita cerrar la sesión y finalizar la restauración de forma inmediata.

No existen códigos de retorno específicos de esta llamada.

Sintaxis

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

Parámetros

dsUint32_t dsmHandle (I)
El manejador que asocia esta llamada con una llamada **dsmInitEx** anterior.

dsmEndGetDataEx

La llamada a función **dsmEndGetDataEx** proporciona el total de bytes fuera de LAN enviados. Es una extensión de la llamada a función **dsmEndGetData**.

Sintaxis

No existen códigos de retorno específicos de esta llamada.

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

Parámetros

dsmEndGetDataExIn_t *dsmEndGetDataExInP (I)
Pasa el objeto final dsmHandle que identifica la sesión y la asocia con llamadas subsiguientes.

dsmEndGetDataExOut_t *dsmEndGetDataExOutP (O)
Esta estructura contiene este parámetro de entrada:

totalLFBytesRecv
El total de bytes de fuera de la LAN que se han recibido.

dsmEndGetObj

La llamada a función **dsmEndGetObj** finaliza una sesión **dsmGetObj** que obtiene datos para un objeto específico.

Inicia la llamada **dsmEndGetObj** después de que se recibe una finalización de datos del objeto. Esto indica que todos los datos se han recibido, o que no se van a recibir más datos de este objeto. Antes de iniciar otra llamada **dsmGetObj**, debe llamar a **dsmEndGetObj**.

En función del momento en que se invoca **dsmEndGetObj**, es posible que la API deba finalizar el proceso de una corriente de datos parcial antes de que se detenga el proceso. No espere una devolución inmediata de esta llamada.

Sintaxis

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

Parámetros

dsUint32_t dsmHandle (I)
El manejador que asocia esta llamada con una llamada **dsmInitEx** anterior.

Códigos de retorno

Los números de código de retorno se proporcionan entre paréntesis ( ).

Tabla 27. Códigos de retorno para dsmEndGetObj

Código de retorno	Explicación
DSM_RC_NO_MEMORY (102)	No queda RAM para completar la solicitud.

dsmEndQuery

La llamada de función **dsmEndQuery** significa el final de una acción **dsmBeginQuery**. El cliente de la aplicación envía **dsmEndQuery** con objeto de finalizar una consulta. Esta llamada se envía después de obtener todas las respuestas de la consulta a través de **dsmGetNextQObj**, o se envía para finalizar una consulta antes de que se devuelvan todos los datos.

Consejo: En este caso, IBM Spectrum Protect continúa enviando los datos de la consulta desde el servidor al cliente, pero la API descarta cualquier dato restante.

Una vez enviado **dsmBeginQuery**, debe enviarse un **dsmEndQuery** antes de iniciar cualquier otra actividad.

No existen códigos de retorno específicos de esta llamada.

Sintaxis

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

Parámetros

dsUint32_t dsmHandle (I)
El manejador que asocia esta llamada con una llamada **dsmInitEx** anterior.

dsmEndSendObj

La llamada de función **dsmEndSendObj** indica el final de los datos enviados a almacenamiento.

Introduzca la llamada de función **dsmEndSendObj** para indicar el final de los datos de las llamadas **dsmSendObj** y **dsmSendData**. Si esto no se lleva a cabo ocurre una infracción de protocolo. La excepción de la regla se produce si llama a **dsmEndTxn** para finalizar la transacción. Esto elimina todos los datos que se enviaron para la transacción.

Sintaxis

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

Parámetros

dsUint32_t dsmHandle (I)
El manejador que asocia esta llamada con una llamada **dsmInitEx** anterior.

Códigos de retorno

Los números de código de retorno se proporcionan entre paréntesis).

Tabla 28. Códigos de retorno para dsmEndSendObj

Código de retorno	Explicación
DSM_RC_NO_MEMORY (102)	No queda RAM para finalizar la solicitud.

dsmEndSendObjEx

La llamada de función **dsmEndSendObjEx** proporciona información adicional y el número de bytes que se procesaron. La información incluye: el total de bytes enviados, información de compresión, bytes fuera de la LAN e información sobre la optimización de almacenamiento.

La llamada de función **dsmEndSendObjEx** es una extensión de la llamada de función **dsmEndSendObj**.

Sintaxis

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

Parámetros

dsmEndSendObjExIn_t *dsmEndSendObjExInP (I)
Este parámetro pasa el dsmhandle del objeto de envío final que identifica la sesión y lo asocia con las llamadas posteriores.

dsmEndSendObjExOut_t *dsmEndSendObjExOutP (O)
Este parámetro pasa la información de envío final del objeto:

Nombre	Descripción
totalBytesSent	El número total de bytes que se leen desde la aplicación.
objCompressed	Un identificador que muestra si se ha comprimido el objeto.
totalCompressedSize	El tamaño total en bytes después de la compresión.
totalLFBytesSent	Los bytes sin LAN totales que se enviaron.

Nombre	Descripción
objDeduplicated	Un indicador que muestra si la interfaz de programación de aplicaciones (API) ha realizado una optimización de almacenamiento en el objeto.
totalDedupSize	Total de bytes enviados durante la optimización de almacenamiento.

Códigos de retorno

Los números de código de retorno se proporcionan entre paréntesis ( ).

Tabla 29. Códigos de retorno para dsmEndSendObjEx

Código de retorno	Explicación
DSM_RC_NO_MEMORY (102)	No queda RAM para finalizar la solicitud.

dsmEndTxn

La llamada a función **dsmEndTxn** finaliza una transacción de IBM Spectrum Protect. Empareje la llamada de función **dsmEndTxn** con **dsmBeginTxn** para identificar a la llamada o conjunto de llamadas que se consideran una transacción. El cliente de aplicación puede especificar en la llamada **dsmEndTxn** si se debe confirmar o finalizar la transacción.

Realice todas estas llamadas dentro de los límites de una transacción:

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

Sintaxis

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

Parámetros

dsUint32_t dsmHandle (I)
El manejador que asocia esta llamada con una llamada **dsmInitEx** anterior.

dsUint8_t vote (I)
Indica si el cliente de aplicación confirma todas las acciones realizadas entre la llamada **dsmBeginTxn** previa y esta llamada. Son posibles los valores siguientes:

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

Utilice DSM_VOTE_ABORT únicamente si la aplicación tiene un motivo para detener la transacción.

dsUint16_t *reason (O)
Si la llamada a **dsmEndTxn** finaliza con un error o el valor de vote no está de acuerdo, este parámetro tiene un código de razón que indica porqué falló el voto. El código de retorno para la llamada puede ser cero, y el código de motivo no. Por lo tanto, el cliente de aplicación debe siempre comprobar los

errores tanto en el código de retorno como en el código de motivo (`if (rc || reason)`) antes de dar por hecho de que la operación ha finalizado correctamente.

Si la aplicación especifica un voto de `DSM_VOTE_ABORT`, el código de razón es `DSM_RS_ABORT_BY_CLIENT (3)`. Consulte Apéndice A, "Archivo de origen de códigos de retorno de la API: `dsmrc.h`", en la página 155 para obtener una lista de los códigos de motivo posibles. Los números del 1 al 50 de la lista de códigos de retorno están reservados para los códigos de motivo. Si el servidor finaliza la transacción, el código de retorno es `DSM_RC_CHECK_REASON_CODE`. En este caso, el valor de motivo contiene más información sobre la causa de la anulación.

Códigos de retorno

Los números de código de retorno se proporcionan entre paréntesis ( ).

Tabla 30. Códigos de retorno para dsmEndTxn

Código de retorno	Explicación
`DSM_RC_ABORT_CRC_FAILED (236)`	El CRC que se recibió del servidor no coincide con el CRC calculado por el cliente.
`DSM_RC_INVALID_VOTE (2011)`	El valor que se especificó para `vote` no es válido.
`DSM_RC_CHECK_REASON_CODE (2302)`	La transacción se anuló. Compruebe el campo de motivo.
`DSM_RC_ABORT_STGPOOL_COPY_CONT_NO (241)`	La grabación a una de las agrupaciones de almacenamiento de copia ha fallado, y la opción de agrupación de almacenamiento IBM Spectrum Protect de `COPYCONTINUE` se establece en `NO`. La transacción finaliza.
`DSM_RC_ABORT_RETRY_SINGLE_TXN (242)`	Este código de anulación indica que la transacción actual fue anulada debido a un problema durante la operación de almacenamiento. El problema puede resolverse enviando cada uno de los archivos en una transacción separada. Este error es típico en las siguientes circunstancias: • La siguiente agrupación de almacenamiento tiene una lista de agrupación de almacenamiento de copias diferente. • La operación se conmuta a esta agrupación a la mitad de una transacción. .

dsmEndTxnEx

La llamada de función **dsmEndTxnEx** proporciona la información de ID de objeto de líder de grupo para que se utilice con la llamada de función **dsmGroupHandler**. Es una extensión de llamada a función **dsmEndTxn**.

Sintaxis

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

Parámetros

dsmEndTxnExIn_t *dsmEndTxnExInP (I)
Esta estructura contiene los siguientes parámetros:

dsmHandle
El manejador que identifica la sesión y la asocia con llamadas IBM Spectrum Protect subsiguientes.

dsUint8_t vote (I)
Indica si el cliente de la aplicación compromete todas las acciones que se realizan entre la llamada **dsmBeginTxn** previa y esta llamada. Los valores posibles son:

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

Utilice DSM_VOTE_ABORT únicamente si la aplicación tiene un motivo para detener la transacción.

dsmEndTxnExOut_t *dsmEndTxnExOutP (O)
Esta estructura contiene los siguientes parámetros:

dsUint16_t *reason (O)
Si la llamada a **dsmEndTxnEx** finaliza con un error o el valor de *vote* no está de acuerdo, este parámetro tiene un código de motivo que indica por qué el voto ha fallado.

Consejo: El código de retorno para la llamada puede ser cero, y el código de motivo no. Por lo tanto, el cliente de aplicación siempre debe comprobar los errores, tanto en el código de retorno como en el código de motivo (`if (rc || reason)`), antes de presuponer que la operación ha finalizado correctamente.

Si la aplicación especifica un voto de DSM_VOTE_ABORT, el código de motivo es DSM_RS_ABORT_BY_CLIENT (3). Consulte Apéndice A, "Archivo de origen de códigos de retorno de la API: `dsmrc.h`", en la página 155 para obtener una lista de los códigos de motivo posibles. Los números del 1 al 50 de la lista de códigos de retorno están reservados para los códigos de motivo. Si el servidor finaliza la transacción, el código de retorno es DSM_RC_CHECK_REASON_CODE. En este caso, el valor de motivo contiene más información sobre la causa de la anulación.

groupLeaderObjId
El ID del objeto líder del grupo que se devuelve cuando se utiliza el indicador DSM_ACTION_OPEN con la llamada **dsmGroupHandler**.

Códigos de retorno

Los números de código de retorno se proporcionan entre paréntesis ( ).

Tabla 31. Códigos de retorno para dsmEndTxnEx

Código de retorno	Explicación
DSM_RC_INVALID_VOTE (2011)	El valor que se especificó para el voto no es válido.
DSM_RC_CHECK_REASON_CODE (2302)	La transacción se anuló. Compruebe el campo de motivo.
DSM_RC_ABORT_STGPOOL_COPY_CONT_NO (241)	La grabación a una de las agrupaciones de almacenamiento de copia ha fallado, y la opción COPYCONTINUE de agrupación de almacenamiento IBM Spectrum Protect se ha configurado en NO. La transacción termina.
DSM_RC_ABORT_RETRY_SINGLE_TXN (242)	Durante una operación de grabación simultánea, un objeto en la transacción va a un destino con diferentes agrupaciones de almacenamiento de copia. Finalice la transacción actual y envíe cada uno de los objetos de nuevo en su propia transacción.

dsmGetData

La llamada de función **dsmGetData** obtiene una corriente de byte de datos de IBM Spectrum Protect y la coloca en el almacenamiento intermedio del llamador. El cliente de la aplicación llama a **dsmGetData** cuando hay más datos para recibir de una llamada previa **dsmGetObj** o **dsmGetData**.

Sintaxis

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

Parámetros

dsUint32_t dsmHandle (I)
El manejador que asocia esta llamada con una llamada **dsmInitEx** anterior.

DataBlk *dataBlkPtr (I/O)
Apunta a una estructura que incluye tanto un puntero al almacenamiento intermedio de los datos que se reciben como el tamaño del almacenamiento intermedio. Cuando se devuelven los datos, esta estructura contiene el número de bytes que se han transferido. Consulte Apéndice B, "Archivos de origen de definiciones de tipo de API", en la página 165 para obtener más información sobre la definición del tipo.

Códigos de retorno

Los números de código de retorno se proporcionan entre paréntesis).

Tabla 32. Códigos de retorno para dsmGetData

Código de retorno	Explicación
DSM_RC_ABORT_INVALID_OFFSET (33)	El desplazamiento especificado durante la recuperación de un objeto parcial es mayor que la longitud del objeto.
DSM_RC_ABORT_INVALID_LENGTH (34)	La longitud que se especificó durante la recuperación de un objeto parcial es mayor que la longitud del objeto, o el desplazamiento y la longitud se extienden más allá del final del objeto.
DSM_RC_FINISHED (121)	Proceso finalizado. Se ha recibido el último almacenamiento intermedio. Compruebe la cantidad de datos en `numBytes` y a continuación llame a IBM Spectrum ProtectdsmEndGetObj.
DSM_RC_NULL_DATABLKPTR (2001)	El puntero de bloqueo de datos es nulo.
DSM_RC_ZERO_BUFLEN (2008)	La longitud del almacenamiento intermedio es cero en el puntero del bloque de datos.
DSM_RC_NULL_BUFPTR (2009)	El puntero del almacenamiento intermedio es nulo en el puntero de bloque de datos.
DSM_RC_WRONG_VERSION_PARM (2065)	La versión de la API del cliente de aplicación es diferente de la versión de la biblioteca de IBM Spectrum Protect.
DSM_RC_MORE_DATA (2200)	Quedan más datos por obtener.

dsmGetBufferData

La llamada a función **dsmGetBufferData** recibe una secuencia de bytes de datos de IBM Spectrum Protect a través de un almacenamiento intermedio. Después de cada llamada la aplicación necesita copiar los datos y liberar el almacenamiento intermedio a través de una llamada a **dsmReleaseBuffer**. Si el número de almacenamientos intermedios que retiene la aplicación es igual al numTsmBuffers especificado en la llamada **dsmInitEx**, se bloquea la función **dsmGetBufferData** hasta que se invoque **dsmReleaseBuffer**.

Sintaxis

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

Parámetros

getDataExIn_t * dsmGetBufferDataExInP (I)
Esta estructura contiene los siguientes parámetros de entrada.

dsUint32_t dsmHandle
El manejador que identifica la sesión y la asocia con una llamada **dsmInitEx** previa.

getDataExOut_t * dsmGetBufferDataExOutP (0)
Esta estructura contienen los siguientes parámetros de salida.

dsUint8_t tsmBufferHandle(0)
El manejador que identifica el almacenamiento intermedio recibido.

char *dataPtr(0)
La dirección en la que se escriben los datos.

dsUint32_t numBytes(0)
El número real de bytes grabados por IBM Spectrum Protect.

Códigos de retorno

Los números de código de retorno se proporcionan entre paréntesis).

Tabla 33. Códigos de retorno para dsmGetBufferData

Código de retorno	Explicación
DSM_RC_BAD_CALL_SEQUENCE (2041)	La llamada no se ha emitido en el estado adecuado.
DSM_RC_OBJ_ENCRYPTED (2049)	Esta función no se puede utilizar en objetos cifrados.
DSM_RC_OBJ_COMPRESSED (2048)	Esta función no se puede utilizar en objetos comprimidos.
DSM_RC_BUFF_ARRAY_ERROR (2045)	Se ha producido un error de agrupación de almacenamiento intermedio.

dsmGetNextQObj

La llamada a función **dsmGetNextQObj** obtiene la siguiente respuesta de consulta de una llamada **dsmBeginQuery** y coloca la respuesta en el almacenamiento intermedio del interlocutor.

La llamada **dsmGetNextQObj** se invoca una o más veces. Cada vez que se llama a una función, se recupera un único registro de consulta o un error o se devuelve un código de razón DSM_RC_FINISHED. Si se devuelve DSM_RC_FINISHED , no hay más datos a procesar. Cuando se recuperan todos los datos de consulta, o si no se necesitan más datos de consulta, envíe la llamada **dsmEndQuery** para finalizar el proceso de consulta.

El parámetro **dataBlkPtr** debe apuntar a un almacenamiento intermedio definido con el tipo de estructura **qryResp*Data**. El contexto en el que se llama a **dsmGetNextQObj** determina el tipo de estructura que se especifica en la respuesta de la consulta.

Sintaxis

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

Parámetros

dsUint32_t dsmHandle (I)
El manejador que asocia esta llamada con una llamada **dsmInitEx** anterior.

DataBlk *dataBlkPtr (I/O)
Apunta a una estructura que incluye tanto un puntero al almacenamiento intermedio de los datos que se reciben como el tamaño del almacenamiento intermedio. Este almacenamiento intermedio es la estructura de respuesta de **qryResp*Data**. A cambio, esta estructura contiene el número de bytes que se han transferido. Esta estructura que está asociada con cada tipo de consulta se describe en la tabla siguiente. Para obtener más información acerca de la definición de tipo de**DataBlk**, consulte el tema siguiente: Apéndice B, "Archivos de origen de definiciones de tipo de API", en la página 165.

Tabla 34. estructura de puntero DataBlk

Consulta	Estructura de respuesta	Campos de interés
qtArchive	**qryRespArchiveData**	**sizeEstimate** Contiene el valor que se pasa en una llamada previa **dsmSendObj**. **mediaClass** Puede tener un valor de MEDIA_FIXED si el objeto está en el disco, o MEDIA_LIBRARY si el objeto está en la cinta. **clientDeduplicated** Indica si este objeto está duplicado por el cliente.

Tabla 34. estructura de puntero DataBlk (continuación)

Consulta	Estructura de respuesta	Campos de interés
qtBackup	**qryRespBackupData**	**restoreOrderExt** Es de tipo dsUint16_t. Ordene este campo cuando se restauran varios objetos en una llamada **dsmBeginGetData**. En el ejemplo de API, dapiqry.c, se muestra cómo clasificar el código para esta llamada. Para ver un ejemplo de clasificación, consulte el tema siguiente: Figura 16 en la página 71. **sizeEstimate** Contiene el valor que se pasa en una llamada previa **dsmSendObj**. **mediaClass** Puede tener un valor de MEDIA_FIXED si el objeto está en el disco, o MEDIA_LIBRARY si el objeto está en la cinta. **clientDeduplicated** Indica si este objeto está duplicado por el cliente.
qtBackupActive	**qryARespBackupData**	
qtBackupGroups	**qryRespBackupData**	**dsBool_t isGroupLeader** Si es verdadero, significa que este objeto es un líder de grupo.
qtOpenGroups	**qryRespBackupData**	**dsBool_t isOpenGroup;** Si es verdadero, significa que este grupo es abierto y no está completo.

Tabla 34. estructura de puntero DataBlk (continuación)

Consulta	Estructura de respuesta	Campos de interés
qtFilespace	**qryRespFSData**	**backStartDate** Contiene la indicación de fecha y hora del servidor cuando el espacio de archivo está actualizado con la acción **backStartDate**. **backCompleteDate** Contiene la indicación de fecha y hora del servidor cuando el espacio de archivo está actualizado con la acción **backCompleteDate**. **lastReplStartDate** Contiene la indicación de fecha y hora de la última vez que se inició una réplica en el servidor. **lastReplCmpltDate** Contiene la indicación de fecha y hora de la última vez que se completó una réplica, incluso si se produjo un error. **lastBackOpDateFromServer** Contiene la última indicación de fecha y hora que se guardó en el servidor. **lastBackOpDateFromLocal** Contiene la última indicación de fecha y hora que se guardó en el cliente.
qtMC	**qryRespMCData** **qryRespMCDetailData**	
qtProxyNodeAuth	**qryRespProxyNodeData** **targetNodeName** **peerNodeName** **hlAddress** **llAddress**	
qtProxyNodePeer	**qryRespProaxyNodeData** **targetNodeName** **peerNodeName** **hlAddress** **llAddress**	

Códigos de retorno

La tabla siguiente describe los códigos de retorno de la llamada a función **dsmGetNextQObj**.

Tabla 35. Códigos de retorno para la llamada a función **dsmGetNextQObj**

Código de retorno	Número de código de retorno	Descripción
DSM_RC_ABORT_NO_MATCH	2	No se ha solicitado una coincidencia para la consulta.

Tabla 35. Códigos de retorno para la llamada a función **dsmGetNextQObj** *(continuación)*

Código de retorno	Número de código de retorno	Descripción
DSM_RC_FINISHED	121	Proceso finalizado (iniciar **dsmEndQuery**). No hay más datos para procesar.
DSM_RC_UNKNOWN_FORMAT	122	El archivo que IBM Spectrum Protect ha intentado restaurar o recuperar tiene un formato desconocido.
DSM_RC_COMM_PROTOCOL_ERROR	136	Error del protocolo de comunicaciones.
DSM_RC_NULL_DATABLKPTR	2001	El puntero no apunta a un bloque de datos.
DSM_RC_INVALID_MCNAME	2025	Nombre de clase de gestión no válido.
DSM_RC_BAD_CALL_SEQUENCE	2041	La secuencia de llamadas no es válida.
DSM_RC_WRONG_VERSION_PARM	2065	La versión de la API del cliente de aplicación es diferente de la versión de biblioteca de IBM Spectrum Protect.
DSM_RC_MORE_DATA	2200	Quedan más datos por obtener.
DSM_RC_BUFF_TOO_SMALL	2210	El almacenamiento intermedio es demasiado pequeño.

dsmGetObj

La llamada de función **dsmGetObj** obtiene los datos de objeto solicitados de la corriente de datos IBM Spectrum Protect y los coloca en el almacenamiento intermedio del llamador. La llamada **dsmGetObj** utiliza el ID de objeto para obtener el siguiente objeto u objeto parcial de la corriente de datos.

Los datos del objeto indicado se colocan en el almacenamiento intermedio al que apunta **DataBlk**. Si hay más datos disponibles, debe realizar una o más llamadas a **dsmGetData** con el fin de recibir los datos del objeto restantes hasta que se devuelva un código de retorno DSM_RC_FINISHED. Compruebe el campo numBytes en **DataBlk** para ver si quedan datos en el almacenamiento intermedio.

Los objetos deben consultarse en el orden en el aparecen en la lista en la llamada **dsmBeginGetData** del parámetro **dsmGetList**. La excepción ocurre cuando el cliente de la aplicación necesita saltarse un objeto en la corriente de datos para obtener un objeto más tarde en la lista. Si el objeto indicado por el ID del objeto no es el siguiente de la corriente, la corriente de datos se procesa hasta que el objeto se localiza o hasta que se finaliza la corriente. Utilice esta función con atención, ya que es posible que sea necesario procesar y descartar grandes cantidades de datos para localizar el objeto solicitado.

Requisito: Si **dsmGetObj** devuelve un código de error (NOT FINISHED o MORE_DATA), la sesión deberá finalizarse para detener la operación de restauración. Esto es especialmente importante si utiliza cifrado y recibe un RC_ENC_WRONG_KEY. Debe iniciar una nueva sesión con la clave correcta.

Sintaxis

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

Parámetros

dsUint32_t dsmHandle (I)
El manejador que asocia esta llamada con una llamada **dsmInitEx** anterior.

ObjID *objIdP (I)
Un puntero al ID del objeto que se va a restaurar.

DataBlk *dataBlkPtr (I/O)
Un puntero al almacenamiento intermedio donde se colocan los datos restaurados.

Códigos de retorno

Los números de código de retorno se proporcionan entre paréntesis).

Tabla 36. Códigos de retorno para dsmGetObj

Código de retorno	Explicación
DSM_RC_ABORT_INVALID_OFFSET (33)	El desplazamiento que se especificó durante la recuperación de un objeto parcial es mayor que la longitud del objeto.
DSM_RC_ABORT_INVALID_LENGTH (34)	La longitud especificada durante la recuperación de un objeto parcial es mayor que la longitud del objeto, o el desplazamiento y la longitud alcanzan más allá del final del objeto.
DSM_RC_FINISHED (121)	Proceso finalizado (iniciar **dsmEndGetObj**).
DSM_RC_WRONG_VERSION_PARM (2065)	La versión de la API del cliente de aplicación es distinta de la versión de la biblioteca de IBM Spectrum Protect.
DSM_RC_MORE_DATA (2200)	Quedan más datos por obtener.
RC_ENC_WRONG_KEY (4580)	La clave que se proporciona en la llamada **dsmInitEx**, o la clave guardada, no coinciden con la clave que se utilizó para cifrar este objeto. Termine la sesión y proporcione la clave adecuada.

dsmGroupHandler

La llamada de función **dsmGroupHandler** realiza una acción en un grupo de archivo lógico dependiendo de los datos introducidos. El cliente relaciona un número de objetos con el fin de hacer referencia y administrar el servidor IBM Spectrum Protect como grupo lógico.

Para obtener más información, consulte el apartado “Agrupación de archivos” en la página 66.

Sintaxis

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

Parámetros

dsmGroupHandlerIn_t *dsmGroupHandlerInP (I)
Pasa los atributos del grupo al API.

groupType
El tipo de grupo. Los valores incluyen:
- DSM_GROUPTYPE_PEER - peer group

actionType
La acción que se va a ejecutar. Los valores incluyen:
- DSM_GROUP_ACTION_OPEN - crea un grupo nuevo
- DSM_GROUP_ACTION_CLOSE - valida y guarda un grupo abierto
- DSM_GROUP_ACTION_ADD - añade a un grupo
- DSM_GROUP_ACTION_ASSIGNTO - asigna a otro grupo
- DSM_GROUP_ACTION_REMOVE- elimina un miembro de un grupo

memberType.
El tipo de grupo del objeto. Los valores incluyen:
- DSM_MEMBERTYPE_LEADER - líder de grupo
- DSM_MEMBERTYPE_MEMBER - miembro de grupo

***uniqueGroupTagP**
Un ID de cadena único que está asociado a un grupo.

leaderObjId
El ID de objeto para el líder del grupo.

***objNameP**
Un puntero al nombre del objeto del líder del grupo.

memberObjList
Una lista de objetos para eliminar o asignar.

dsmGroupHandlerOut_t *dsmGroupHandlerOutP (0)
Pasa la dirección de la estructura que completa la API. Se devuelve el número de la versión de la estructura.

Códigos de retorno

Los números de código de retorno se proporcionan entre paréntesis).

Tabla 37. Códigos de retorno para dsmGroupHandler

Código de retorno	Explicación
DSM_RC_ABORT_INVALID_GROUP_ACTION (237)	Se intentó realizar una operación no válida en un líder o miembro de grupo.

dsmInit

La llamada de función **dsmInit** inicia una sesión de la API y conecta al cliente con el almacenamiento IBM Spectrum Protect. El cliente de la aplicación sólo puede tener una sesión activa abierta a la vez. Para abrir otra sesión con parámetros distintos, utilice la llamada **dsmTerminate** primero con objeto de finalizar la sesión actual.

Para permitir la consulta entre nodos y restaurar o recuperar datos, utilice las opciones de cadena *-fromnode* y *-fromowner* . Para obtener más información, consulte el apartado “Acceder a objetos de nodos y propietarios” en la página 26.

Sintaxis

```
dsInt16_t  dsmInit  (dsUint32_t       *dsmHandle,
   dsmApiVersion *dsmApiVersionP,
   char          *clientNodeNameP,
   char          *clientOwnerNameP,
```

```
char          *clientPasswordP,
char          *applicationType,
char          *configfile,
char          *options);
```

Parámetros

dsUint32_t *dsmHandle (O)
El manejador que identifica la sesión de iniciación y que la asocia con llamadas subsiguientes IBM Spectrum Protect.

dsmApiVersion *dsmApiVersionP (I)
Un puntero a la estructura de datos que identifica la versión de la API que el cliente de la aplicación está utilizando en esta sesión. La estructura contiene los valores de tres constantes, DSM_API_VERSION, DSM_API_RELEASE y DSM_API_LEVEL, que se configuran en el archivo dsmapitd.h. Se debe realizar una llamada previa a **dsmQueryApiVersion** para garantizar que hay compatibilidad entre la versión de la interfaz de programación de aplicaciones (API) de la aplicación y la versión de la biblioteca de la API instalada en la estación de trabajo del usuario.

char *clientNodeNameP (I)
Este parámetro es un puntero al nodo de la sesión IBM Spectrum Protect. Todas las sesiones deben tener un nombre de nodo asociado con ellas. La constante DSM_MAX_NODE_LENGTH, en el archivo dsmapitd.h configura el tamaño máximo permitido del nombre del nodo.

El nombre del nodo no distingue entre mayúsculas y minúsculas.

Si este parámetro está configurado en NULO, y *passwordaccess* está configurado en *prompt*, la API intenta obtener primero el nombre del nodo de la cadena de opciones que se pasaron. Si no se encuentra allí, entonces la API intenta obtener el nombre del nodo del archivo de configuración o de los archivos de opciones. Si este intento de obtener el nombre de nodo falla, la API de UNIX o Linux utiliza el nombre del host del sistema, mientras que las API de otros sistemas operativos utilizan el código de retorno DSM_RC_REJECT_ID_UNKNOWN.

Este parámetro debe ser NULO si la opción *passwordaccess* en el archivo dsm.sys está configurada en *generate*. La API entonces utiliza el nombre del host del sistema.

char *clientOwnerNameP (I)
Este parámetro es un puntero al propietario de la sesión IBM Spectrum Protect. Si el sistema operativo en donde se inicia la sesión es un sistema de varios usuarios, un nombre de propietario de NULO (el usuario root) tiene la autorización de realizar copias de seguridad, archivar, restaurar o recuperar objetos que pertenecen a la aplicación, independientemente del propietario del objeto.

El nombre del propietario es sensible a mayúsculas y minúsculas.

Este parámetro debe ser NULO si la opción *passwordaccess* en el archivo dsm.sys está configurada en *generate*. La API entonces utiliza el ID del usuario del inicio de sesión.

Nota: En un sistema operativo de varios usuarios, si *passwordaccess* está configurado en *prompt*, no es necesario que el nombre del propietario se correlacione con el ID de usuario activo de la sesión que ejecuta la aplicación.

char *clientPasswordP (I)
Este parámetro es un puntero a la contraseña del nodo en el que se ejecuta la

sesión IBM Spectrum Protect. La constante DSM_MAX_VERIFIER_LENGTH en el archivo dsmapitd.h configura el tamaño máximo permitido de una contraseña.

La contraseña no es sensible a mayúsculas y minúsculas.

Excepto cuando se inicia primero el archivo de contraseña, el valor del este parámetro se ignora si *passwordaccess* está configurado en *generate*.

char *applicationType (I)
Este parámetro identifica la aplicación que está ejecutando la sesión. El cliente de la aplicación define el valor.

Cada vez que un cliente de aplicación de la API inicia una sesión con el servidor, el tipo de aplicación (o plataforma) del cliente se actualiza en el servidor. Recomendamos que el valor del tipo de aplicación contenga una abreviatura del sistema operativo porque este valor se introduce en el campo **platform** del servidor. La longitud máxima de la cadena es DSM_MAX_PLATFORM_LENGTH.

Para ver el valor actual del tipo de aplicación, llame a **dsmQuerySessInfo**.

char *configfile (I)
Este parámetro apunta a una cadena de caracteres que contiene el nombre calificado al completo del archivo de configuración de la API. Las opciones que se especifican en el archivo de configuración de la API modifican sus especificaciones en el archivo de opciones del cliente. Los archivos de opciones se definen cuando se instala IBM Spectrum Protect (cliente o API).

char *options (I)
Apunta a una cadena de caracteres que puede contener opciones de archivo como:

- *Compressalways*
- *Servername* (UNIX o Linux únicamente)
- *TCPServeraddr*
- *Fromnode*
- *Fromowner*
- *EnableClientEncryptKey*

El cliente de la aplicación puede utilizar la lista de opciones para modificar los valores de estas opciones definidas por el archivo de configuración.

El formato de la opción es:
1. Cada opción que se especifica en la lista de opción empieza con un guión (-) y lo sigue la palabra clave de la opción.
2. A su vez, a la palabra clave le sigue un signo de igual (=) y a continuación el parámetro de la opción.
3. Si el parámetro de la opción contiene un espacio en blanco, cierre el parámetro con comillas sencillas o dobles.
4. Si se especifica más de una opción, separe las opciones con espacios en blanco.

Si las opciones son NULO, los valores de todas las opciones se toman del archivo de opciones del usuario o del archivo de configuración de la API.

Códigos de retorno

Los números de código de retorno se proporcionan entre paréntesis ( ).

Tabla 38. Códigos de retorno de dsmInit

Código de retorno	Explicación
DSM_RC_ABORT_SYSTEM_ERROR (1)	El servidor ha detectado un error del sistema y ha notificado a los clientes.
DSM_RC_REJECT_VERIFIER_EXPIRED (52)	La contraseña ha caducado y debe actualizarse.
DSM_RC_REJECT_ID_UNKNOWN (53)	No se ha encontrado el nombre de nodo.
DSM_RC_AUTH_FAILURE (137)	Ha ocurrido un fallo de autenticación.
DSM_RC_NO_STARTING_DELIMITER (148)	No hay un delimitador de inicio en el patrón.
DSM_RC_NEEDED_DIR_DELIMITER (149)	Se necesita un delimitador de directorio inmediatamente antes y después de la serie de metadatos de "directorios coincidentes" ("...") y no se ha encontrado ninguno.
DSM_RC_NO_PASS_FILE (168)	El archivo de contraseña no está disponible.
DSM_RC_UNMATCHED_QUOTE (177)	En la cadena de opción hay una comilla sin correlacionar.
DSM_RC_NLS_CANT_OPEN_TXT (0610)	No se puede abrir el archivo de texto del mensaje.
DSM_RC_INVALID_OPT (400)	Una entrada en la cadena de opción no es válida.
DSM_RC_INVALID_DS_HANDLE (2014)	Manejador DSM no válido.
DSM_RC_NO_OWNER_REQD (2032)	El parámetro del propietario debe ser NULO cuando *passwordaccess* está configurado en *generate*.
DSM_RC_NO_NODE_REQD (2033)	El parámetro del nodo debe ser NULO cuando *passwordaccess* está configurado en *generate*.
DSM_RC_WRONG_VERSION (2064)	La versión de la interfaz de programación de aplicaciones (API) de la aplicación tiene un valor más alto que la versión IBM Spectrum Protect.
DSM_RC_PASSWD_TOOLONG (2103)	La contraseña especificada es demasiado larga.
DSM_RC_NO_OPT_FILE (2220)	No se ha encontrado el archivo de configuración.
DSM_RC_INVALID_KEYWORD (2221)	Una palabra clave especificada en una cadena de opciones no es válida.
DSM_RC_PATTERN_TOO_COMPLEX (2222)	El patrón de inclusión/exclusión es demasiado complicado para que lo interprete IBM Spectrum Protect.
DSM_RC_NO_CLOSING_BRACKET (2223)	No hay paréntesis de cierre en el patrón.
DSM_RC_INVALID_SERVER (2225)	En un entorno de varios usuarios, no se ha encontrado el servidor en el archivo de configuración del sistema.
DSM_RC_NO_HOST_ADDR (2226)	No hay información suficiente para conectar con el host.
DSM_RC_MACHINE_SAME (2227)	El nombre de nodo definido en el archivo de opciones no puede ser el mismo que el nombre del host del sistema.
DSM_RC_NO_API_CONFIGFILE (2228)	No puede abrir el archivo de configuración.
DSM_RC_NO_INCLEXCL_FILE (2229)	No se ha encontrado el archivo de inclusión/exclusión.
DSM_RC_NO_SYS_OR_INCLEXCL (2230)	No se ha encontrado el archivo dsm.sys o el de inclusión/exclusión.

Conceptos relacionados:

- Visión general del archivo de opciones del cliente
- Opciones de proceso

dsmInitEx

La llamada de función **dsmInitEx** inicia una sesión de la API utilizando los parámetros adicionales para una verificación extendida.

Sintaxis

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

Parámetros

dsUint32_t *dsmHandleP (O)
El manejador que identifica la sesión de iniciación y que la asocia con llamadas subsiguientes IBM Spectrum Protect.

dsmInitExIn_t *dsmInitExInP
Esta estructura contiene los siguientes parámetros de entrada:

dsmApiVersion *dsmApiVersionP (I)
Este parámetro es un puntero a la estructura de datos que identifica la versión de la API que el cliente de la aplicación está utilizando en esta sesión. La estructura contiene los valores de las cuatro constantes, DSM_API_VERSION, DSM_API_RELEASE, DSM_API_LEVEL y DSM_API_SUBLEVEL que se definen en el archivo `dsmapitd.h`. Invoque **dsmQueryApiVersionEx** y compruebe que la versión de la API del cliente de aplicación y la versión de la biblioteca de la API instalada en la estación de trabajo del usuario sean compatibles.

char *clientNodeNameP (I)
Este parámetro es un puntero al nodo de la sesión IBM Spectrum Protect. Todas las sesiones deben estar asociadas con un nombre de nodo. La constante DSM_MAX_NODE_LENGTH del archivo `dsmapitd.h` establece el tamaño máximo de un nombre de nodo.

El nombre del nodo no distingue entre mayúsculas y minúsculas.

Si este parámetro está configurado en NULO, y **passwordaccess** está configurado en `prompt`, la API intenta obtener primero el nombre del nodo de la cadena de opciones que se pasaron. Si no se encuentra allí, entonces la API intenta obtener el nombre del nodo del archivo de configuración o de los archivos de opciones. Si este intento de obtener el nombre del nodo falla, la API de UNIX o Linux utiliza el nombre del host del sistema, mientras que las API de otros sistemas operativos devuelven `DSM_RC_REJECT_ID_UNKNOWN`.

Este parámetro debe ser NULL si la opción **passwordaccess** del archivo `dsm.sys` se define como `generate`. La API entonces utiliza el nombre del host del sistema.

char *clientOwnerNameP (I)
Este parámetro es un puntero al propietario de la sesión IBM Spectrum Protect. Si el sistema operativo es una plataforma de varios usuarios, un nombre de propietario de NULO (el usuario root) tiene la autorización de realizar una copia de seguridad, archivar, restaurar o recuperar los objetos que pertenecen a la aplicación, independientemente del propietario del objeto.

El nombre del propietario es sensible a mayúsculas y minúsculas.

Este parámetro debe ser NULL si la opción **passwordaccess** del archivo dsm.sys se define como generate. La API entonces utiliza el ID del usuario del inicio de sesión.

Consejo: En una plataforma de varios usuarios, si **passwordaccess** está configurado en prompt, no es necesario que el nombre del propietario se correlacione con el ID del usuario activo de la sesión que ejecuta la aplicación.

char *clientPasswordP (I)
Un puntero a la contraseña del nodo en donde se ejecuta la sesión IBM Spectrum Protect. La constanteDSM_MAX_VERIFIER_LENGTH del archivo dsmapitd.h define el tamaño máximo permitido para una contraseña.

La contraseña no es sensible a mayúsculas y minúsculas.

Excepto cuando se inicia primero el archivo de contraseña, el valor de este parámetro se ignora si **passwordaccess** está configurado en generate.

char *userNameP;
Un puntero al nombre de usuario administrativo que tiene la autorización del cliente para este nodo.

char *userPasswordP;
Un puntero a la contraseña del parámetro **userName**, si se proporciona un valor.

char *applicationType (I)
Identifica la aplicación que está ejecutando la sesión IBM Spectrum Protect. El cliente de la aplicación identifica el valor.

Cada vez que un cliente de aplicación de la API inicia una sesión con el servidor, el tipo de aplicación (o sistema operativo) del cliente se actualiza en el servidor. El valor se especifica en el campo **platform** del servidor. Considere la posibilidad de utilizar un ID de sistema operativo en el valor. La longitud máxima de la cadena se define en la constante DSM_MAX_PLATFORM_LENGTH.

Para ver el valor actual del tipo de la aplicación, llame a **dsmQuerySessInfo**.

char *configfile (I)
Apunta a una cadena de caracteres que contiene el nombre completo del archivo de configuración de la API. Las opciones que se especifican en el archivo de configuración de la API modifican sus especificaciones en el archivo de opciones del cliente. Los archivos de opciones se definen cuando se instala IBM Spectrum Protect (cliente o API).

char *options (I)
Apunta a una cadena de caracteres que puede contener opciones de archivo como:
- Compressalways
- Servername (Solo sistemas UNIX y Linux)
- TCPServeraddr (no para sistemas UNIX)
- Fromnode
- Fromowner

El cliente de la aplicación puede utilizar la lista de opciones para modificar los valores de estas opciones definidas por el archivo de configuración.

Las opciones tienen el formato siguiente:
1. Cada opción que se especifica en la lista de opción empieza con un guión (-) y lo sigue la palabra clave de la opción.

2. A la palabra clave le sigue un signo de igual (=) y a continuación el parámetro de opción.
3. Si el parámetro de la opción contiene un espacio en blanco, cierre el parámetro con comillas simples o dobles.
4. Si se especifica más de una opción, separe las opciones con espacios en blanco.

Si las opciones son NULO, los valores de todas las opciones se toman del archivo de opciones del usuario o del archivo de configuración de la API.

dirDelimiter
El delimitador del directorio se prefija en los nombres del espacio de archivo, alto y bajo rango. Debe especificar el parámetro **dirDelimiter** sólo si la aplicación altera temporalmente los valores predeterminados del sistema. En un entorno UNIX o Linux, el valor predeterminado es la barra inclinada (/). En un entorno de Windows, el valor predeterminado es la barra invertida (\).

useUnicode
Un indicador booleano que indica si está habilitado Unicode. El distintivo **useUnicode** debe establecerse en `false` para la interoperatividad entre plataformas de sistemas UNIX y Windows.

bCrossPlatform
Un distintivo booleano que debe establecerse (`bTrue`) para la interoperatividad entre plataformas de sistemas UNIX y Windows. Cuando está establecido el distintivo `bCrossPlatform`, la API se asegura de que los espacios de archivo no sean Unicode y que la aplicación no utilice Unicode. Una aplicación Windows que utiliza Unicode no es compatible con las aplicaciones que utilizan codificaciones no de Unicode. El distintivo `bCrossPlatform` no se debe establecer para una aplicación Windows que utiliza Unicode.

UseTsmBuffers
Indica si se ha de utilizar la supresión de copia de almacenamiento intermedio.

numTsmBuffers
El número de almacenamientos intermedios cuando `useTsmBuffers=bTrue`.

bEncryptKeyEnabled
Indica si se utiliza el cifrado con una clave gestionada por la aplicación.

encryptionPasswordP
La contraseña del cifrado.

Restricción: Cuando `encryptkey=save`, si existe una clave cifrada, se omite el valor especificado en **encryptionPasswordP**.

dsmAppVersion *appVersionP (I)
Este parámetro apunta a la estructura de datos que identifica la información de la versión de la aplicación que comienza una sesión de API. La estructura contiene los valores de las cuatro constantes, applicationVersion, applicationRelease, applicationLevel y applicationSubLevel, que se definen en el archivo `tsmapitd.h`.

dsmInitExOut_t *dsmInitExOut P
Esta estructura contienen los parámetros de salida.

dsUint32_t *dsmHandle (0)
El manejador que identifica esta sesión de inicialización y que la asocia con llamadas API subsiguientes.

infoRC
Información adicional acerca del código de retorno. Compruebe el código de retorno de la función y el valor de **infoRC**. Si el valor de **infoRC** es DSM_RC_REJECT_LASTSESS_CANCELED (69), the IBM Spectrum Protect indica que el administrador ha cancelado la última sesión.

Códigos de retorno

Los números de código de retorno se proporcionan entre paréntesis ( ).

Tabla 39. Códigos de retorno de **dsmInitEx**

Código de retorno	Explicación
DSM_RC_ABORT_SYSTEM_ERROR (1)	El servidor de IBM Spectrum Protect ha detectado un error del sistema y ha notificado a los clientes.
DSM_RC_REJECT_VERIFIER_EXPIRED (52)	La contraseña ha caducado y debe actualizarse. La siguiente llamada debe ser **dsmChangePW** con el manejador devuelto en esta llamada.
DSM_RC_REJECT_ID_UNKNOWN (53)	No se ha encontrado el nombre de nodo.
DSM_RC_TA_COMM_DOWN (103)	El vínculo de comunicaciones no funciona.
DSM_RC_AUTH_FAILURE (137)	Ha ocurrido un fallo de autenticación.
DSM_RC_NO_STARTING_DELIMITER (148)	No hay un delimitador de inicio en el patrón.
DSM_RC_NEEDED_DIR_DELIMITER (149)	Se necesita un delimitador de directorio inmediatamente antes y después de "match directories" meta-string ("..."), pero no se ha encontrado.
DSM_RC_NO_PASS_FILE (168)	El archivo de contraseña no está disponible.
DSM_RC_UNMATCHED_QUOTE (177)	En la cadena de opción hay una comilla no coincidente.
DSM_RC_NLS_CANT_OPEN_TXT (0610)	No se puede abrir el archivo de texto del mensaje.
DSM_RC_INVALID_OPT (2013)	Una entrada en la cadena de opción no es válida.
DSM_RC_INVALID_DS_HANDLE (2014)	Manejador DSM no válido.
DSM_RC_NO_OWNER_REQD (2032)	El parámetro del propietario debe ser NULO cuando **passwordaccess** está configurado en `generate`.
DSM_RC_NO_NODE_REQD (2033)	El parámetro del nodo debe ser NULL cuando se establece **passwordaccess** en generate.
DSM_RC_WRONG_VERSION (2064)	La versión de la interfaz de programación de aplicaciones (API) de la aplicación tiene un valor más alto que la versión de IBM Spectrum Protect.
DSM_RC_PASSWD_TOOLONG (2103)	La contraseña especificada es demasiado larga.
DSM_RC_NO_OPT_FILE (2220)	No se ha encontrado un archivo de configuración.
DSM_RC_INVALID_KEYWORD (2221)	Una palabra clave especificada en una cadena de opciones no es válida.
DSM_RC_PATTERN_TOO_COMPLEX (2222)	El patrón de inclusión/exclusión es demasiado complicado para que IBM Spectrum Protect lo interprete.
DSM_RC_NO_CLOSING_BRACKET (2223)	No hay paréntesis de cierre en el patrón.
DSM_RC_INVALID_SERVER (2225)	En un entorno de varios usuarios, no se ha encontrado el servidor en el archivo de configuración del sistema.
DSM_RC_NO_HOST_ADDR (2226)	No hay información suficiente para conectar con el host.

Tabla 39. Códigos de retorno de **dsmInitEx** *(continuación)*

Código de retorno	Explicación
DSM_RC_MACHINE_SAME (2227)	El nombre de nodo definido en el archivo de opciones no puede ser el mismo que el nombre del host del sistema.
DSM_RC_NO_API_CONFIGFILE (2228)	No puede abrir el archivo de configuración.
DSM_RC_NO_INCLEXCL_FILE (2229)	No se ha encontrado el archivo de inclusión/exclusión.
DSM_RC_NO_SYS_OR_INCLEXCL (2230)	Bien `dsm.sys` o el archivo include-exclude no se han encontrado.

Conceptos relacionados:

- Visión general del archivo de opciones del cliente
- Opciones de proceso

dsmLogEvent

La llamada de función **dsmLogEvent** registra un mensaje de usuario (ANE4991 I) en el archivo de registro del servidor, en el registro de error local o en ambos. Una estructura de tipo **logInfo** se pasa en la llamada. Esta llamada debe realizarse con el estado **InSession** dentro de una sesión. No debe ejecutarse durante una operación de envío, obtención o consulta. Para recuperar los mensajes registrados en el servidor, utilice el mandato **query actlog** mediante el cliente administrativo.

Consulte el diagrama de estado resumido, Figura 20 en la página 79.

Sintaxis

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

Parámetros

dsUint32_t dsmHandle(I)
El manejador que asocia esta llamada con una llamada **dsmInitEx** anterior.

logInfo *logInfoP (I)
Pasa el mensaje y el destino. El cliente de aplicación es responsable de asignar almacenamiento para la estructura.

Los campos en la estructura **logInfo** son:

mensaje
El texto del mensaje que se va a registrar. Esta debe ser una cadena finalizada en nulo. La longitud máxima es DSM_MAX_RC_MSG_LENGTH.

dsmLogtype
Especifica dónde registrar el mensaje. Como valores posibles se incluyen: **logServer**, **logLocal**, **logBoth**.

Códigos de retorno

Los números de código de retorno se proporcionan entre paréntesis).

Tabla 40. Códigos de retorno para dsmLogEvent

Código de retorno	Explicación
DSM_RC_STRING_TOO_LONG (2120)	La cadena del mensaje es demasiado larga.

dsmLogEventEx

La llamada a función **dsmLogEventEx** registra un mensaje en el archivo de registro del servidor, en el registro de errores local o en ambos. Esta llamada debe realizarse con el estado **InSession** dentro de una sesión. La llamada no se puede realizar dentro de una llamada de consulta, envío u obtención.

Diagrama de estado de resumen: Para obtener una descripción general de las interacciones de sesión, consulte el diagrama de estado de resumen en el siguiente tema:

Figura 20 en la página 79

La gravedad determina el número de mensaje IBM Spectrum Protect. Para ver los mensajes que están registrados en el servidor, utilice el mandato **query actlog** a través del cliente administrativo. Utilice la opción de cliente de IBM Spectrum Protect, `errorlogretention`, para borrar el archivo de registro de errores del cliente si la aplicación genera numerosos mensajes de cliente escritos en el registro de cliente, `dsmLogType`, `logLocal` o `logBoth`. Para obtener más información, consulte la documentación del servidor de IBM Spectrum Protect.

Sintaxis

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

Parámetros

dsUint32_t dsmHandle(I)
El manejador que asocia esta llamada con una llamada **dsmInitEx** anterior.

dsmLogExIn_t *dsmLogExInP
Esta estructura contiene los parámetros de entrada.

dsmLogSeverity severity;
Este parámetro es la gravedad del evento. Los valores posibles son:

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

char appMsgID[8];
Este parámetro es una cadena que sirve para identificar el mensaje de aplicación específico. Un formato adecuado tiene tres caracteres seguidos por cuatro números, por ejemplo: `DSM0250`.

dsmLogType logType;
Este parámetro especifica dónde dirigir el evento. El parámetro tiene los siguientes valores posibles:

- logServer
- logLocal
- logBoth

char *message;
Este parámetro es el texto del mensaje de evento que se registra. El texto debe ser una serie finalizada en nulo. La longitud máxima es DSM_MAX_RC_MSG_LENGTH.

Restricción: Los mensajes que van al servidor deben estar en inglés. Los mensajes que no son en inglés no se muestran correctamente.

dsmLogExOut_t *dsmLogExOutP
Esta estructura contienen los parámetros de salida. Actualmente no existen parámetros de salida.

Códigos de retorno

Los números de código de retorno se proporcionan entre paréntesis ( ).

Tabla 41. Códigos de retorno para dsmLogEventEx

Código de retorno	Explicación
DSM_RC_STRING_TOO_LONG (2120)	La cadena del mensaje es demasiado larga.

dsmQueryAccess

La llamada de función **dsmQueryAccess** consulta las reglas de autorización de acceso del servidor en las copias de seguridad de los objetos o en las copias archivadas. Un puntero a una agrupación de reglas de acceso pasa a la llamada, y se devuelve la agrupación rellenada. Un puntero al número de reglas pasa para indicar cuántas reglas hay en la agrupación.

No existen códigos de retorno específicos de esta llamada.

Sintaxis

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

Parámetros

dsUint32_t dsmHandle (I)
El manejador que asocia esta llamada con una llamada **dsmInitEx** anterior.

qryRespAccessData **accessListP (O)
Un puntero a una agrupación de elementos qryRespAccessData que asigna la biblioteca API. Cada elemento se corresponde con una regla de acceso. El número de elementos de la agrupación se devuelve en el parámetro **numberOfRules** . La información que se devuelve en cada elemento qryRespAccessData incluye los siguientes datos:

Nombre	Descripción
ruleNumber	El ID para la regla de acceso. Esto identifica la regla que se ha de suprimir.
AccessType	El tipo de archivado o copia de seguridad.
Nodo	El nodo al que se da acceso.
Propietario	El usuario al que se le da acceso.
objName	Descriptores de espacio de archivo de alto y bajo rango.

dsUint32_t *numberOfRules (0)
Devuelve el número de reglas en la agrupación accessList.

dsmQueryApiVersion

La llamada de función **dsmQueryApiVersion** realiza una solicitud de consulta de la versión de la biblioteca de la API a la que accede el cliente de aplicación.

Todas las actualizaciones de la API se realizan en un formato compatible con versiones posteriores. Cualquier cliente de la aplicación con una versión API o release inferior o igual a la biblioteca API en la estación de trabajo del usuario final opera sin cambios. Tenga en cuenta que en caso de que la llamada **dsmQueryApiVersion** devuelva una versión o release de la versión más antigua que la del cliente de aplicación algunas llamadas API pueden mejorar de una forma que no es compatible por la versión de la API más antigua del usuario.

El número de la versión API de la aplicación se almacenada en el archivo de encabezado dsmapitd.h como constantes DSM_API_VERSION, DSM_API_RELEASE, y DSM_API_LEVEL.

No existen códigos de retorno específicos de esta llamada.

Sintaxis

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

Parámetros

dsmApiVersion *apiVersionP (0)
Este parámetro es un puntero a la estructura que contiene la versión de la biblioteca API, release y los componentes de nivel. Por ejemplo, si la versión de la biblioteca es 1.1.0, entonces, después de volver de la llamada, los campos de la estructura contienen los siguientes valores:

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

dsmQueryApiVersionEx

La llamada de función **dsmQueryApiVersionEx** realiza una solicitud de consulta de la versión de la biblioteca API a la que accede el cliente de la aplicación.

Todas las actualizaciones al API se llevan a cabo en un formato compatible con la versión superior. Cualquier cliente de la aplicación con una versión API o release inferior o igual a la biblioteca API en la estación de trabajo del usuario final opera sin cambios. Consulte el resumen de los cambios de código en el archivo README_api_enu para obtener información sobre las excepciones de compatibilidad superior. Si la llamada **dsmQueryApiVersionEx** devuelve una versión o release de una versión distinta a la del cliente de la aplicación, tenga en cuenta

antes de proceder que algunas llamadas API pueden mejorar de una forma que no es soportada por la versión antigua de la API del usuario final.

El número de versión de API de la aplicación se almacena en el archivo de encabezado dsmapitd.h como constantes DSM_API_VERSION, DSM_API_RELEASE, DSM_API_LEVEL, y DSM_API_SUBLEVEL.

No existen códigos de retorno específicos de esta llamada.

Sintaxis

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

Parámetros

dsmApiVersionEx *apiVersionP (O)
Este parámetro es un puntero a la estructura que contiene la versión, release, nivel y subnivel de los componentes de la biblioteca API. Por ejemplo, si la versión de la biblioteca es 5.5.0.0, entonces, después de volver de la llamada, los campos de la estructura contienen los siguientes valores:

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

dsmQueryCliOptions

La llamada de función **dsmQueryCliOptions** consulta valores de opción importantes en los archivos de opción del usuario. Una estructura de tipo **optStruct** se pasa en la llamada y contiene la información. Esta llamada se realiza antes de invocar **dsmInitEx**, y determina la configuración antes de la sesión.

No existen códigos de retorno específicos a esta llamada.

Sintaxis

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

Parámetros

optStruct *optstructP (I/O)
Este parámetro pasa la dirección de la estructura que la API finaliza. El cliente de la aplicación es responsable de asignar almacenamiento para la estructura. Al devolverse correctamente, se introduce la información apropiada en los campos de la estructura.

La siguiente información se devuelve en la estructura **optStruct** :

Nombre	Descripción
dsmiDir	El valor de la variable de entorno DSMI_DIR.
dsmiConfig	El archivo de opciones de cliente, según se especifica en la variable de entorno DSMI_CONFIG.
serverName	El nombre del servidor IBM Spectrum Protect.
commMethod	El método de comunicación seleccionado. Consulte #defines para DSM_COMM_* en el archivo.
serverAddress	La dirección del servidor que se basa en el método de comunicación.
nodeName	El nombre de nodo de cliente (máquina).

Nombre	Descripción
compression	Este campo proporciona información con respecto a la opción de compresión.
passwordAccess	Los valores son: *bTrue* para generate, y *bFalse* para prompt.

Conceptos relacionados:

Opciones de proceso

dsmQuerySessInfo

La llamada de función **dsmQuerySessInfo** inicia una solicitud de consulta a IBM Spectrum Protect sobre información relacionada con la operación de la sesión especificada en **dsmHandle**. Una estructura de tipo **ApiSessInfo** se pasa en la llamada, con toda la información relacionada con la sesión disponible. Esta llamada se inicia después de una llamada **dsmInitEx** correcta.

La información que se devuelve en la estructura **ApiSessInfo** incluye lo siguiente:
- Información del servidor: número de puerto, fecha y hora, y tipo
- Valores predeterminados del cliente: tipo de aplicación, permisos de supresión, delimitadores y límites de transacción
- Información de la sesión: ID de inicio de sesión y propietario
- Datos de política: dominio, conjunto de políticas activas y periodo de gracia de retención

Consulte Apéndice B, "Archivos de origen de definiciones de tipo de API", en la página 165 para obtener más información sobre el contenido de la estructura que se pasa y de cada campo de la misma.

Sintaxis

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

Parámetros

dsUint32_t dsmHandle (I)
El manejador que asocia esta llamada con una llamada **dsmInitEx** anterior.

ApiSessInfo *SessInfoP (I/O)
Este parámetro pasa la dirección de la estructura que la API introduce. El cliente de la aplicación es responsable de asignar el almacenamiento para la estructura y de completar las entradas de los campos que indican la versión de la estructura que se utiliza. Cuando la operación es correcta, los campos de la estructura se rellenan con la información adecuada. **adsmServerName** es el nombre que se proporciona en el mandato **define server** del servidor IBM Spectrum Protect. Si el valor del campo archiveRetentionProtection es true, el servidor está habilitado para la protección de retención.

Códigos de retorno

Los números de código de retorno se proporcionan entre paréntesis).

Tabla 42. Códigos de retorno para dsmQuerySessInfo

Código de retorno	Explicación
DSM_RC_NO_SESS_BLK (2006)	Información de sesión de servidor no bloqueada.
DSM_RC_NO_POLICY_BLK (2007)	La información de la política del servidor no está disponible.

Tabla 42. Códigos de retorno para dsmQuerySessInfo (continuación)

Código de retorno	Explicación
DSM_RC_WRONG_VERSION_PARM (2065)	La versión de la API del cliente de aplicación es distinta de la versión de la biblioteca de IBM Spectrum Protect.

dsmQuerySessOptions

La llamada de función **dsmQuerySessOptions** consulta valores de opción importantes que son válidos en la sesión especificada en **dsmHandle**. Una estructura de tipo optStruct se pasa en la llamada y contiene la información.

Esta llamada se inicia después de una llamada **dsmInitEx** correcta. Es posible que los valores que se devuelvan sean distintos de los valores devueltos en una llamada **dsmQueryCliOptions**, depende de los valores pasados a la llamada **dsmInitEx**, principalmente optString, y optFile. Para obtener más información sobre la opción de precedencia, consulte "Conceptos básicos de los archivos de opciones y configuración" en la página 1.

No existen códigos de retorno específicos de esta llamada.

Sintaxis

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

Parámetros

dsUint32_t dsmhandle(I)
El manejador que asocia esta llamada con una llamada **dsmInitEx** anterior.

optStruct *optstructP (I/O)
Este parámetro pasa la dirección de la estructura que la API finaliza. El cliente de aplicación es responsable de asignar almacenamiento para la estructura. Cuando la operación es correcta, los campos de la estructura se rellenan con la información adecuada.

La información que se devuelve en la estructura optStruct es:

Nombre	Descripción
dsmiDir	El valor de la variable de entorno DSMI_DIR.
dsmiConfig	El archivo dsm.opt que especifica la variable de entorno DSMI_CONFIG.
serverName	El nombre de la sentencia del servidor IBM Spectrum Protect en el archivo de opciones.
commMethod	El método de comunicación que fue seleccionado. Consulte #defines para DSM_COMM_* en el archivo dsmapitd.h.
serverAddress	La dirección del servidor basada en el método de comunicación.
nodeName	El nombre del nodo del cliente (máquina).
compression	El valor de la opción de compresión (bTrue=on y bFalse=off).
compressAlways	El valor de la opción compressalways (bTrue=on y bFalse=off).
passwordAccess	Valor bTrue para generar y bFalse para solicitud.

Conceptos relacionados:

Opciones de proceso

dsmRCMsg

La llamada de función **dsmRCMsg** obtiene el texto del mensaje asociado con el código de retorno de la API.

El parámetro **msg** muestra el código de retorno del prefijo del mensaje entre paréntesis ( ), seguido del texto del mensaje. Por ejemplo, una llamada a **dsmRCMsg** puede devolver lo siguiente:

```
ANS0264E (RC2300) Sólo puede ejecutar el usuario raíz dsmChangePW o dsmDeleteFS.
```

Para algunos idiomas cuyos caracteres son diferentes en ANSII y en páginas de código OEM, es posible que sea necesario convertir cadenas de ANSII a OEM antes de imprimirlas (por ejemplo, en los conjuntos de caracteres de bytes únicos de Europa del este). A continuación se muestra un ejemplo:

```
dsmRCMsg(dsmHangle, rc, msgBuf);
#ifdef WIN32
#ifndef WIN64
CharToOemBuff(msgBuf, msgBuf, strlen(msgBuf));
#endif
#endif
printf("
```

Sintaxis

```
dsInt16_t dsmRCMsg  (dsUint32_t       dsmHandle,
   dsInt16_t        dsmRC,
   char        *msg);
```

Parámetros

dsUint32_t dsmHandle (I)
El manejador que asocia esta llamada con una llamada **dsmInitEx** anterior.

dsInt16_t dsmRC (I)
El código de retorno de la API del texto del mensaje asociado. Los códigos de retorno de la API aparecen en una lista en el archivo dsmrc.h. Para obtener más información, consulte el apartado Apéndice A, "Archivo de origen de códigos de retorno de la API: dsmrc.h", en la página 155.

char *msg (O)
Este parámetro es el texto del mensaje que está asociado con el código de retorno, **dsmRC**. El llamador es responsable de asignar espacio suficiente al texto del mensaje.

La longitud máxima para **msg** está definida como DSM_MAX_RC_MSG_LENGTH.

En plataformas que tienen NLS y la opción de idiomas en archivos de mensaje, la API devuelve una cadena de mensaje en el idioma nacional.

Códigos de retorno

Los números de código de retorno se proporcionan entre paréntesis).

Tabla 43. Códigos de retorno para dsmRCMsg

Código de retorno	Explicación
DSM_RC_NULL_MSG (2002)	El parámetro **msg** para la llamada dsmRCMsg es un puntero NULO.
DSM_RC_INVALID_RETCODE (2021)	El código de retorno que pasó a la llamada **dsmRCMsg** no es un código válido.

Tabla 43. Códigos de retorno para dsmRCMsg (continuación)

Código de retorno	Explicación
DSM_RC_NLS_CANT_OPEN_TXT (0610)	No se puede abrir el archivo de texto del mensaje.

dsmRegisterFS

La llamada de función **dsmRegisterFS** registra un espacio de archivo nuevo con el servidor IBM Spectrum Protect. Registre un espacio de archivo primero antes de realizar ninguna copia de seguridad de éste.

Los clientes de aplicación no deben utilizar los mismos nombres de espacio de archivo que utilizaría un cliente de copia de seguridad/archivado.

- En UNIX o Linux, ejecute el mandato **df** para estos nombres.
- En Windows, estos nombres son por lo general las etiquetas de volumen que están asociadas con las distintas unidad en el sistema.

Sintaxis

```
dsInt16_t dsmRegisterFS  (dsUint32_t        dsmHandle,
   regFSData     *regFilespaceP);
```

Parámetros

dsUint32_t dsmHandle (I)
El manejador que asocia esta llamada con una llamada **dsmInitEx** anterior.

regFSData *regFilespaceP (I)
Este parámetro pasa el nombre del espacio de archivo y asocia la información que necesita registrar con el servidor IBM Spectrum Protect.

Consejo: El campo *fstype* incluye el prefijo **"API:".** Todas las consultas del espacio de archivo muestran esta cadena. Por ejemplo, si el usuario pasa *myfstype* para *fstype* en **dsmRegisterFS**, se devuelve la cadena de valor real del servidor como `API:myfstype` cuando se realiza una consulta. Este prefijo distingue objetos API de copias de seguridad o archivados de objeto.

El área que se puede utilizar para **fsInfo** es ahora DSM_MAX_USER_FSINFO_LENGTH.

Códigos de retorno

Los números de código de retorno se proporcionan entre paréntesis ( ).

Tabla 44. Códigos de retorno para dsmRegisterFS

Código de retorno	Explicación
DSM_RC_INVALID_FSNAME (2016)	Nombre de espacio de archivo no válido.
DSM_RC_INVALID_DRIVE_CHAR (2026)	La letra de la unidad no es un carácter alfabético.
DSM_RC_NULL_FSNAME (2027)	Nombre de espacio de archivo nulo.
DSM_RC_FS_ALREADY_REGED (2062)	El espacio de archivo ya está registrado.
DSM_RC_WRONG_VERSION_PARM (2065)	La versión de la API del cliente de aplicación es distinta de la versión de la biblioteca de IBM Spectrum Protect.
DSM_RC_FSINFO_TOOLONG (2106)	La información del espacio de archivo es demasiado larga.

dsmReleaseBuffer

La función **dsmReleaseBuffer** devuelve un almacenamiento intermedio a IBM Spectrum Protect. La aplicación llama a **dsmReleaseBuffer** después de que se haya llamado un **dsmGetDataEx** y que la aplicación haya movido todos los datos fuera del almacenamiento intermedio y esté listo para liberarlo. **dsmReleaseBuffer** requiere que **dsmInitEx** se llame con ***UseTsmBuffers*** configurada en *btrue* y que se haya proporcionado un valor que no sea cero en *numTsmBuffers*. También se debe invocar **dsmReleaseBuffer** si la aplicación está a punto de invocar **dsmTerminate** y todavía retiene almacenamientos intermedios de datos.

dsmReleaseBufferSyntax

```
dsInt16_t  dsmReleaseBuffer  (releaseBufferIn_t     *dsmReleaseBufferInP,
                              releaseBufferOut_t    *dsmReleaseBufferOutP) ;
```

Parámetros

releaseBufferIn_t * dsmReleaseBufferInP (I)
Esta estructura contiene los siguientes parámetros de entrada.

dsUint32_t dsmHandle (I)
El manejador que asocia esta llamada con una llamada **dsmInitEx** previa.

dsUint8_t tsmBufferHandle(I)
El manejador que identifica este almacenamiento intermedio.

char *dataPtr(I)
La dirección a la que se graba la aplicación.

Códigos de retorno

Los números de código de retorno se proporcionan entre paréntesis).

Tabla 45. Códigos de retorno para dsmReleaseBuffer

Código de retorno	Explicación
DSM_RC_BAD_CALL_SEQUENCE	La llamada no se ha emitido en el estado adecuado.
DSM_RC_INVALID_TSMBUFFER	El manejador o el valor de **dataPtr** no son válidos.
DSM_RC_BUFF_ARRAY_ERROR	Se ha producido un error de agrupación de almacenamiento intermedio.

dsmRenameObj

La llamada de función **dsmRenameObj** renombra el alto y bajo rango del objeto. En copias de seguridad de objetos, pasa el nombre de objeto actual y cambia el nombre de alto rango o el de bajo rango. En objetos archivados, pasa el nombre del espacio de archivo y el ID del objeto y cambia el nombre de alto rango o el de bajo rango. Utilice esta llamada de función dentro de las llamadas **dsmBeginTxn** y **dsmEndTxn**.

El indicador de fusión determina si el nombre de una copia de seguridad de un objeto duplicada se fusiona con las copias de seguridad existentes. Si el nombre nuevo se corresponde con un objeto existente y la fusión ocurre, el objeto actual se transforma con el nombre nuevo y pasa a ser la versión activa del nuevo nombre mientras que el objeto activa que tenía dicho nombre se transforma en la copia más

inactiva del objeto. Si el nombre nuevo se corresponde con un objeto existente y no ocurre la fusión, la función devuelve el código de retorno DSM_RC_ABORT_DUPLICATE_OBJECT.

Restricción: Sólo el propietario del objeto puede cambiar el nombre.

La llamada de función **dsmRenameObj** prueba estas condiciones de fusión:

- El objeto actual **dsmObjName** y el nuevo objeto de nivel superior o inferior deben coincidir con el propietario, grupo de copia y clase de gestión.
- Debe haberes realizado una copia de seguridad más reciente del **dsmObjName** actual que del objeto activo con el mismo nombre.
- Sólo debe existir una copia activa del **dsmObjName** actual sin copias inactivas.

Sintaxis

```
dsInt16_t dsmRenameObj  (dsmRenameIn_t     *dsmRenameInP,
                         dsmRenameOut_t    *dsmRenameOutP);
```

Parámetros

dsUint32_t dsmHandle (I)
El manejador que asocia esta llamada con una llamada **dsmInitEx** anterior.

dsmRenameIn_t *dsmRenameInP
Esta estructura contiene los parámetros de entrada.

dsUint8_t repository (I);
Este parámetro indica si el espacio de archivo que se va a eliminar se encuentra en el repositorio de copia de seguridad o en el repositorio de archivado.

dsmObjName *objNameP (I);
Este parámetro es un puntero a la estructura que contiene el nombre del espacio del archivo, el nombre de nivel superior, el nombre de nivel inferior y el tipo de objeto.

char newHl [DSM_MAX_HL_LENGTH + 1];
Este parámetro especifica el nombre nuevo de nivel superior.

char newLl [DSM_MAX_LL_LENGTH + 1];
Este parámetro especifica el nombre nuevo de nivel inferior.

dsBool_t merge;
Este parámetro determina si una copia de seguridad se fusiona con objetos nombrados duplicados. Los valores son verdadero o falso.

ObjID;
El ID de objeto de los objetos archivados.

dsmRenameOut_t *dsmRnameOutP
Esta estructura contienen los parámetros de salida.

Nota: No hay parámetros de salida.

Códigos de retorno

Los números de código de retorno se proporcionan entre paréntesis).

Tabla 46. Códigos de retorno de dsmRenameObj

Código de retorno	Explicación
DSM_RC_ABORT_MERGE_ERROR (45)	El servidor ha detectado un error de fusión.
DSM_RC_ABORT_DUPLICATE_OBJECT (32)	El objeto ya existe y la fusión es falsa.
DSM_RC_ABORT_NO_MATCH (2)	No se ha encontrado el objeto.
DSM_RC_REJECT_SERVER_DOWNLEVEL (58)	El servidor IBM Spectrum Protect debe ser de la versión 3.7.4.0 o posterior para que funcione esta función.

dsmRequestBuffer

La función **dsmRequestBuffer** devuelve un almacenamiento intermedio a IBM Spectrum Protect. La aplicación invoca **dsmRequestBuffer** después de que se haya invocado **dsmGetDataEx** y de que la aplicación haya movido todos los datos fuera del almacenamiento intermedio y esté lista para liberarlo.

dsmReleaseBuffer requiere que se haya invocado **dsmInitEx** con la opción *UseTsmBuffers* configurada en *btrue* y que *numTsmBuffers* tenga un valor distinto a cero. También se debe invocar **dsmReleaseBuffer** si la aplicación está a punto de invocar **dsmTerminate** y todavía retiene almacenamientos intermedios de IBM Spectrum Protect.

Sintaxis

```
dsInt16_t  dsmRequestBuffer   (getBufferIn_t      *dsmRequestBufferInP,
                               getBufferOut_t     *dsmRequestBufferOutP) ;
```

Parámetros

getBufferIn_t * dsmRequestBufferInP (I)
Esta estructura contiene los siguientes parámetros de entrada:

dsUint32_t dsmHandle
El manejador que identifica la sesión y la asocia con una llamada **dsmInitEx** previa.

getBufferOut_t *dsmRequestBufferOut P (0)
Esta estructura contienen los parámetros de salida.

dsUint8_t tsmBufferHandle(0)
El manejador que identifica este almacenamiento intermedio.

char *dataPtr(0)
La dirección a la que se graba la aplicación.

dsUint32_t *bufferLen(0)
Número máximo de bytes que se pueden grabar a este almacenamiento intermedio.

Códigos de retorno

Los números de código de retorno se proporcionan entre paréntesis).

Tabla 47. Código de retorno de dsmRequestBuffer

Código de retorno	Explicación
DSM_RC_BAD_CALL_SEQUENCE (33)	La llamada no se ha emitido en el estado adecuado.
DSM_RC_SENDDATA_WITH_ZERO_SIZE (34)	Si el objeto que se está enviando es de longitud 0, no se permiten llamadas a **dsmReleaseBuffer**.
DSM_RC_BUFF_ARRAY_ERROR (121)	No se ha podido obtener un almacenamiento intermedio válido.

dsmRetentionEvent

La llamada de función **dsmRetentionEvent** envía una lista de ID de objetos al servidor de IBM Spectrum Protect con una operación de evento de retención para que se realice en estos objetos. Utilice esta llamada de función dentro de las llamadas **dsmBeginTxn** y **dsmEndTxn**.

Nota: El servidor debe ser de la versión 5.2.2.0 o posterior para que funcione esta función.

El número máximo de objetos en una llamada está limitado al valor de *maxObjPerTxn* que se devuelve en la estructura *ApisessInfo* desde una llamada **dsmQuerySessInfo**.

Solo un propietario de un objeto puede enviar un evento en dicho objeto.

Los siguientes eventos son posibles:

eventRetentionActivate
Se pueden emitir sólo en objetos que están vinculados a un evento basado en la clase de gestión. Al enviar este evento se activa el evento para este objeto y el estado de la retención de este objeto cambia de DSM_ARCH_RETINIT_PENDING a DSM_ARCH_RETINIT_STARTED.

eventHoldObj
Este evento emite una retención o una retención de supresión en el objeto, de modo que el objeto no caduque ni se suprima hasta que se libere.

eventReleaseObj
Este evento sólo se puede emitir en un objeto que tiene un valor de DSM_ARCH_HELD_TRUE en el campo ***objectHeld*** y elimina la retención en el objeto reanudando la política de retención original.

Antes de enviar ***dsmRetentionEvent***, envíe la secuencia de la consulta que se describe en "Consultas para el sistema IBM Spectrum Protect" en la página 34 con el fin de obtener la información del objeto. La llamada a **dsmGetNextQObj** devuelve una estructura de datos llamada **qryRespArchiveData** en consultas de archivado. Esta estructura de datos contiene la información necesaria de **dsmRetentionEvent**.

Sintaxis

```
extern dsInt16_t DSMLINKAGE dsmRetentionEvent(
  dsmRetentionEventIn_t          *ddsmRetentionEventInP,
  dsmRetentionEventOut_t         *dsmRetentionEventOutP
   );
```

Parámetros

dsmRetentionEventIn_t *dsmRetentionEventP
Esta estructura contiene los siguientes parámetros de entrada:

dsUint16_t stVersion;
Este parámetro indica la versión de la estructura.

dsUint32_t dsmHandle (I)
El manejador que asocia esta llamada con una llamada **dsmInitEx** anterior.

dsmEventType_t evenType (I);
Este parámetro indica el tipo de evento. Consulte el principio de esta sección para obtener información acerca del significado de los valores posibles: **eventRetentionActivate**, **eventHoldObj**, **eventReleaseObj**

dsmObjList_t objList;
Este parámetro indica una lista de ID de objetos para señalizar.

Códigos de retorno

Los números de código de retorno se proporcionan entre paréntesis ( ).

Tabla 48. Códigos de retorno de dsmRetentionEvent

Código de retorno	Explicación
DSM_RC_ABORT_NODE_NOT_AUTHORIZED (36)	El nodo o usuario no tiene una autorización adecuada.
DSM_RC_ABORT_TXN_LIMIT_EXCEEDED (249)	Hay demasiados objetos en la transacción.
DSM_RC_ABORT_OBJECT_ALREADY_HELD (250)	El objeto ya está retenido, no es posible emitir otra retención.
DSM_RC_REJECT_SERVER_DOWNLEVEL (58)	El servidor debe ser de la V5.2.2.0 o posterior para que funcione esta función.

dsmSendBufferData

La llamada de función **dsmSendBufferData** envía una corriente de bytes de datos a IBM Spectrum Protect a través de un almacenamiento intermedio proporcionado en una llamada **dsmReleaseBuffer** anterior. El cliente aplicación puede pasar cualquier tipo de datos para que se almacenen en el servidor. Normalmente estos datos son datos de archivos, pero no necesariamente. Puede llamar a **dsmSendBufferData** varias veces, si la corriente de bytes de datos que está enviando es demasiado larga. Independientemente de si la llamada falla o se procesa correctamente, se libera el almacenamiento intermedio.

Restricción: Al utilizar la opción *useTsmBuffers*, aunque se incluya un objeto para compresión, el objeto no se comprime.

Sintaxis

```
dsInt16_t  dsmSendBufferData   (sendBufferDataIn_t        *dsmSendBufferDataExInP,
                                sendBufferDataOut_t       *dsmSendBufferDataOutP) ;
```

Parámetros

sendBufferDataIn_t * dsmSendBufferDataInP (I)
Esta estructura contiene los siguientes parámetros de entrada.

dsUint32_t dsmHandle (I)
El manejador que asocia esta llamada con una llamada **dsmInitEx** anterior.

dsUint8_t tsmBufferHandle(I)
El manejador que identifica el almacenamiento intermedio que se va a enviar.

char *dataPtr(I)
La dirección en la que se graban los datos de la aplicación.

dsUint32_t numBytes(I)
El número real de bytes grabados por la aplicación (siempre debe ser menos que el valor proporcionando en **dsmReleaseBuffer**).

Códigos de retorno

Los números de código de retorno se proporcionan entre paréntesis).

Tabla 49. Códigos de retorno para dsmSendBufferData

Código de retorno	Explicación
DSM_RC_BAD_CALL_SEQUENCE (2041)	La llamada no se ha emitido en el estado adecuado.
DSM_RC_INVALID_TSMBUFFER (2042)	El manejador o el valor de **dataPtr** no son válidos.
DSM_RC_BUFF_ARRAY_ERROR (2045)	Se ha producido un error de agrupación de almacenamiento intermedio.
DSM_RC_TOO_MANY_BYTES (2043)	El valor de ***numBytes*** es mayor que el tamaño del almacenamiento intermedio proporcionado en la llamada **dsmReleaseBuffer**.

dsmSendData

La llamada de función **dsmSendData** envía una corriente de bytes de datos a IBM Spectrum Protect a través de un almacenamiento intermedio. El cliente aplicación puede pasar cualquier tipo de datos para que se almacenen en el servidor. Normalmente estos datos son datos de archivos, pero no necesariamente. Puede llamar a **dsmSendData** varias veces, si la corriente de bytes de datos que desea enviar es demasiado larga.

Restricción: El cliente de la aplicación no puede volver a utilizar el almacenamiento intermedio especificado en **dsmSendData** hasta que la llamada **dsmSendData** devuelva los datos.

Consejo: Si IBM Spectrum Protect devuelve el código 157 (DSM_RC_WILL_ABORT), inicie una llamada a **dsmEndSendObj** y a continuación a **dsmEndTxn** con un voto de DSM_VOTE_COMMIT. La aplicación recibe entonces el código de retorno 2302 (DSM_RC_CHECK_REASON_CODE) y vuelve a pasarlo al usuario de la aplicación. Esto informa al usuario acerca del motivo por el que el servidor está finalizando la transacción.

Sintaxis

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

Parámetros

dsUint32_t dsmHandle (I)
El manejador que asocia esta llamada con una llamada **dsmInitEx** anterior.

DataBlk *dataBlkPtr (I/O)
Este parámetro apunta a una estructura que incluye tanto un puntero al almacenamiento intermedio al que se le van a enviar los datos como el tamaño del almacenamiento intermedio. Cuando se devuelven los datos, esta

estructura contiene el número de bytes que se han transferido. Consulte Apéndice B, "Archivos de origen de definiciones de tipo de API", en la página 165 para obtener más información sobre la definición del tipo.

Códigos de retorno

Los números de código de retorno se proporcionan entre paréntesis).

Tabla 50. Códigos de retorno para dsmSendData

Código de retorno	Explicación
DSM_RC_NO_COMPRESS_MEMORY (154)	No hay memoria suficiente disponible para llevar a cabo una compresión o expansión de datos.
DSM_RC_COMPRESS_GREW (155)	Durante la compresión los datos comprimidos crecen en tamaño comparados con los datos originales.
DSM_RC_WILL_ABORT (157)	Ha ocurrido un error desconocido e inesperado, y la transacción se ha detenido.
DSM_RC_WRONG_VERSION_PARM (2065)	La versión de la API de la aplicación es distinta de la versión de la biblioteca de IBM Spectrum Protect.
DSM_RC_NEEDTO_ENDTXN (2070)	Necesita finalizar la transacción.
DSM_RC_OBJ_EXCLUDED (2080)	La lista de inclusión/exclusión excluye al objeto.
DSM_RC_OBJ_NOBCG (2081)	El objeto no tiene un grupo de copia de seguridad y no se enviará al servidor.
DSM_RC_OBJ_NOACG (2082)	El objeto no tiene un grupo de archivado y no se enviará al servidor.
DSM_RC_SENDDATA_WITH_ZERO_SIZE (2107)	El objeto no puede enviar datos con un valor *sizeEstimate* de cero bytes.

dsmSendObj

La llamada de función **dsmSendObj** inicia una solicitud para enviar un único objeto al almacenamiento. Se pueden realizar varias llamadas **dsmSendObj** y llamadas asociadas **dsmSendData** dentro de los límites de una transacción por motivos de rendimiento.

La llamada **dsmSendObj** procesa los datos del objeto como una corriente de datos de bytes en buffers de memoria. El parámetro **dataBlkPtr** en la llamada **dsmSendObj** permite al cliente de la aplicación:

- Pasar los datos y los atributos (los atributos se pasan a través de **objAttrPtr**) del objeto en una única llamada.
- Especificar parte de los datos del objeto a través de la llamada**dsmSendObj** y el resto de los datos a través de una o más llamadas **dsmSendData**.

Como alternativa, el cliente de la aplicación puede especificar sólo los atributos a través de la llamada **dsmSendObj** y especificar los datos del objeto a través de una o más llamadas a **dsmSendData**. Para este método, configure **dataBlkPtr** en NULO en la llamada **dsmSendObj**.

Consejo: En determinados tipos de objetos, es posible que los datos de la corriente de bytes no se asocien con los datos; por ejemplo, una entrada de directorio sin atributos ampliados.

Antes de llamar a **dsmSendObj**, se debe realiza una llamada **dsmBindMC** previa con el fin de vincular adecuadamente una clase de gestión con el objeto que desea archivar o realizar una copia de seguridad. La API mantiene esta vinculación para que pueda asociar a la clase de gestión adecuada con el objeto cuando se envía al servidor. Si permite que la clase de gestión vinculada a una llamada **dsmSendObj** sea un tipo de objeto del directorio (DSM_OBJ_DIRECTORY), es posible que la clase predeterminada no sea la clase de gestión. En su lugar, se utiliza la clase de gestión con mayor tiempo de retención. Si existe más de una clase de gestión con este tiempo de retención, se utiliza la primera que se encuentra.

Siga todos los datos de objeto que se envían al almacenamiento con una llamada **dsmEndSendObj**. Si no tiene datos de objeto para enviar al servidor, o todos los datos se encontraban dentro de la llamada **dsmSendObj**, inicie una llamada **dsmEndSendObj** antes de iniciar otra llamada **dsmSendObj**. Si se requerían varios envíos de datos a través de la llamada **dsmSendData**, **dsmEndSendObj** sigue el último envío para indicar el cambio de estado.

Consejo: Si IBM Spectrum Protect devuelve el código 157 (DSM_RC_WILL_ABORT), inicie una llamada a **dsmEndTxn** con un voto de DSM_VOTE_COMMIT. La aplicación recibe el código de retorno 2302 (DSM_RC_CHECK_REASON_CODE) y se lo vuelve a pasar al usuario de la aplicación. Esto informa al usuario por qué el servidor está terminando la transacción.

Si el código de retorno es 11 (DSM_RS_ABORT_NO_REPOSIT_SPACE), es posible que *sizeEstimate* sea demasiado pequeño para la cantidad real de datos. La aplicación necesita determinar un *sizeEstimate* más preciso y volver a enviar los datos.

Sintaxis

```
dsInt16_t dsmSendObj  (dsUint32_t       dsmHandle,
   dsmSendType  sendType,
   void        *sendBuff,
   dsmObjName  *objNameP,
   ObjAttr     *objAttrPtr,
   DataBlk     *dataBlkPtr);
```

Parámetros

dsUint32_t dsmHandle (I)
El manejador que asocia esta llamada con una llamada **dsmInitEx** anterior.

dsmSendType sendType (I)
Este parámetro especifica el tipo de envío que se está realizando. Dentro de los valores posibles se incluyen los siguientes:

Nombre	Descripción
stBackup	Un objeto de copia de seguridad que se envía al servidor.
stArchive	Una objeto de copia archivada que se envía al servidor.
stBackupMountWait	Un objeto de copia de seguridad al que desea que espere el servidor antes de montar el dispositivo necesario, tal como una cinta.
stArchiveMountWait	Una copia de un objeto archivado para el que desea que el servidor espere hasta que el dispositivo necesario, como una cinta, se monte.

Nota: Utilice los tipos **MountWait** si existe alguna posibilidad de que el usuario de la aplicación envíe datos a una cinta.

void *sendBuff (I)
Este parámetro es un puntero a la estructura que contiene otra información específica a **sendType** en la llamada. Actualmente, sólo un **sendType** de **stArchive** tiene una estructura asociada. Esta estructura se denomina **sndArchiveData** y contiene la descripción de la copia archivada.

dsmObjName *objNameP (I)
Este parámetro es un puntero a la estructura que contiene el nombre del espacio del archivo, el nombre de alto y bajo rango y el tipo de objeto. Para obtener más información, consulte el apartado "ID y nombres de objeto" en la página 23.

ObjAttr *objAttrPtr (I)
Este parámetro pasa atributos de objeto de interés a la aplicación. Consulte Apéndice B, "Archivos de origen de definiciones de tipo de API", en la página 165 para obtener más información sobre la definición del tipo.

Los atributos son:

- **owner** se refiere al propietario del objeto. Determinar si el propietario se declara ser un nombre específico o una cadena vacía es importante cuando se obtiene el objeto de vuelta del almacenamiento IBM Spectrum Protect. Para obtener más información, consulte el apartado "Acceder a objetos como propietario de sesión" en la página 25.
- **sizeEstimate** es la mejor estimación total del tamaño de los objetos de los datos que se envían al servidor. Es importante que intente ser lo más preciso posible en la estimación de tamaño, porque el servidor utiliza este atributo para optimizar la asignación del espacio y la colocación del objeto dentro de sus recursos de almacenamiento.

  Si el tamaño estimado que se ha especificado es considerablemente más pequeño que el número real de bytes que se están enviando, es posible que el servidor tenga dificultades para asignar espacio suficiente y finalice la transacción con un código de motivo 11 (DSM_RS_ABORT_NO_REPOSIT_SPACE).

  Nota: la estimación de tamaño es del tamaño total del objeto de datos en bytes.

  Los objetos con un tamaño más pequeño que DSM_MIN_COMPRESS_SIZE no se comprimen.

  Si el objeto no tiene datos de bit (sólo la información de atributo de esta llamada), el **sizeEstimate** debe ser cero.

  Nota: Comenzando con la versión 5.1.0, el destino de la copia dentro de una transacción no se comprueba la coherencia en objetos de longitud cero.
- **objCompressed** es un valor booleano que indica si los datos del objeto ya han sido comprimidos o no.

  Si el objeto está comprimido (objeto *compressed=bTrue*), IBM Spectrum Protect no intente comprimirlo de nuevo. Si no está comprimido, IBM Spectrum Protect decide si comprimir el objeto, en función del valor de la opción `compression` establecida por el administrador y establecida en los orígenes de configuración de API.

  Si la aplicación va a utilizar restauraciones o recuperaciones parciales de objetos, no es posible comprimir los datos mientras los envía. Para implementar esto, configure *ObjAttr.objCompressed* en *bTrue*.

- **objInfo** guarda información acerca del objeto en concreto.

  Restricción: La información no se almacena aquí automáticamente. Cuando se utiliza este atributo, debe establecer el atributo *objInfoLength*, para mostrar la longitud de *objInfo*.
- **mcNameP** contiene el nombre de la clase de gestión que ignora la clase de gestión que se obtiene de **dsmBindMC**.
- **disableDeduplication** es un valor booleano. Cuando se establece en true (verdadero) el cliente no elimina los duplicados de este objeto.

DataBlk *dataBlkPtr (I/O)

Este parámetro apunta a una estructura que incluye tanto un puntero al almacenamiento intermedio de datos que se van a archivar o realizar una copia de seguridad como el tamaño de dicho almacenamiento intermedio. Este parámetro se aplica únicamente a **dsmSendObj**. Si desea empezar a enviar datos a una llamada **dsmSendData** subsiguiente, en lugar de en la llamada **dsmSendObj**, configure el puntero del almacenamiento intermedio en la estructura DataBlk en NULO. Cuando se devuelven los datos, esta estructura contiene el número de bytes que se han transferido. Consulte Apéndice B, "Archivos de origen de definiciones de tipo de API", en la página 165 para obtener más información sobre la definición del tipo.

Códigos de retorno

Los números de código de retorno se proporcionan entre paréntesis ( ).

Tabla 51. Códigos de retorno para dsmSendObj

Código de retorno	Explicación
DSM_RC_NO_COMPRESS_MEMORY (154)	No hay memoria suficiente disponible para llevar a cabo una compresión o expansión de datos.
DSM_RC_COMPRESS_GREW (155)	Durante la compresión los datos comprimidos crecen en tamaño comparados con los datos originales.
DSM_RC_WILL_ABORT (157)	Ha ocurrido un error desconocido e inesperado, y la transacción se ha detenido.
DSM_RC_TL_NOACG (186)	La clase de gestión para este archivo no cuenta con un grupo de copia válido para el tipo de envío.
DSM_RC_NULL_OBJNAME (2000)	Nombre de objeto nulo.
DSM_RC_NULL_OBJATTRPTR (2004)	Puntero de atributo de objeto nulo.
DSM_RC_INVALID_OBJTYPE (2010)	Tipo de objeto no válido.
DSM_RC_INVALID_OBJOWNER (2019)	Propietario de objeto no válido.
DSM_RC_INVALID_SENDTYPE (2022)	Tipo de envío no válido.
DSM_RC_WILDCHAR_NOTALLOWED (2050)	Los caracteres comodín no están permitidos.
DSM_RC_FS_NOT_REGISTERED (2061)	El espacio de archivo no está registrado.
DSM_RC_WRONG_VERSION_PARM (2065)	La versión de la API del cliente de aplicación es distinta de la versión de la biblioteca de IBM Spectrum Protect.
DSM_RC_NEEDTO_ENDTXN (2070)	Necesita finalizar la transacción.
DSM_RC_OBJ_EXCLUDED (2080)	La lista de inclusión/exclusión excluye al objeto.
DSM_RC_OBJ_NOBCG (2081)	El objeto no tiene un grupo de copia de seguridad y no se enviará al servidor.
DSM_RC_OBJ_NOACG (2082)	El objeto no tiene un grupo de archivado y no se enviará al servidor.

Tabla 51. Códigos de retorno para dsmSendObj (continuación)

Código de retorno	Explicación
DSM_RC_DESC_TOOLONG (2100)	La descripción es demasiado larga.
DSM_RC_OBJINFO_TOOLONG (2101)	La información del objeto es demasiado larga.
DSM_RC_HL_TOOLONG (2102)	El calificador de alto rango es demasiado largo.
DSM_RC_FILESPACE_TOOLONG (2104)	El nombre del espacio de archivo es demasiado largo.
DSM_RC_LL_TOOLONG (2105)	El calificador de bajo rango es demasiado largo.
DSM_RC_NEEDTO_CALL_BINDMC (2301)	debe llamarse primero a **dsmBindMC**.

dsmSetAccess

La llamada de función **dsmSetAccess** da a otros usuarios o nodos acceso a versiones de copia de seguridad o copias archivadas de los objetos, acceso a todos los objetos o acceso a un grupo determinado. Cuando se otorga acceso a otro usuario, dicho usuario puede consultar, restaurar o recuperar los archivos. Este mandato soporta comodines en los siguientes campos: *fs*, *hl*, *ll*, *nodo*, *propietario*.

Nota: No puede conceder acceso a copias de seguridad y copias archivadas con un único mandato. Debe especificar copia de seguridad o archivado.

Sintaxis

```
dsInt16_t DSMLINKAGE dsmSetAccess
        (dsUint32_t          dsmHandle,
         dsmSetAccessType    accessType,
         dsmObjName          *objNameP,
         char                *node,
         char                *owner);
```

Parámetros

dsUint32_t dsmHandle (I)
El manejador que asocia esta llamada con una llamada **dsmInitEx** anterior.

dsmAccessType accessType (I)
Este parámetro especifica el tipo de objeto para el que desea dar acceso. Dentro de los valores posibles se incluyen los siguientes:

Nombre	Descripción
atBackup	Especifica que el acceso se configura para copias de seguridad de objetos.
atArchive	Especifica que el acceso se configura para objetos archivados.

dsmObjName *objNameP (I)
Este parámetro es un puntero a la estructura que contiene el nombre de espacio de archivo, y el nombre de alto y bajo rango del objeto.

Nota: Con el fin de especificar todos los espacios del archivo, utilice un asterisco (*) en el nombre del espacio del archivo.

char *node (I)
Este parámetro es un puntero al nombre del nodo al que se le concede acceso. Para cualquier nodo, especifique un asterisco (*).

char *owner (I)
Este parámetro es un puntero al nombre de usuario en el nodo al que se le concede acceso. Para todos los usuarios, especifique un asterisco (*).

Códigos de retorno

Los números de código de retorno se proporcionan entre paréntesis ( ).

Tabla 52. Códigos de retorno para dsmSetAccess

Código de retorno	Explicación
DSM_RC_INVALID_ACCESS_TYPE (2110)	Tipo de acceso especificado no válido.
DSM_RC_FILE_SPACE_NOT_FOUND (124)	No se ha encontrado el espacio de archivo especificado en el servidor.
DSM_RC_QUERY_COMM_FAILURE (2111)	Error de comunicación durante la consulta del servidor.
DSM_RC_NO_FILES_BACKUP (2112)	No se realizaron copias de seguridad en este espacio de archivo.
DSM_RC_NO_FILES_ARCHIVE (2113)	No se realizaron archivaciones en este espacio de archivo.
DSM_RC_INVALID_SETACCESS (2114)	Formulación de acceso no válida.

dsmSetUp

La llamada de función **dsmSetUp** sobrescribe los valores de la variable del entorno. Llame a **dsmSetUp** antes de a **dsmInitEx**. Los valores que se pasan en la estructura **envSetUp** sobrescriben las variables de entorno existentes o predeterminadas. Si especifica NULL en un campo, los valores se toman del entorno. Si no configura un valor, se toman los valores predeterminados.

Requisitos:

1. Si utiliza **dsmSetUp**, siempre llamea **dsmTerminate** antes de **dsmCleanUp**.
2. La instrumentación de la API solo se puede activar si se ha establecido el distintivo de prueba INSTRUMENT: API en el archivo de configuración y se utilizan las llamadas **dsmSetUp** o **dsmCleanUp** en la aplicación.

Sintaxis

```
dsInt16_t DSMLINKAGE dsmSetUp
        (dsBool_t    mtFlag,
         envSetUp    *envSetUpP);
```

Parámetros

dsBool_t mtFlag (I)
Este parámetro especifica si la API se utilizará en una hebra única o en un modo de varias hebras. Los valores incluyen:

```
DSM_SINGLETHREAD
DSM_MULTITHREAD
```

Requisito: El indicador de varias hebras debe estar activado para que se lleve a cabo una transferencia de datos sin LAN.

envSetUp *envSetUpP(I)
Este parámetro es un puntero a la estructura que retiene los valores de sobrescritura. Especifique NULO si no quiere modificar las variables de entorno existentes. Los campos en la estructura **envSetUp** incluyen:

Nombre	Descripción
dsmiDir	Una ruta de directorio calificada al completo que contiene un archivo de mensaje en UNIX o Linux. También especifica los directorios dsmtca y dsm.sys.
dsmiConfig	El nombre completo del archivo de opciones del cliente.
dsmiLog	La ruta completa del directorio del registro de errores.
argv	Pase el nombre de argv[0] del programa de llamada si la aplicación debe ejecutarse con autoridad de usuario autorizado. Para obtener más información, consulte el apartado "Configuración de la opción `passwordaccess` a `generate` sin TCA" en la página 21.
logName	El nombre del archivo de un registro de error si la aplicación no utiliza dsierror.log.
inclExclCaseSensitive	Indica si las normas de inclusión/exclusión distinguen o no entre mayúsculas y minúsculas. Este parámetro se puede utilizar únicamente en Windows, se ignora en otros sistemas.

Códigos de retorno

Los números de código de retorno se proporcionan entre paréntesis ( ).

Tabla 53. Códigos de retorno para dsmSetUp

Código de retorno	Explicación
DSM_RC_ACCESS_DENIED (106)	Se ha negado el acceso al archivo o directorio especificado.
DSM_RC_INVALID_OPT (0400)	Se ha encontrado una opción no válida.
DSM_RC_NO_HOST_ADDR (0405)	La TCPSERVERADDRESS de este servidor no está definida en la sección de nombre de servidor del archivo de opciones del sistema.
DSM_RC_NO_OPT_FILE (0406)	No se ha podido encontrar el archivo de opciones especificado por el nombre de archivo.
DSM_RC_MACHINE_SAME (0408)	El NODENAME definido en el archivo de opciones no puede ser el mismo que el del sistema *HostName*.
DSM_RC_INVALID_SERVER (0409)	El archivo de opciones del sistema no contiene la opción SERVERNAME.
DSM_RC_INVALID_KEYWORD (0410)	Se ha encontrado una palabra clave de opción no válida en el archivo de configuración **dsmInitEx**, la cadena de opción, dsm.sys o dsm.opt.
DSM_RC_PATTERN_TOO_COMPLEX (0411)	El patrón de inclusión o exclusión emitido es demasiado complicado para que IBM Spectrum Protect lo interprete de forma precisa.
DSM_RC_NO_CLOSING_BRACKET (0412)	El patrón de inclusión o exclusión no está construido correctamente. Falta el paréntesis de cierre.
DSM_RC_NLS_CANT_OPEN_TXT (0610)	El sistema no puede abrir el archivo de texto del mensaje.
DSM_RC_NLS_INVALID_CNTL_REC (0612)	El sistema no es capaz de utilizar el archivo de texto del mensaje.
DSM_RC_NOT_ADSM_AUTHORIZED (0927)	Debe ser el usuario autorizado para tener multihebra y generar *passwordaccess*.
DSM_RC_NO_INCLEXCL_FILE (2229)	No se ha encontrado el archivo de inclusión/exclusión.
DSM_RC_NO_SYS_OR_INCLEXCL (2230)	No se ha encontrado el dsm.sys o el archivo de inclusión/exclusión.

dsmTerminate

La llamada de función **dsmTerminate** finaliza una sesión con el servidor IBM Spectrum Protect y limpia el entorno IBM Spectrum Protect.

Sintaxis

No existen códigos de retorno específicos para esta llamada.

```
dsInt16_t dsmTerminate (dsUint32_t dsmHandle);
```

Parámetros

dsUint32_t dsmHandle (I)
El manejador que asocia esta llamada con una llamada **dsmInitEx** anterior.

dsmUpdateFS

La llamada a función **dsmUpdateFS** actualiza un espacio de archivos en el almacenamiento IBM Spectrum Protect. Esta actualización garantiza que el administrador tenga un registro actualizado de su espacio de archivos.

Sintaxis

```
dsInt16_t dsmUpdateFS  (dsUint32_t      dsmHandle,
   char         *fs,
   dsmFSUpd     *fsUpdP,
   dsUint32_t    fsUpdAct);
```

Parámetros

dsUint32_t dsmHandle (I)
El manejador que asocia esta llamada con una llamada **dsmInitEx** anterior.

char *fs (I)
Este parámetro es un puntero al nombre del espacio de archivo.

dsmFSUpd *fsUpdP (I)
Este parámetro es un puntero a la estructura que tiene los campos correctos para la actualización que desea. Complete únicamente aquellos campos que necesitan actualización.

dsUint32_t fsUpdAct (I)
Un mapa de 2 bytes que indica cuál de los campos actualizar. Las máscaras de bits tienen los siguientes valores:

- DSM_FSUPD_FSTYPE
- DSM_FSUPD_FSINFO

  Consejo: Para del sistema operativo Windows, el valor de la letra de la unidad de **dsmDOSAttrib** también se actualiza cuando se selecciona **FSINFO**.
- DSM_FSUPD_OCCUPANCY
- DSM_FSUPD_CAPACITY
- DSM_FSUPD_BACKSTARTDATE
- DSM_FSUPD_BACKCOMPLETEDATE

Para obtener una descripción de estas máscaras de bits, consulte las definiciones de DSM_FSUPD en el tema siguiente: Apéndice B, “Archivos de origen de definiciones de tipo de API”, en la página 165.

Códigos de retorno

La tabla siguiente lista los códigos de retorno para la llamada a función **dsmUpdateFS**.

Tabla 54. Los códigos de retorno de dsmUpdateFS

Código de retorno	Número de código de retorno	Descripción
DSM_RC_FS_NOT_REGISTERED	2061	El nombre del espacio de archivos no está registrado.
DSM_RC_WRONG_VERSION_PARM	2065	La versión de la API del cliente de aplicación es distinta de la versión de la biblioteca de IBM Spectrum Protect.
DSM_RC_FSINFO_TOOLONG	2106	La información del espacio de archivo es demasiado larga.

dsmUpdateObj

La llamada de función **dsmUpdateObj** actualiza la metainformación asociada con un objeto archivado o copia de seguridad activos que ya está en el servidor. Los datos de bit de la aplicación no se ven afectados. Para actualizar un objeto, debe asignar un nombre específico sin comodines. Para actualizar un objeto archivado, configure **dsmSendType** en **stArchive**. Solo se actualiza el último objeto archivado nombrado.

Sólo puede iniciar la llamada **dsmUpdateObj** en el estado de la sesión; no se puede llamar dentro de una transacción porque realiza su propia transacción. Además, solo se puede actualizar un objeto cada vez.

Restricción: En un sistema operativo UNIX o Linux, si cambia el campo de propietario no puede consultar o restaurar al objeto a no ser que sea el usuario root.

Sintaxis

```
dsInt16_t  dsmUpdateObj
   (dsUint32_t      dsmHandle,
    dsmSendType     sendType,
    void           *sendBuff,
    dsmObjName     *objNameP,
    ObjAttr        *objAttrPtr,  /* objInfo */
    dsUint16_t      objUpdAct);  /* action bit vector */
```

Parámetros

Las descripciones del campo son las mismas que aquellas en **dsmSendObj**, con las siguientes excepciones:

dsmObjName *objNameP (I)
No puede utilizar un comodín.

ObjAttr *objAttrPtr (I)
El campo **objCompressed** se ignora para esta llamada.

Otras diferencias son:

- **propietario**. Si especifica un campo **propietario** nuevo, el propietario cambia.

- **sizeEstimate**. Si especifica un valor que no es cero debe ser la cantidad real de datos enviados en bytes. El valor se almacena en los metadatos de IBM Spectrum Protect para su futuro uso.
- **objInfo**. Este atributo contiene la información nueva que se debe colocar en el campo **objInfo**. Configure **objInfoLength** con la longitud del nuevo **obiInfo**.

dsUint16_t objUpdAct
Las máscaras de bit y las acciones posibles de **objUpdAct** son:

DSM_BACKUPD_MC
Actualiza la clase de gestión del objeto.

DSM_BACKUPD_OBJINFO
Actualiza **objInfo**, **objInfoLength** y **sizeEstimate**.

DSM_BACKUPD_OWNER
Actualiza al propietario del objeto.

DSM_ARCHUPD_DESCR
Actualiza el campo **Descripción**. Introduzca el valor de la descripción nueva a través del parámetro **SendBuff**. Consulte el programa de ejemplo para su uso adecuado.

DSM_ARCHUPD_OBJINFO
Actualiza **objInfo**, **objInfoLength** y **sizeEstimate**.

DSM_ARCHUPD_OWNER
Actualiza al propietario del objeto.

Códigos de retorno

Los números de código de retorno se proporcionan entre paréntesis).

Tabla 55. Códigos de retorno para dsmUpdateObj

Código de retorno	Explicación
DSM_RC_INVALID_ACTION (2232)	Acción no válida.
DSM_RC_FS_NOT_REGISTERED (2061)	El espacio de archivo no está registrado.
DSM_RC_BAD_CALL_SEQUENCE (2041)	La secuencia de llamadas no es válida.
DSM_RC_WILDCHAR_NOTALLOWED (2050)	No están permitidos los caracteres comodín.
DSM_RC_ABORT_NO_MATCH (2)	La consulta anterior no coincide.

dsmUpdateObjEx

La llamada de función **dsmUpdateObjEx** actualiza la meta información que está asociada con los objetos de archivado o las copias de seguridad en el servidor. Los datos de bit de la aplicación no se ven afectados. Para actualizar un objeto, debe asignar un nombre sin comodines o bien especificar el ID del objeto con el fin de actualizar un objeto archivado específico. No se pueden utilizar caracteres comodín cuando se especifica el nombre. Para actualizar un objeto al que se le ha realizado una copia de seguridad, configure el parámetro **dsmSendType** en **stBackup**. Para actualizar un objeto archivado, configure el parámetro **dsmSendType** en **stArchive**.

Sólo puede iniciar la llamada **dsmUpdateObjEx** en el estado de la sesión; no se puede llamar dentro de una transacción porque realiza su propia transacción. Los objetos sólo se pueden actualizar de uno en uno.

Restricción: En un sistema operativo UNIX o Linux, si cambia el campo del propietario no puede consultar o restaurar al objeto a no ser que sea el usuario root. Sólo se puede actualizar la versión activa actual de una copia de seguridad.

Sintaxis

```
dsInt16_t  dsmUpdateObjEx
   (dsmUpdateObjExIn_t   *dsmUpdateObjExInP,
    dsmUpdateObjExOut_t  *dsmUpdateObjExOutP);
```

Parámetros

dsmUpdateObjExIn_t *dsmUpdateObjExInP
Esta estructura contiene los siguientes parámetros de entrada:

dsUint16_t stVersion (I)
La versión actual de la estructura que se utiliza.

dsUint32_t dsmHandle (I)
El manejador que asocia esta llamada con una llamada **dsmInitEx** anterior.

dsmSendType sendType (I)
El tipo de envío que se está realizando. El valor puede ser:

stBackup
Un objeto de copia de seguridad que se envía al servidor.

stArchive
Una objeto de copia archivada que se envía al servidor.

dsmObjName *objNameP (I)
Un puntero a la estructura que contiene el nombre del espacio del archivo, el nombre de alto y bajo rango y el tipo de objeto. No puede utilizar un comodín.

ObjAttr *objAttrPtr (I)
Pasa atributos de objeto a la aplicación. Los valores que se actualizan dependen del indicador en el campo **objUpdAct**. El atributo **objCompressed** se omite en esta llamada.

Los atributos son:

- **owner** cambia el propietario si se especifica un nombre nuevo.
- **sizeEstimate** es la cantidad de datos real que se envía en bytes. El valor se almacena en los metadatos de IBM Spectrum Protect para uso futuro.
- **objCompressed** es un valor booleano que indica si los datos del objeto ya han sido comprimidos o no.
- **objInfo** es un atributo que contiene la información nueva colocada en el campo **objInfo**. Configure **objInfoLength** con la longitud del nuevo **objInfo**.
- **mcNameP** contiene el nombre de la clase de gestión que ignora la clase de gestión que se obtiene de **dsmBindMC**.

dsUint32_t objUpdAct
Especifica las máscaras de bit y las acciones de **objUpdAct** son:

DSM_BACKUPD_MC
Actualiza la clase de gestión del objeto.

DSM_BACKUPD_OBJINFO
Actualiza el objeto de información (**objInfo**), la longitud del objeto de información (**objInfoLength**), y la cantidad de datos que se envían (**sizeEstimate**) en la copia de seguridad del objeto.

DSM_BACKUPD_OWNER
Actualiza al propietario de la copia de seguridad del objeto.

DSM_ARCHUPD_DESCR
Actualiza el campo **Description** del objeto archivado. Introduzca el valor de la descripción nueva a través del parámetro **sendBuff**.

DSM_ARCHUPD_OBJINFO
Actualiza el objeto de la información (**objInfo**), la longitud del objeto de información (**objInfoLength**), y la cantidad de datos que se envían (**sizeEstimate**) en el objeto archivado.

DSM_ARCHUPD_OWNER
Actualiza el propietario del objeto archivado.

ObjID archObjId
Especifica el ID del objeto único de un objeto archivado determinado. Debido a que varios objetos archivados pueden tener el mismo nombre, este parámetro identifica uno en concreto. Es posible obtener el ID del objeto utilizando una llamada de consulta de archivado.

dsmUpdateObjExOut_t *dsmUpdateObjExOutP
Esta estructura contiene los parámetros de salida:

dsUint16_t stVersion (I)
La versión actual de la estructura que se utiliza.

Códigos de retorno

Los números del código de retorno se proporcionan entre paréntesis ( ) en la siguiente tabla.

Tabla 56. Códigos de retorno de dsmUpdateObjEx

Código de retorno	Explicación
DSM_RC_INVALID_ACTION (2012)	Acción no válida.
DSM_RC_FS_NOT_REGISTERED (2061)	El espacio de archivo no está registrado.
DSM_RC_BAD_CALL_SEQUENCE (2041)	La secuencia de llamadas no es válida.
DSM_RC_WILDCHAR_NOTALLOWED (2050)	No están permitidos los caracteres comodín.
DSM_RC_ABORT_NO_MATCH (2)	La consulta anterior no coincide.

Apéndice A. Archivo de origen de códigos de retorno de la API: dsmrc.h

El archivo de cabecera dsmrc.h contiene todos los códigos de retorno que la API puede devolver a una aplicación.

La información que se proporciona aquí ofrece una copia de puntual del archivo dsmrc.h que se distribuye con la API. Vea el archivo en el paquete de distribución de la API para la última versión.

```
/*****************************************************************
* Tivoli Storage Manager                                         *
* API Client Component                                           *
*                                                                *
* (C) Copyright IBM Corporation 1993,2010                        *
*****************************************************************/

/*****************************************************************/
/* Header File Name:  dsmrc.h                                    */
/*                                                               */
/* Descriptive-name:  Return codes from Tivoli Storage Manager APIs   */
/*****************************************************************/
#ifndef _H_DSMRC
#define _H_DSMRC


#ifndef DSMAPILIB

#ifndef _H_ANSMACH
typedef int RetCode ;
#endif

#endif


#define DSM_RC_SUCCESSFUL               0 /* successful completion   */
#define DSM_RC_OK                       0 /* successful completion   */

#define DSM_RC_UNSUCCESSFUL            -1 /* unsuccessful completion */

/* 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    /* Partial Object Retrieve */
#define DSM_RS_ABORT_INVALID_LENGTH          34    /* Partial Object Retrieve */
#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


/* Definitions for server signon reject codes                          */
/* These error codes are in the range (51 to 99) inclusive.            */
#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 /* client type not licensed */
#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 /* processing aborted by user        */
#define DSM_RC_NO_MEMORY           102 /* no RAM left to complete request   */
#define DSM_RC_TA_COMM_DOWN       2021 /* no longer used                    */
#define DSM_RC_FILE_NOT_FOUND      104 /* specified file not found          */
#define DSM_RC_PATH_NOT_FOUND      105 /* specified path doesn't exist      */
#define DSM_RC_ACCESS_DENIED       106 /* denied due to improper permission */
#define DSM_RC_NO_HANDLES          107 /* no more file handles available    */
#define DSM_RC_FILE_EXISTS         108 /* file already exists               */
#define DSM_RC_INVALID_PARM        109 /* invalid parameter passed. CRITICAL*/
#define DSM_RC_INVALID_HANDLE      110 /* invalid file handle passed        */
#define DSM_RC_DISK_FULL           111 /* out of disk space                 */
#define DSM_RC_PROTOCOL_VIOLATION  113 /* call protocol violation. CRITICAL */
#define DSM_RC_UNKNOWN_ERROR       114 /* unknown system error. CRITICAL    */
#define DSM_RC_UNEXPECTED_ERROR    115 /* unexpected error. CRITICAL        */
#define DSM_RC_FILE_BEING_EXECUTED 116 /* No write is allowed               */
#define DSM_RC_DIR_NO_SPACE        117 /* directory can't be expanded       */
#define DSM_RC_LOOPED_SYM_LINK     118 /* too many symbolic links were
                                          encountered in translating path.  */
#define DSM_RC_FILE_NAME_TOO_LONG  119 /* file name too long                */
#define DSM_RC_FILE_SPACE_LOCKED   120 /* filespace is locked by the system */
#define DSM_RC_FINISHED            121 /* finished processing               */
#define DSM_RC_UNKNOWN_FORMAT      122 /* unknown format                    */
#define DSM_RC_NO_AUTHORIZATION    123 /* server response when the client has
                                          no authorization to read another
                                          host's owner backup/archive data  */
#define DSM_RC_FILE_SPACE_NOT_FOUND 124/* specified file space not found    */
#define DSM_RC_TXN_ABORTED         125 /* transaction aborted               */
#define DSM_RC_SUBDIR_AS_FILE      126 /* Subdirectory name exists as file  */
#define DSM_RC_PROCESS_NO_SPACE    127 /* process has no more disk space.   */
#define DSM_RC_PATH_TOO_LONG       128 /* a directory path being build became
                                          too long                          */
#define DSM_RC_NOT_COMPRESSED      129 /* file thought to be compressed is
                                          actually not                      */
#define DSM_RC_TOO_MANY_BITS       130 /* file was compressed using more bits
                                          then the expander can handle      */
#define DSM_RC_SYSTEM_ERROR        131 /* internal system error             */
#define DSM_RC_NO_SERVER_RESOURCES 132 /* server out of resources.          */
#define DSM_RC_FS_NOT_KNOWN        133 /* the file space is not known by the
                                          server                            */
#define DSM_RC_NO_LEADING_DIRSEP   134 /* no leading directory separator    */
#define DSM_RC_WILDCARD_DIR        135 /* wildcard character in directory
                                          path when not allowed             */
#define DSM_RC_COMM_PROTOCOL_ERROR 136 /* communications protocol error     */
#define DSM_RC_AUTH_FAILURE        137 /* authentication failure            */
#define DSM_RC_TA_NOT_VALID        138 /* TA not a root and/or SUID program */
#define DSM_RC_KILLED              139 /* process killed.                   */

#define DSM_RC_RETRY               143 /* retry same operation again        */

#define DSM_RC_WOULD_BLOCK         145 /* operation would cause the system to
                                          block waiting for input.          */
#define DSM_RC_TOO_SMALL           146 /* area for compiled pattern small   */
#define DSM_RC_UNCLOSED            147 /* no closing bracket in pattern     */
#define DSM_RC_NO_STARTING_DELIMITER 148 /* pattern has to start with
                                            directory delimiter             */
#define DSM_RC_NEEDED_DIR_DELIMITER 149 /* a directory delimiter is needed
                                          immediately before and after the
                                          "match directories" metastring
                                          ("...") and one wasn't found      */
#define DSM_RC_UNKNOWN_FILE_DATA_TYPE 150 /* structured file data type is
                                             unknown                        */
#define DSM_RC_BUFFER_OVERFLOW     151 /* data buffer overflow              */


#define DSM_RC_NO_COMPRESS_MEMORY  154 /* Compress/Expand out of memory     */
#define DSM_RC_COMPRESS_GREW       155 /* Compression grew                  */
#define DSM_RC_INV_COMM_METHOD     156 /* Invalid comm method specified     */
#define DSM_RC_WILL_ABORT          157 /* Transaction will be aborted       */
#define DSM_RC_FS_WRITE_LOCKED     158 /* File space is write locked        */
#define DSM_RC_SKIPPED_BY_USER     159 /* User wanted file skipped in the
```

```
                                        case of ABORT_DATA_OFFLINE        */
#define DSM_RC_TA_NOT_FOUND        160 /* TA not found in it's directory    */
#define DSM_RC_TA_ACCESS_DENIED    161 /* Access to TA is denied            */
#define DSM_RC_FS_NOT_READY        162 /* File space not ready              */
#define DSM_RC_FS_IS_BAD           163 /* File space is bad                 */
#define DSM_RC_FIO_ERROR           164 /* File input/output error           */
#define DSM_RC_WRITE_FAILURE       165 /* Error writing to file             */
#define DSM_RC_OVER_FILE_SIZE_LIMIT 166 /* File over system/user limit      */
#define DSM_RC_CANNOT_MAKE         167 /* Could not create file/directory,
                                        could be a bad name               */
#define DSM_RC_NO_PASS_FILE        168 /* password file needed and user is
                                        not root                          */
#define DSM_RC_VERFILE_OLD         169 /* password stored locally doesn't
                                        match the one at the host         */
#define DSM_RC_INPUT_ERROR         173 /* unable to read keyboard input     */
#define DSM_RC_REJECT_PLATFORM_MISMATCH 174 /* Platform name doesn't match
                                           up with what the server says
                                           is the platform for the client */
#define DSM_RC_TL_NOT_FILE_OWNER   175 /* User trying to backup a file is not
                                        the file's owner.                 */
#define DSM_RC_COMPRESSED_DATA_CORRUPTED 176 /* Compressed data is corrupted*/
#define DSM_RC_UNMATCHED_QUOTE     177 /* missing starting or ending quote  */

#define DSM_RC_SIGNON_FAILOVER_MODE 178      /* Failed over to the replication server,
                                                running in failover mode */
#define DSM_RC_FAILOVER_MODE_FUNC_BLOCKED 179 /* function is blocked because
                                                 session is in failover mode */

/*----------------------------------------------------------------------------*/
/* Return codes 180-199 are reserved for Policy Set handling                  */
/*----------------------------------------------------------------------------*/
#define DSM_RC_PS_MULTBCG          181 /* Multiple backup copy groups in 1 MC*/
#define DSM_RC_PS_MULTACG          182 /* Multiple arch.  copy groups in 1 MC*/
#define DSM_RC_PS_NODFLTMC         183 /* Default MC name not in policy set  */
#define DSM_RC_TL_NOBCG            184 /* Backup req, no backup copy group   */
#define DSM_RC_TL_EXCLUDED         185 /* Backup req, excl. by in/ex filter  */
#define DSM_RC_TL_NOACG            186 /* Archive req, no archive copy group */
#define DSM_RC_PS_INVALID_ARCHMC   187 /* Invalid MC name in archive override*/
#define DSM_RC_NO_PS_DATA          188 /* No policy set data on the server   */
#define DSM_RC_PS_INVALID_DIRMC    189 /* Invalid directory MC specified in
                                        the options file.                 */
#define DSM_RC_PS_NO_CG_IN_DIR_MC  190 /* No backup copy group in directory MC.
                                        Must specify an MC using DirMC
                                        option.                           */

#define DSM_RC_WIN32_UNSUPPORTED_FILE_TYPE 280 /* File is not of
                                                  Win32 type FILE_TYPE_DISK  */

/*----------------------------------------------------------------------------*/
/* Return codes for the Trusted Communication Agent                           */
/*----------------------------------------------------------------------------*/
#define DSM_RC_TCA_NOT_ROOT        161 /* Access to TA is denied             */
#define DSM_RC_TCA_ATTACH_SHR_MEM_ERR  200 /* Error attaching shared memory  */
#define DSM_RC_TCA_SHR_MEM_BLOCK_ERR   200 /* Shared memory block error      */
#define DSM_RC_TCA_SHR_MEM_IN_USE      200 /* Shared memory block error      */
#define DSM_RC_TCA_SHARED_MEMORY_ERROR 200 /* Shared memory block error      */
#define DSM_RC_TCA_SEGMENT_MISMATCH    200 /* Shared memory block error      */
#define DSM_RC_TCA_FORK_FAILED     292 /* Error forking off TCA process      */
#define DSM_RC_TCA_DIED            294 /* TCA died unexpectedly              */
#define DSM_RC_TCA_INVALID_REQUEST 295 /* Invalid request sent to TCA        */
#define DSM_RC_TCA_SEMGET_ERROR    297 /* Error getting semaphores           */
#define DSM_RC_TCA_SEM_OP_ERROR    298 /* Error in semaphore set or wait     */
#define DSM_RC_TCA_NOT_ALLOWED     299 /* TCA not allowed (multi thread)     */

/*----------------------------------------------------------------------------*/
/* 400-430  for options                                                       */
/*----------------------------------------------------------------------------*/
#define DSM_RC_INVALID_OPT           400 /* invalid option                   */
#define DSM_RC_NO_HOST_ADDR          405 /* Not enuf info to connect server  */
#define DSM_RC_NO_OPT_FILE           406 /* No default user configuration file*/
#define DSM_RC_MACHINE_SAME          408 /* -MACHINENAME same as real name   */
#define DSM_RC_INVALID_SERVER        409 /* Invalid server name from client  */
#define DSM_RC_INVALID_KEYWORD       410 /* Invalid option keyword           */
#define DSM_RC_PATTERN_TOO_COMPLEX   411 /* Can't match Include/Exclude entry*/
#define DSM_RC_NO_CLOSING_BRACKET    412 /* Missing closing bracket inc/excl */
#define DSM_RC_OPT_CLIENT_NOT_ACCEPTING 417/* Client doesn't accept this option
                                           from the server                 */
#define DSM_RC_OPT_CLIENT_DOES_NOT_WANT 418/* Client doesn't want this value
                                           from the server                 */
```

```
#define DSM_RC_OPT_NO_INCLEXCL_FILE  419   /* inclexcl file not found        */
#define DSM_RC_OPT_OPEN_FAILURE      420   /* can't open file               */
#define DSM_RC_OPT_INV_NODENAME      421/* used for Windows if nodename=local
                                        machine when CLUSTERNODE=YES       */
#define DSM_RC_OPT_NODENAME_INVALID  423/* generic invalid nodename          */
#define DSM_RC_OPT_ERRORLOG_CONFLICT 424/* both logmax & retention specified  */
#define DSM_RC_OPT_SCHEDLOG_CONFLICT 425/* both logmax & retention specified  */
#define DSM_RC_CANNOT_OPEN_TRACEFILE 426/* cannot open trace file             */
#define DSM_RC_CANNOT_OPEN_LOGFILE   427/* cannot open error log file         */
#define DSM_RC_OPT_SESSINIT_LF_CONFLICT 428/* both sessioninit=server and
               enablelanfree=yes are specified*/
#define DSM_RC_OPT_OPTION_IGNORE     429/* option will be ignored            */
#define DSM_RC_OPT_DEDUP_CONFLICT    430/* cannot open error log file        */
#define DSM_RC_OPT_HSMLOG_CONFLICT 431/* both logmax & retention specified  */


/*-----------------------------------------------------------------------------*/
/* 600 to 610 for volume label codes                                           */
/*-----------------------------------------------------------------------------*/
#define DSM_RC_DUP_LABEL           600 /* duplicate volume label found       */
#define DSM_RC_NO_LABEL            601 /* drive has no label                 */

/*-----------------------------------------------------------------------------*/
/* Return codes for message file processing                                    */
/*-----------------------------------------------------------------------------*/
#define DSM_RC_NLS_CANT_OPEN_TXT   610 /* error trying to open msg txt file  */
#define DSM_RC_NLS_CANT_READ_HDR   611 /* error trying to read header        */
#define DSM_RC_NLS_INVALID_CNTL_REC 612 /* invalid control record            */
#define DSM_RC_NLS_INVALID_DATE_FMT 613 /* invalid default date format       */
#define DSM_RC_NLS_INVALID_TIME_FMT 614 /* invalid default time format       */
#define DSM_RC_NLS_INVALID_NUM_FMT 615 /* invalid default number format      */

/*-----------------------------------------------------------------------------*/
/* Return codes 620-630 are reserved for log message return codes              */
/*-----------------------------------------------------------------------------*/
#define DSM_RC_LOG_CANT_BE_OPENED  620 /* error trying to open error log     */
#define DSM_RC_LOG_ERROR_WRITING_TO_LOG 621 /* error occurred writing to
                                       log file                             */
#define DSM_RC_LOG_NOT_SPECIFIED   622 /* no error log file was specified    */


/*-----------------------------------------------------------------------------*/
/* Return codes 900-999 TSM CLIENT ONLY                                        */
/*-----------------------------------------------------------------------------*/
#define DSM_RC_NOT_ADSM_AUTHORIZED  927 /* Must be ADSM authorized to perform*/
                                    /* action : root user or pwd auth    */
#define DSM_RC_REJECT_USERID_UNKNOWN 940 /* userid unknown on server         */
#define DSM_RC_FILE_IS_SYMLINK      959 /* errorlog or trace is a symbolic
                                       link
                                                                       */

#define DSM_RC_DIRECT_STORAGE_AGENT_UNSUPPORTED 961 /* Direct connection to SA not supported */
#define DSM_RC_FS_NAMESPACE_DOWNLEVEL 963 /* Long namespace has been removed from
                                     from the Netware volume */
#define DSM_RC_CONTINUE_NEW_CONSUMER   972 /* Continue processing using a new consumer */
#define DSM_RC_CONTINUE_NEW_CONSUMER_NODEDUP   973 /* Continue processing using a new consumer no dedup*/
#define DSM_RC_CONTINUE_NEW_CONSUMER_NOCOMPRESS 976 /* Continue processing using a new consumer no compression */


#define DSM_RC_SERVER_SUPPORTS_FUNC    994 /* the server supports this function  */
#define DSM_RC_SERVER_AND_SA_SUPPORT_FUNC 995 /* Both server and SA support func */
#define DSM_RC_SERVER_DOWNLEVEL_FUNC   996 /* The server is downlevel for func   */
#define DSM_RC_STORAGEAGENT_DOWNLEVEL  997 /* the storage agent is downlevel     */
#define DSM_RC_SERVER_AND_SA_DOWNLEVEL 998 /* both server and SA downlevel       */


/* TCP/IP error codes */
#define DSM_RC_TCPIP_FAILURE       -50 /* TCP/IP communications failure      */
#define DSM_RC_CONN_TIMEDOUT       -51 /* TCP/IP connection attempt timedout */
#define DSM_RC_CONN_REFUSED        -52 /* TCP/IP connection refused by host  */
#define DSM_RC_BAD_HOST_NAME       -53 /* TCP/IP invalid host name specified */
#define DSM_RC_NETWORK_UNREACHABLE -54 /* TCP/IP host name unreachable       */
#define DSM_RC_WINSOCK_MISSING     -55 /* TCP/IP WINSOCK.DLL missing         */
#define DSM_RC_TCPIP_DLL_LOADFAILURE -56 /* Error from LoadLibrary           */
#define DSM_RC_TCPIP_LOADFAILURE   -57 /* Error from GetProcAddress   */
#define DSM_RC_TCPIP_USER_ABORT    -58 /* User aborted while in TCP/IP layer */

/*-----------------------------------------------------------------------------*/
/* Return codes (-71)-(-90) are reserved for CommTSM error codes               */
```

```
/*----------------------------------------------------------------------------*/
#define DSM_RC_TSM_FAILURE         -71 /* TSM communications failure         */
#define DSM_RC_TSM_ABORT           -72 /* Session aborted abnormally         */

/*comm3270 error codes - no longer used*/
#define DSM_RC_COMM_TIMEOUT       2021   /* no longer used                    */
#define DSM_RC_EMULATOR_INACTIVE  2021   /* no longer used                    */
#define DSM_RC_BAD_HOST_ID        2021   /* no longer used                    */
#define DSM_RC_HOST_SESS_BUSY     2021   /* no longer used                    */
#define DSM_RC_3270_CONNECT_FAILURE 2021 /* no longer used                    */
#define DSM_RC_NO_ACS3ELKE_DLL    2021   /* no longer used                    */
#define DSM_RC_EMULATOR_ERROR     2021   /* no longer used                    */
#define DSM_RC_EMULATOR_BACKLEVEL 2021   /* no longer used                    */
#define DSM_RC_CKSUM_FAILURE      2021   /* no longer used                    */


/* The following Return codes are for EHLLAPI for Windows                    */
#define DSM_RC_3270COMMError_DLL        2021   /* no longer used             */
#define DSM_RC_3270COMMError_GetProc    2021   /* no longer used             */
#define DSM_RC_EHLLAPIError_DLL         2021   /* no longer used             */
#define DSM_RC_EHLLAPIError_GetProc     2021   /* no longer used             */
#define DSM_RC_EHLLAPIError_HostConnect 2021   /* no longer used             */
#define DSM_RC_EHLLAPIError_AllocBuff   2021   /* no longer used             */
#define DSM_RC_EHLLAPIError_SendKey     2021   /* no longer used             */
#define DSM_RC_EHLLAPIError_PacketChk   2021   /* no longer used             */
#define DSM_RC_EHLLAPIError_ChkSum      2021   /* no longer used             */
#define DSM_RC_EHLLAPIError_HostTimeOut 2021   /* no longer used             */
#define DSM_RC_EHLLAPIError_Send        2021   /* no longer used             */
#define DSM_RC_EHLLAPIError_Recv        2021   /* no longer used             */
#define DSM_RC_EHLLAPIError_General     2021   /* no longer used             */
#define DSM_RC_PC3270_MISSING_DLL       2021   /* no longer used             */
#define DSM_RC_3270COMM_MISSING_DLL     2021   /* no longer used             */


/* NETBIOS error codes */
#define DSM_RC_NETB_ERROR         -151 /* Could not add node to LAN          */
#define DSM_RC_NETB_NO_DLL        -152 /* The ACSNETB.DLL could not be loaded*/
#define DSM_RC_NETB_LAN_ERR       -155 /* LAN error detected                 */
#define DSM_RC_NETB_NAME_ERR      -158 /* Netbios error on Add Name          */
#define DSM_RC_NETB_TIMEOUT       -159 /* Netbios send timeout               */
#define DSM_RC_NETB_NOTINST       -160 /* Netbios not installed - DOS        */
#define DSM_RC_NETB_REBOOT        -161 /* Netbios config err - reboot DOS    */

/* Named Pipe error codes */
#define DSM_RC_NP_ERROR                       -190

/* CPIC error codes */
#define DSM_RC_CPIC_ALLOCATE_FAILURE          2021 /* no longer used      */
#define DSM_RC_CPIC_TYPE_MISMATCH             2021 /* no longer used      */
#define DSM_RC_CPIC_PIP_NOT_SPECIFY_ERR       2021 /* no longer used      */
#define DSM_RC_CPIC_SECURITY_NOT_VALID        2021 /* no longer used      */
#define DSM_RC_CPIC_SYNC_LVL_NO_SUPPORT       2021 /* no longer used      */
#define DSM_RC_CPIC_TPN_NOT_RECOGNIZED        2021 /* no longer used      */
#define DSM_RC_CPIC_TP_ERROR                  2021 /* no longer used      */
#define DSM_RC_CPIC_PARAMETER_ERROR           2021 /* no longer used      */
#define DSM_RC_CPIC_PROD_SPECIFIC_ERR         2021 /* no longer used      */
#define DSM_RC_CPIC_PROGRAM_ERROR             2021 /* no longer used      */
#define DSM_RC_CPIC_RESOURCE_ERROR            2021 /* no longer used      */
#define DSM_RC_CPIC_DEALLOCATE_ERROR          2021 /* no longer used      */
#define DSM_RC_CPIC_SVC_ERROR                 2021 /* no longer used      */
#define DSM_RC_CPIC_PROGRAM_STATE_CHECK       2021 /* no longer used      */
#define DSM_RC_CPIC_PROGRAM_PARAM_CHECK       2021 /* no longer used      */
#define DSM_RC_CPIC_UNSUCCESSFUL              2021 /* no longer used      */
#define DSM_RC_UNKNOWN_CPIC_PROBLEM           2021 /* no longer used      */
#define DSM_RC_CPIC_MISSING_LU                2021 /* no longer used      */
#define DSM_RC_CPIC_MISSING_TP                2021 /* no longer used      */
#define DSM_RC_CPIC_SNA6000_LOAD_FAIL         2021 /* no longer used      */
#define DSM_RC_CPIC_STARTUP_FAILURE           2021 /* no longer used      */

/*----------------------------------------------------------------------------*/
/* Return codes -300 to -307 are reserved for IPX/SPX communications         */
/*----------------------------------------------------------------------------*/
#define DSM_RC_TLI_ERROR                      2021 /* no longer used      */
#define DSM_RC_IPXSPX_FAILURE                 2021 /* no longer used      */
#define DSM_RC_TLI_DLL_MISSING                2021 /* no longer used      */
#define DSM_RC_DLL_LOADFAILURE                2021 /* no longer used      */
#define DSM_RC_DLL_FUNCTION_LOADFAILURE       2021 /* no longer used      */
#define DSM_RC_IPXCONN_REFUSED                2021 /* no longer used      */
#define DSM_RC_IPXCONN_TIMEDOUT               2021 /* no longer used      */
```

```
#define DSM_RC_IPXADDR_UNREACHABLE            2021 /* no longer used      */
#define DSM_RC_CPIC_MISSING_DLL               2021 /* no longer used      */
#define DSM_RC_CPIC_DLL_LOADFAILURE           2021 /* no longer used      */
#define DSM_RC_CPIC_FUNC_LOADFAILURE          2021 /* no longer used      */

/*=== Shared Memory Protocol error codes   ===*/
#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 /* Object name pointer is NULL      */
#define DSM_RC_NULL_DATABLKPTR     2001 /* dataBlkPtr is NULL               */
#define DSM_RC_NULL_MSG            2002 /* msg parm in dsmRCMsg is NULL     */

#define DSM_RC_NULL_OBJATTRPTR     2004 /* Object Attr Pointer is NULL      */

#define DSM_RC_NO_SESS_BLK         2006 /* no server session info           */
#define DSM_RC_NO_POLICY_BLK       2007 /* no policy hdr     info           */
#define DSM_RC_ZERO_BUFLEN         2008 /* bufferLen is zero for dataBlkPtr */
#define DSM_RC_NULL_BUFPTR         2009 /* bufferPtr is NULL for dataBlkPtr */

#define DSM_RC_INVALID_OBJTYPE     2010 /* invalid object type              */
#define DSM_RC_INVALID_VOTE        2011 /* invalid vote                     */
#define DSM_RC_INVALID_ACTION      2012 /* invalid action                   */
#define DSM_RC_INVALID_DS_HANDLE   2014 /* invalid ADSM handle              */
#define DSM_RC_INVALID_REPOS       2015 /* invalid value for repository     */
#define DSM_RC_INVALID_FSNAME      2016 /* fs should start with dir delim   */
#define DSM_RC_INVALID_OBJNAME     2017 /* invalid full path name           */
#define DSM_RC_INVALID_LLNAME      2018 /* ll should start with dir delim   */
#define DSM_RC_INVALID_OBJOWNER    2019 /* invalid object owner name        */
#define DSM_RC_INVALID_ACTYPE      2020 /* invalid action type              */
#define DSM_RC_INVALID_RETCODE     2021 /* dsmRC in dsmRCMsg is invalid     */
#define DSM_RC_INVALID_SENDTYPE    2022 /* invalid send type                */
#define DSM_RC_INVALID_PARAMETER   2023 /* invalid parameter                */
#define DSM_RC_INVALID_OBJSTATE    2024 /* active, inactive, or any match?  */
#define DSM_RC_INVALID_MCNAME      2025 /* Mgmt class name not found        */
#define DSM_RC_INVALID_DRIVE_CHAR  2026 /* Drive letter is not alphabet     */
#define DSM_RC_NULL_FSNAME         2027 /* Filespace name is NULL           */
#define DSM_RC_INVALID_HLNAME      2028 /* hl should start with dir delim   */

#define DSM_RC_NUMOBJ_EXCEED       2029 /* BeginGetData num objs exceeded   */

#define DSM_RC_NEWPW_REQD          2030 /* new password is required         */
#define DSM_RC_OLDPW_REQD          2031 /* old password is required         */
#define DSM_RC_NO_OWNER_REQD       2032 /* owner not allowed. Allow default */
#define DSM_RC_NO_NODE_REQD        2033 /* node not allowed w/ pw=generate  */
#define DSM_RC_KEY_MISSING         2034 /* key file can't be found          */
#define DSM_RC_KEY_BAD             2035 /* content of key file is bad       */

#define DSM_RC_BAD_CALL_SEQUENCE   2041 /* Sequence of DSM calls not allowed*/
#define DSM_RC_INVALID_TSMBUFFER   2042 /* invalid value for tsmbuffhandle or dataPtr */
#define DSM_RC_TOO_MANY_BYTES      2043 /* too many bytes copied to buffer  */
#define DSM_RC_MUST_RELEASE_BUFFER 2044 /* cant exit app needs to release buffers */
#define DSM_RC_BUFF_ARRAY_ERROR    2045 /* internal buff array error        */
#define DSM_RC_INVALID_DATABLK     2046 /* using tsmbuff datablk should be null */
#define DSM_RC_ENCR_NOT_ALLOWED    2047 /* when using tsmbuffers encription not allowed */
#define DSM_RC_OBJ_COMPRESSED      2048 /* Can't restore using tsmBuff on compressed object */
#define DSM_RC_OBJ_ENCRYPTED       2049 /* Cant restore using tsmbuff an encr obj */
#define DSM_RC_WILDCHAR_NOTALLOWED 2050 /* Wild card not allowed for hl,ll  */
#define DSM_RC_POR_NOT_ALLOWED     2051 /* Can't use partial object restore with tsmBuffers */
#define DSM_RC_NO_ENCRYPTION_KEY   2052 /* Encryption key not found*/
#define DSM_RC_ENCR_CONFLICT       2053 /* mutually exclusive options */

#define DSM_RC_FSNAME_NOTFOUND     2060 /* Filespace name not found         */
#define DSM_RC_FS_NOT_REGISTERED   2061 /* Filespace name not registered    */
#define DSM_RC_FS_ALREADY_REGED    2062 /* Filespace already registered     */
#define DSM_RC_OBJID_NOTFOUND      2063 /* No object id to restore          */
#define DSM_RC_WRONG_VERSION       2064 /* Wrong level of code              */
#define DSM_RC_WRONG_VERSION_PARM  2065 /* Wrong level of parameter struct  */

#define DSM_RC_NEEDTO_ENDTXN       2070 /* Need to call dsmEndTxn           */

#define DSM_RC_OBJ_EXCLUDED        2080 /* Object is excluded by MC         */
#define DSM_RC_OBJ_NOBCG           2081 /* Object has no backup copy group  */
#define DSM_RC_OBJ_NOACG           2082 /* Object has no archive copy group */

#define DSM_RC_APISYSTEM_ERROR     2090 /* API internal error               */

#define DSM_RC_DESC_TOOLONG             2100 /* description is too long          */
```

```
#define DSM_RC_OBJINFO_TOOLONG          2101 /* object attr objinfo too long     */
#define DSM_RC_HL_TOOLONG               2102 /* High level qualifier is too long */
#define DSM_RC_PASSWD_TOOLONG           2103 /* password is too long             */
#define DSM_RC_FILESPACE_TOOLONG        2104 /* filespace name is too long       */
#define DSM_RC_LL_TOOLONG               2105 /* Low level qualifier is too long  */
#define DSM_RC_FSINFO_TOOLONG           2106 /* filespace length is too big      */
#define DSM_RC_SENDDATA_WITH_ZERO_SIZE  2107 /* send data w/ zero est            */

/*=== new return codes for dsmaccess ===*/
#define DSM_RC_INVALID_ACCESS_TYPE 2110 /* invalid access type              */
#define DSM_RC_QUERY_COMM_FAILURE  2111 /* communication error during query */
#define DSM_RC_NO_FILES_BACKUP     2112 /* No backed up files for this fs   */
#define DSM_RC_NO_FILES_ARCHIVE    2113 /* No archived files for this fs    */
#define DSM_RC_INVALID_SETACCESS   2114 /* invalid set access format        */

/*=== new return codes for dsmaccess ===*/
#define DSM_RC_STRING_TOO_LONG     2120 /* String parameter too long        */

#define DSM_RC_MORE_DATA           2200 /* There are more data to restore   */

#define DSM_RC_BUFF_TOO_SMALL      2210 /* DataBlk buffer too small for qry */

#define DSM_RC_NO_API_CONFIGFILE   2228 /*specified API confg file not found*/
#define DSM_RC_NO_INCLEXCL_FILE    2229 /* specified inclexcl file not found*/
#define DSM_RC_NO_SYS_OR_INCLEXCL  2230 /* either dsm.sys or inclexcl file
                                           specified in dsm.sys not found   */
#define DSM_RC_REJECT_NO_POR_SUPPORT 2231 /* server doesn't have POR support*/

#define DSM_RC_NEED_ROOT           2300 /* API caller must be root          */
#define DSM_RC_NEEDTO_CALL_BINDMC  2301 /* dsmBindMC must be called first   */
#define DSM_RC_CHECK_REASON_CODE   2302 /* check reason code from dsmEndTxn */
#define DSM_RC_NEEDTO_ENDTXN_DEDUP_SIZE_EXCEEDED 2303 /* max dedup bytes exceeded */

/*=== return codes 2400 - 2410 used by lic file see agentrc.h ===*/

/*=== return codes 2410 - 2430 used by Oracle agent see agentrc.h ===*/

#define DSM_RC_ENC_WRONG_KEY       4580 /* the key provided is incorrect    */
#define DSM_RC_ENC_NOT_AUTHORIZED  4582 /* user is not allowed to decrypt   */
#define DSM_RC_ENC_TYPE_UNKNOWN    4584 /* encryption type unknown          */

/*============================================================================
   Return codes (4600)-(4624) are reserved for clustering
============================================================================*/
#define DSM_RC_CLUSTER_INFO_LIBRARY_NOT_LOADED        4600
#define DSM_RC_CLUSTER_LIBRARY_INVALID                4601
#define DSM_RC_CLUSTER_LIBRARY_NOT_LOADED             4602
#define DSM_RC_CLUSTER_NOT_MEMBER_OF_CLUSTER          4603
#define DSM_RC_CLUSTER_NOT_ENABLED                    4604
#define DSM_RC_CLUSTER_NOT_SUPPORTED                  4605
#define DSM_RC_CLUSTER_UNKNOWN_ERROR                  4606


/*============================================================================
   Return codes (5701)-(5749) are reserved for proxy
============================================================================*/
#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

/*============================================================================
   Return codes 5801 - 5849 are reserved for cryptography/security
============================================================================*/

#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

/*============================================================================
```

```
   Return codes 6300 - 6399 are reserved for client-side deduplication
================================================================================*/
#define DSM_RC_DIGEST_VALIDATION_ERROR               6300  /* End-to-end digest validation err  */
#define DSM_RC_DATA_FINGERPRINT_ERROR                6301 /* Failure in Rabin fingeprinting    */
#define DSM_RC_DATA_DEDUP_ERROR                      6302 /* Error converting data into chunks */


#endif /* _H_DSMRC */
```

Referencia relacionada:

Códigos de retorno de la API

Apéndice B. Archivos de origen de definiciones de tipo de API

Este apéndice contiene definiciones de estructura, definiciones de tipo y constantes de la API. El primer archivo de cabecera, `dsmapitd.h` y `tsmapitd.h`, ilustra las definiciones que son comunes a todos los sistemas operativos.

El segundo archivo de encabezado, `dsmapips.h`, proporciona un ejemplo de definiciones que son específicas a un sistema operativo particular; en este ejemplo, la plataforma Windows.

El tercer archivo, `release.h`, incluye la información de versión y release.

La información que se proporciona aquí contiene una copia puntual de los archivos que se distribuyen con la API. Vea los archivos en el paquete de distribución de la API para la última versión.

```
/*************************************************************************
* Tivoli Storage Manager                                                 *
* API Client Component                                                   *
*                                                                        *
* (C) Copyright IBM Corporation 1993,2010                                *
*************************************************************************/

/*************************************************************************
* Header File Name: dsmapitd.h
*
* Environment:     ***********************************************
*                  ** This is a platform-independent source file **
*
*                  ***********************************************
*
* Design Notes:    This file contains basic data types and constants
*                  includable by all client source files. The constants
*                  within this file should be set properly for the
*                  particular machine and operating system on which the
*                  client software is to be run.
*
*                  Platform specific definitions are included in dsmapips.h
*
* Descriptive-name: Definitions for Tivoli Storage manager API constants
*------------------------------------------------------------------------*/

#ifndef _H_DSMAPITD
#define _H_DSMAPITD


#include "dsmapips.h"      /* Platform specific definitions*/
#include "release.h"

/*=== set the structure alignment to pack the structures ===*/
#if (_OPSYS_TYPE == DS_WINNT) && !defined(_WIN64)
#pragma pack(1)
#endif

#ifdef _MAC
/*=========================================================================
  choices are:
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                                      */
/*<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>*/
/*-------------------------------------------------------------------------+
|  API Version, Release, and Level to use in dsmApiVersion on dsmInit()    |
+-------------------------------------------------------------------------*/
#define DSM_API_VERSION     COMMON_VERSION
#define DSM_API_RELEASE     COMMON_RELEASE
#define DSM_API_LEVEL       COMMON_LEVEL
#define DSM_API_SUBLEVEL    COMMON_SUBLEVEL

/*-------------------------------------------------------------------------+
| Maximum field lengths                                                    |
+-------------------------------------------------------------------------*/
#define DSM_MAX_CG_DEST_LENGTH        30        /* copy group destination */
#define DSM_MAX_CG_NAME_LENGTH        30        /* copy group name        */
#define DSM_MAX_DESCR_LENGTH          255       /* archive description    */
#define DSM_MAX_DOMAIN_LENGTH         30        /* policy domain name     */
#define DSM_MAX_FSINFO_LENGTH         500       /* filespace info         */
#define DSM_MAX_USER_FSINFO_LENGTH    480       /* max user filespace info*/
#define DSM_MAX_FSNAME_LENGTH         1024      /* filespace name         */
#define DSM_MAX_FSTYPE_LENGTH         32        /* filespace type         */
#define DSM_MAX_HL_LENGTH             1024      /* object high level name */
#define DSM_MAX_ID_LENGTH             64        /* session node name      */
#define DSM_MAX_LL_LENGTH             256       /* object low level name  */
#define DSM_MAX_MC_NAME_LENGTH        30        /* management class name  */
#define DSM_MAX_OBJINFO_LENGTH        255       /* object info            */
#define DSM_MAX_EXT_OBJINFO_LENGTH    1500      /* Extended object info   */
#define DSM_MAX_OWNER_LENGTH          64        /* object owner name      */
#define DSM_MAX_PLATFORM_LENGTH       16        /* application type       */
#define DSM_MAX_PS_NAME_LENGTH        30        /* policy set name        */
#define DSM_MAX_SERVERTYPE_LENGTH     32        /* server platform type   */
#define DSM_MAX_VERIFIER_LENGTH       64        /* password               */
#define DSM_PATH_MAX                  1024      /* API config file path   */
#define DSM_NAME_MAX                  255       /* API config file name   */
#define DSM_MAX_NODE_LENGTH           64        /* node/machine name      */
#define DSM_MAX_RC_MSG_LENGTH         1024      /* msg parm for dsmRCMsg  */
#define DSM_MAX_SERVER_ADDRESS        1024     /* server address */

#define DSM_MAX_MC_DESCR_LENGTH       DSM_MAX_DESCR_LENGTH /* mgmt class  */
#define DSM_MAX_SERVERNAME_LENGTH     DSM_MAX_ID_LENGTH /* server name    */
#define DSM_MAX_GET_OBJ               4080     /* max objs on BeginGetData */
#define DSM_MAX_PARTIAL_GET_OBJ       1300 /* max partial objs on BeginGetData*/
#define DSM_MAX_COMPRESSTYPE_LENGTH   32       /* max compression algorithm name */

/*-------------------------------------------------------------------------+
| Minimum field lengths                                                    |
+-------------------------------------------------------------------------*/
#define DSM_MIN_COMPRESS_SIZE  2048 /* minimum number of bytes an object  */
                                    /* needs before compression is allowed*/

/*-------------------------------------------------------------------------+
|  Values for mtFlag in dsmSetup call                                      |
+-------------------------------------------------------------------------*/
#define DSM_MULTITHREAD      bTrue
#define DSM_SINGLETHREAD     bFalse

/*-------------------------------------------------------------------------+
|  Values for object type in dsmObjName structure                          |
|  Note: These values must be kept in sync with dsmcomm.h                  |
+-------------------------------------------------------------------------*/
#define DSM_OBJ_FILE                  0x01 /*object has attrib info & data*/
#define DSM_OBJ_DIRECTORY             0x02 /*obj has only attribute info  */
#define DSM_OBJ_RESERVED1             0x04 /* for future use              */
#define DSM_OBJ_RESERVED2             0x05 /* for future use              */
```

```
#define DSM_OBJ_RESERVED3            0x06 /* for future use             */
#define DSM_OBJ_WILDCARD             0xFE /* Any object type            */
#define DSM_OBJ_ANY_TYPE             0xFF /* for future use             */

/*-------------------------------------------------------------------------+
| Type definition for compressedState in QryResp                           |
+-------------------------------------------------------------------------*/
#define DSM_OBJ_COMPRESSED_UNKNOWN    0
#define DSM_OBJ_COMPRESSED_YES        1
#define DSM_OBJ_COMPRESSED_NO         2

/*-------------------------------------------------------------------------+
| Definitions for "group type" field in tsmGrouphandlerIn_t                |
+-------------------------------------------------------------------------*/

#define  DSM_GROUPTYPE_NONE         0x00   /* Not a group member      */
#define  DSM_GROUPTYPE_RESERVED1    0x01   /* for future use          */
#define  DSM_GROUPTYPE_PEER         0x02   /* Peer group              */
#define  DSM_GROUPTYPE_RESERVED2    0x03   /* for future use          */

/*-------------------------------------------------------------------------+
| Definitions for "member type" field in tsmGrouphandlerIn_t               |
+-------------------------------------------------------------------------*/

#define  DSM_MEMBERTYPE_LEADER      0x01   /* group leader  */
#define  DSM_MEMBERTYPE_MEMBER      0x02   /* group member  */

/*-------------------------------------------------------------------------+
| Definitions for "operation type" field in tsmGrouphandlerIn_t            |
+-------------------------------------------------------------------------*/
#define DSM_GROUP_ACTION_BEGIN         0x01
#define DSM_GROUP_ACTION_OPEN          0x02 /* create new group             */
#define DSM_GROUP_ACTION_CLOSE         0x03 /* commit and save an open group */
#define DSM_GROUP_ACTION_ADD           0x04 /* Append to a group */
#define DSM_GROUP_ACTION_ASSIGNTO      0x05 /* Assign to a another group */
#define DSM_GROUP_ACTION_REMOVE        0x06 /* remove a member from a group */

/*-------------------------------------------------------------------------+
| Values for copySer in DetailCG structures for Query Mgmt Class response  |
+-------------------------------------------------------------------------*/
#define Copy_Serial_Static          1   /*Copy Serialization Static       */
#define Copy_Serial_Shared_Static   2   /*Copy Serialization Shared Static*/
#define Copy_Serial_Shared_Dynamic  3   /*Copy Serialization Shared Dynamic*/
#define Copy_Serial_Dynamic         4   /*Copy Serialization Dynamic      */

/*-------------------------------------------------------------------------+
| Values for copyMode in DetailCG structures for Query Mgmt Class response |
+-------------------------------------------------------------------------*/
#define Copy_Mode_Modified          1   /*Copy Mode Modified              */
#define Copy_Mode_Absolute          2   /*Copy Mode Absolute              */

/*-------------------------------------------------------------------------+
|  Values for objState in qryBackupData structure                          |
+-------------------------------------------------------------------------*/
#define DSM_ACTIVE            0x01      /* query only active objects      */
#define DSM_INACTIVE          0x02      /* query only inactive objects    */
#define DSM_ANY_MATCH         0xFF      /* query all backup objects       */

/*-------------------------------------------------------------------------+
| Boundary values for dsmDate.year field in qryArchiveData structure       |
+-------------------------------------------------------------------------*/
#define DATE_MINUS_INFINITE         0x0000      /* lowest boundary        */
#define DATE_PLUS_INFINITE          0xFFFF      /* highest upper boundary */

/*-------------------------------------------------------------------------+
| Bits masks for update action parameter on dsmUpdateFS()                  |
+-------------------------------------------------------------------------*/
#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)

/*-------------------------------------------------------------------------+
| Bits mask  for backup update action parameter on dsmUpdateObj()          |
+-------------------------------------------------------------------------*/
#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)

/*-------------------------------------------------------------------------+
|  Values for repository parameter on dsmDeleteFS()                        |
+-------------------------------------------------------------------------*/
#define DSM_ARCHIVE_REP       0x0A      /* archive repository             */
#define DSM_BACKUP_REP        0x0B      /* backup repository              */
#define DSM_REPOS_ALL         0x01      /* all respository types          */

/*-------------------------------------------------------------------------+
|  Values for vote parameter on dsmEndTxn()                                |
+-------------------------------------------------------------------------*/
#define DSM_VOTE_COMMIT  1             /* commit current transaction      */
#define DSM_VOTE_ABORT   2             /* roll back current transaction   */

/*-------------------------------------------------------------------------+
|  Values for various flags returned in ApiSessInfo structure.             |
+-------------------------------------------------------------------------*/
/* Client compression field codes */
#define COMPRESS_YES    1    /* client must compress data             */
#define COMPRESS_NO     2    /* client must NOT compress data         */
#define COMPRESS_CD     3    /* client determined                     */

/* Archive delete permission codes. */
#define ARCHDEL_YES     1    /* archive delete allowed                */
#define ARCHDEL_NO      2    /* archive delete NOT allowed            */

/* Backup delete permission codes. */
#define BACKDEL_YES     1    /* backup delete allowed                 */
#define BACKDEL_NO      2    /* backup delete NOT allowed             */


/*-------------------------------------------------------------------------+
  Values for various flags returned in optStruct structure.              |
--------------------------------------------------------------------------*/
#define DSM_PASSWD_GENERATE  1
#define DSM_PASSWD_PROMPT    0

#define DSM_COMM_TCP        1    /* tcpip         */
#define DSM_COMM_NAMEDPIPE  2    /* Named pipes   */
#define DSM_COMM_SHM        3    /* Shared Memory */

/* obsolete 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    /* For IPX/SPX support */
#define DSM_COMM_NETBIOS    12    /* NETBIOS */
#define DSM_COMM_400COMM    12
#define DSM_COMM_CLIO       12    /* CLIO/S */
/*-------------------------------------------------------------------------+
|  Values for userNameAuthorities in dsmInitEx for future use              |
+-------------------------------------------------------------------------*/
#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)

/*-------------------------------------------------------------------------+
|  Values for encryptionType on dsmEndSendObjEx, 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)

/*-------------------------------------------------------------------------+
| Definitions for mediaClass field.                                        |
+-------------------------------------------------------------------------*/
/*
 *  The following constants define a hierarchy of media access classes.
 *  Lower numbers indicate media which can supply faster access to data.
 */

/* Fixed: represents the class of on-line, fixed media (such as
          hard disks). */
#define  MEDIA_FIXED          0x10

/* Library: represents the class of mountable media accessible
            through a mechanical mounting device. */
#define  MEDIA_LIBRARY        0x20

/* future use */
#define  MEDIA_NETWORK        0x30

/* future use */
#define  MEDIA_SHELF          0x40

/* future use */
#define  MEDIA_OFFSITE        0x50

/* future use */
#define  MEDIA_UNAVAILABLE    0xF0


/*-------------------------------------------------------------------------+
| Type definition for partial object data for dsmBeginGetData()            |
+-------------------------------------------------------------------------*/
typedef struct
{
    dsUint16_t stVersion;     /* Structure version               */
    dsStruct64_t   partialObjOffset;   /* offset into object to begin reading*/
    dsStruct64_t   partialObjLength;   /* amount of object to read          */
} PartialObjData ;               /* partial object data                  */

#define PartialObjDataVersion 1  /*                                      */

/*-------------------------------------------------------------------------+
|  Type definition for date structure                                      |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t    year;                /* year, 16-bit integer (e.g., 1990) */
   dsUint8_t     month;               /* month, 8-bit integer (1 - 12)     */
   dsUint8_t     day;                 /* day. 8-bit integer (1 - 31)       */
   dsUint8_t     hour;                /* hour, 8-bit integer (0 - 23)      */
   dsUint8_t     minute;              /* minute, 8-bit integer (0 - 59)    */
   dsUint8_t     second;              /* second, b-bit integer (0 - 59)    */
}dsmDate ;

/*-------------------------------------------------------------------------+
```

```
|  Type definition for Object ID on dsmGetObj() and in dsmGetList structure|
+-------------------------------------------------------------------------*/
typedef dsStruct64_t  ObjID ;

/*-------------------------------------------------------------------------+
| Type definition for dsmQueryBuff on dsmBeginQuery()                      |
+-------------------------------------------------------------------------*/
typedef void dsmQueryBuff ;

/*-------------------------------------------------------------------------+
| Type definition for dsmGetType parameter on dsmBeginGetData()            |
+-------------------------------------------------------------------------*/
typedef enum
{
        gtBackup = 0x00,                    /* Backup processing type    */
        gtArchive                           /* Archive processing type   */
} dsmGetType ;

/*-------------------------------------------------------------------------+
|  Type definition for dsmQueryType parameter on dsmBeginQuery()           |
+-------------------------------------------------------------------------*/
typedef enum
{
   qtArchive = 0x00,                /* Archive query type                */
   qtBackup,                        /* Backup query type                 */
   qtBackupActive,                  /* Fast query for active backup files */
   qtFilespace,                     /* Filespace query type              */
   qtMC,                            /* Mgmt. class query type            */
   qtReserved1,                     /* future use                        */
   qtReserved2,                     /* future use                        */
   qtReserved3,                     /* future use                        */
   qtReserved4,                     /* future use                        */
   qtBackupGroups,                  /* group leaders in a specific fs    */
   qtOpenGroups,                    /* Open groups in a specific fs      */
   qtReserved5,                     /* future use                        */
   qtProxyNodeAuth,                 /* nodes that his node can proxy to  */
   qtProxyNodePeer,                 /* Peer nodes with the same target   */
   qtReserved6,                     /* future use                        */
   qtReserved7,                     /* future use                        */
   qtReserved8                      /* future use                        */
}dsmQueryType ;

/*-------------------------------------------------------------------------+
|  Type definition sendType parameter on dsmBindMC() and dsmSendObj()      |
+-------------------------------------------------------------------------*/
typedef enum
{
   stBackup = 0x00,                        /* Backup processing type    */
   stArchive,                              /* Archive processing type   */
   stBackupMountWait,           /* Backup processing with mountwait on   */
   stArchiveMountWait           /* Archive processing with mountwait on  */
}dsmSendType ;

/*-------------------------------------------------------------------------+
|  Type definition for delType parameter on dsmDeleteObj()                 |
+-------------------------------------------------------------------------*/
typedef enum
{
   dtArchive = 0x00,                             /* Archive delete type */
   dtBackup,                          /* Backup delete (deactivate) type */
   dtBackupID                        /* Backup delete (remove)     type */
}dsmDelType ;

/*-------------------------------------------------------------------------+
|  Type definition sendType parameter on dsmSetAccess()                    |
+-------------------------------------------------------------------------*/
typedef enum
{
   atBackup = 0x00,                        /* Backup processing type    */
   atArchive                              /* Archive processing type   */
```

```
}dsmAccessType;

/*-------------------------------------------------------------------------+
|  Type definition for API Version on dsmInit() and dsmQueryApiVersion()   |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t version;       /* API version                     */
   dsUint16_t release;       /* API release                     */
   dsUint16_t level;         /* API level                       */
}dsmApiVersion;

/*-------------------------------------------------------------------------+
|  Type definition for API Version on dsmInit() and dsmQueryApiVersion()   |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t     stVersion;           /* Structure version               */
   dsUint16_t version;       /* API version                     */
   dsUint16_t release;       /* API release                     */
   dsUint16_t level;         /* API level                       */
   dsUint16_t subLevel;      /* API sub level                   */
   dsmBool_t  unicode;       /* API unicode?                    */
}dsmApiVersionEx;


#define apiVersionExVer       2

/*-------------------------------------------------------------------------+
|  Type definition for Application Version on dsmInit()                    |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t     stVersion;           /* Structure version               */
   dsUint16_t   applicationVersion;   /* application version number   */
   dsUint16_t   applicationRelease;   /* application release number   */
   dsUint16_t   applicationLevel;     /* application level number     */
   dsUint16_t   applicationSubLevel;  /* application sub level number */
} dsmAppVersion;

#define appVersionVer      1

/*-------------------------------------------------------------------------+
|  Type definition for object name used on BindMC, Send, Delete, Query     |
+-------------------------------------------------------------------------*/
typedef struct S_dsmObjName
{
   char       fs[DSM_MAX_FSNAME_LENGTH + 1] ;              /* Filespace name */
   char       hl[DSM_MAX_HL_LENGTH + 1] ;                 /* High level name */
   char       ll[DSM_MAX_LL_LENGTH + 1] ;                  /* Low level name */
   dsUint8_t  objType;          /* for object type values, see defines above */
}dsmObjName;


/*-------------------------------------------------------------------------+
|  Type definition for Backup delete info on dsmDeleteObj()                |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t       stVersion ;                   /* structure version      */
   dsmObjName       *objNameP ;                   /* object name            */
   dsUint32_t       copyGroup ;                   /* copy group             */
}delBack ;

#define  delBackVersion    1

/*-------------------------------------------------------------------------+
|  Type definition for Archive delete info on dsmDeleteObj()               |
+-------------------------------------------------------------------------*/
typedef struct
```

```
{
   dsUint16_t       stVersion ;                    /* structure version      */
   dsStruct64_t       objId ;                        /* object ID              */
}delArch ;

#define  delArchVersion  1

/*-------------------------------------------------------------------------+
|  Type definition for Backup ID delete info on dsmDeleteObj()             |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t       stVersion ;                    /* structure version      */
   dsStruct64_t       objId ;                        /* object ID              */
}delBackID;

#define  delBackIDVersion  1

/*-------------------------------------------------------------------------+
|  Type definition for delete info on dsmDeleteObj()                       |
+-------------------------------------------------------------------------*/
typedef union
{
   delBack   backInfo ;
   delArch   archInfo ;
   delBackID backIDInfo ;
}dsmDelInfo ;

/*-------------------------------------------------------------------------+
|  Type definition for Object Attribute parameter on dsmSendObj()          |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t stVersion;     /* Structure version               */
   char         owner[DSM_MAX_OWNER_LENGTH + 1];      /* object owner */
   dsStruct64_t sizeEstimate;             /* Size estimate in bytes of the object */
   dsmBool_t    objCompressed;            /* Is object already compressed? */
   dsUint16_t   objInfoLength;            /* length of object-dependent info   */
   char         *objInfo;      /* object-dependent info */
   char         *mcNameP;                /* mgmnt class name for override  */
   dsmBool_t    disableDeduplication;         /* force no dedup for this object    */
   dsmBool_t    useExtObjInfo;                   /* use ext obj info up to 1536 */
}ObjAttr;

#define ObjAttrVersion  4



/*-------------------------------------------------------------------------+
| Type definition for mcBindKey returned on dsmBindMC()                   |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t      stVersion;            /* structure version   */
   char        mcName[DSM_MAX_MC_NAME_LENGTH + 1];
                                       /* Name of mc bound to object. */
   dsmBool_t   backup_cg_exists;                           /* True/false */
   dsmBool_t   archive_cg_exists;                          /* True/false */
   char        backup_copy_dest[DSM_MAX_CG_DEST_LENGTH + 1];
                                           /* Backup copy dest. name */
   char        archive_copy_dest[DSM_MAX_CG_DEST_LENGTH + 1];
                                               /* Arch copy dest.name */
}mcBindKey;

#define mcBindKeyVersion  1


/*-------------------------------------------------------------------------+
| Type definition for object list on dsmBeginGetData()                     |
+-------------------------------------------------------------------------*/
```

```
typedef struct
{
   dsUint16_t       stVersion ;                    /* structure version      */
   dsUint32_t       numObjId ;          /* number of object IDs in the list */
   ObjID           *objId ;             /* list of object IDs to restore*/
   PartialObjData  *partialObjData;     /*list of partial obj data info */
}dsmGetList ;

#define dsmGetListVersion     2  /* default if not using Partial Obj data */
#define dsmGetListPORVersion  3  /* version if using Partial Obj data     */

/*-------------------------------------------------------------------------+
|  Type definition for DataBlk used to Get or Send data                    |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t       stVersion ;                    /* structure version      */
   dsUint32_t bufferLen;                /* Length of buffer passed below */
   dsUint32_t numBytes;                 /* Actual number of bytes read from */
                                        /* or written to the buffer */
   char       *bufferPtr;               /* Data buffer */
   dsUint32_t numBytesCompressed;       /* on send actual bytes compressed */
   dsUint16_t reserved;                 /* for future use                  */
}DataBlk;

#define DataBlkVersion  3


/*-------------------------------------------------------------------------+
|  Type definition for Mgmt Class queryBuffer on dsmBeginQuery()           |
+-------------------------------------------------------------------------*/
typedef struct S_qryMCData
{
   dsUint16_t      stVersion;           /* structure version   */
   char         *mcName;                      /* Mgmt class name */
                     /* single name to get one or empty string to get all*/
   dsmBool_t    mcDetail;                          /* Want details or not? */
}qryMCData;

#define qryMCDataVersion  1

/*=== values for RETINIT ===*/
#define ARCH_RETINIT_CREATE 0
#define ARCH_RETINIT_EVENT  1

/*-------------------------------------------------------------------------+
|  Type definition for Archive Copy Group details on Query MC response     |
+-------------------------------------------------------------------------*/
typedef struct S_archDetailCG
{
   char         cgName[DSM_MAX_CG_NAME_LENGTH + 1];       /* Copy group name */
   dsUint16_t   frequency;                     /* Copy (archive) frequency */
   dsUint16_t   retainVers;                           /* Retain version */
   dsUint8_t    copySer;       /* for copy serialization values, see defines */
   dsUint8_t    copyMode;       /* for copy mode values, see defines above */
   char         destName[DSM_MAX_CG_DEST_LENGTH + 1];      /* Copy dest name */
   dsmBool_t    bLanFreeDest;           /* Destination has lan free path?   */
   dsmBool_t    reserved;                         /* Not currently used */
   dsUint8_t    retainInit;             /* possible values see above        */
   dsUint16_t   retainMin;             /* if retInit is EVENT num of days */
   dsmBool_t    bDeduplicate;           /* destination has dedup enabled    */
}archDetailCG;

/*-------------------------------------------------------------------------+
|  Type definition for Backup Copy Group details on Query MC response      |
+-------------------------------------------------------------------------*/
typedef struct S_backupDetailCG
{
   char         cgName[DSM_MAX_CG_NAME_LENGTH + 1];       /* Copy group name */
   dsUint16_t   frequency;                               /* Backup frequency */
```

```
   dsUint16_t   verDataExst;                        /* Versions data exists */
   dsUint16_t   verDataDltd;                       /* Versions data deleted */
   dsUint16_t   retXtraVers;                       /* Retain extra versions */
   dsUint16_t   retOnlyVers;                        /* Retain only versions */
   dsUint8_t    copySer;       /* for copy serialization values, see defines */
   dsUint8_t    copyMode;       /* for copy mode values, see defines above */
   char         destName[DSM_MAX_CG_DEST_LENGTH + 1];      /* Copy dest name */
   dsmBool_t    bLanFreeDest;           /* Destination has lan free path?   */
   dsmBool_t    reserved;                          /* Not currently used */
   dsmBool_t    bDeduplicate;            /* destination has dedup enabled    */
}backupDetailCG;


/*-------------------------------------------------------------------------+
|  Type definition for Query Mgmt Class detail response on dsmGetNextQObj()|
+-------------------------------------------------------------------------*/
typedef struct S_qryRespMCDetailData
{
   dsUint16_t      stVersion;            /* structure version   */
   char            mcName[DSM_MAX_MC_NAME_LENGTH + 1];            /* mc name */
   char            mcDesc[DSM_MAX_MC_DESCR_LENGTH + 1]; /*mc description */
   archDetailCG    archDet;                /* Archive copy group detail */
   backupDetailCG  backupDet;               /* Backup copy group detail */
}qryRespMCDetailData;

#define qryRespMCDetailDataVersion  4

/*-------------------------------------------------------------------------+
| Type definition for Query Mgmt Class summary response on dsmGetNextQObj()|
+-------------------------------------------------------------------------*/
typedef struct S_qryRespMCData
{
   dsUint16_t      stVersion;            /* structure version   */
   char            mcName[DSM_MAX_MC_NAME_LENGTH + 1];            /* mc name */
   char         mcDesc[DSM_MAX_MC_DESCR_LENGTH + 1];       /* mc description */
}qryRespMCData;

#define qryRespMCDataVersion  1



/*-------------------------------------------------------------------------+
|  Type definition for Archive queryBuffer on dsmBeginQuery()              |
+-------------------------------------------------------------------------*/
typedef struct S_qryArchiveData
{
   dsUint16_t      stVersion;            /* structure version   */
   dsmObjName   *objName;                    /* Full dsm name of object */
   char        *owner;            /* owner name */
                  /* for maximum date boundaries, see defines above */
   dsmDate      insDateLowerBound;      /* low bound archive insert date */
   dsmDate      insDateUpperBound;       /* hi bound archive insert date */
   dsmDate      expDateLowerBound;          /* low bound expiration date */
   dsmDate      expDateUpperBound;           /* hi bound expiration date */
   char         *descr;                   /* archive description */
} qryArchiveData;

#define qryArchiveDataVersion  1



/*=== values for retentionInitiated field ===*/
#define DSM_ARCH_RETINIT_UNKNOWN 0 /* ret init is unknown (down-level srv)  */
#define DSM_ARCH_RETINIT_STARTED 1 /* retention clock is started            */
#define DSM_ARCH_RETINIT_PENDING 2 /* retention clock is not started        */


/*=== Values for objHeld ===*/
#define DSM_ARCH_HELD_UNKNOWN 0    /* unknown hold status (down-level srv)  */
#define DSM_ARCH_HELD_FALSE   1    /* object is NOT in a delete hold state  */
```

```
#define DSM_ARCH_HELD_TRUE    2    /* object is in a delete hold state      */

/*-------------------------------------------------------------------------+
| Type definition for Query Archive response on dsmGetNextQObj()           |
+-------------------------------------------------------------------------*/
typedef struct S_qryRespArchiveData
{
   dsUint16_t      stVersion;           /* structure version   */
   dsmObjName      objName;                    /* Filespace name qualifier */
   dsUint32_t  copyGroup;                       /* copy group number */
   char            mcName[DSM_MAX_MC_NAME_LENGTH + 1];           /* mc name */
   char            owner[DSM_MAX_OWNER_LENGTH + 1];           /* owner name */
   dsStruct64_t    objId;                                  /* Unique copy id */
   dsStruct64_t    reserved;                      /* backward compatability */
   dsUint8_t       mediaClass;                        /* media access class */
   dsmDate         insDate;                       /* archive insertion date */
   dsmDate         expDate;                   /* expiration date for object */
   char            descr[DSM_MAX_DESCR_LENGTH + 1];   /* archive description */
   dsUint16_t      objInfolen;            /* length of object-dependent info*/
   char            reservedObjInfo[DSM_MAX_OBJINFO_LENGTH];  /*object-dependent info */
   dsUint160_t     restoreOrderExt;                         /* restore order */
   dsStruct64_t    sizeEstimate;             /* size estimate stored by user*/
   dsUint8_t       compressType;                          /* Compression flag*/
   dsUint8_t       retentionInitiated;   /* object waiting on retention event*/
   dsUint8_t       objHeld;  /*object is on retention "hold" see values above*/
   dsUint8_t       encryptionType;                    /* type of encryption */
   dsmBool_t       clientDeduplicated;             /* obj deduplicated by API*/
   char            objInfo[DSM_MAX_EXT_OBJINFO_LENGTH];  /*object-dependent info */
   char            compressAlg[DSM_MAX_COMPRESSTYPE_LENGTH + 1];  /* compression algorithm name */
}qryRespArchiveData;

#define qryRespArchiveDataVersion  7

/*-------------------------------------------------------------------------+
|  Type definition for Archive sendBuff parameter on dsmSendObj()          |
+-------------------------------------------------------------------------*/
typedef struct S_sndArchiveData
{
   dsUint16_t      stVersion;           /* structure version   */
   char         *descr;                   /* archive description */
}sndArchiveData;

#define sndArchiveDataVersion  1

/*-------------------------------------------------------------------------+
|  Type definition for Backup queryBuffer on dsmBeginQuery()               |
+-------------------------------------------------------------------------*/
typedef struct S_qryBackupData
{
   dsUint16_t      stVersion;           /* structure version   */
   dsmObjName  *objName;          /* full dsm name of object */
   char        *owner;            /* owner name */
   dsUint8_t   objState;          /* object state selector */
   dsmDate     pitDate;           /* Date value for point in time restore */
                                  /* for possible values, see defines above */
}qryBackupData;

#define qryBackupDataVersion  2

typedef struct
{
  dsUint8_t    reserved1;
  dsStruct64_t reserved2;
} reservedInfo_t;                /* for future use */

/*-------------------------------------------------------------------------+
| Type definition for Query Backup response on dsmGetNextQObj()            |
+-------------------------------------------------------------------------*/
typedef struct S_qryRespBackupData
{
```

```
   dsUint16_t      stVersion;            /* structure version   */
   dsmObjName  objName;                       /* full dsm name of object */
   dsUint32_t  copyGroup;                           /* copy group number */
   char            mcName[DSM_MAX_MC_NAME_LENGTH + 1];            /* mc name */
   char            owner[DSM_MAX_OWNER_LENGTH + 1];            /* owner name */
   dsStruct64_t    objId;                                /* Unique object id */
   dsStruct64_t    reserved;                         /* backward compatability */
   dsUint8_t       mediaClass;                           /* media access class */
   dsUint8_t       objState;                        /* Obj state, active, etc. */
   dsmDate     insDate;                          /* backup insertion date */
   dsmDate         expDate;                     /* expiration date for object */
   dsUint16_t      objInfolen;             /* length of object-dependent info*/
   char            reservedObjInfo[DSM_MAX_OBJINFO_LENGTH];  /*object-dependent info */
   dsUint160_t     restoreOrderExt;                          /* restore order */
   dsStruct64_t    sizeEstimate;             /* size estimate stored by user */
   dsStruct64_t    baseObjId;
   dsUint16_t      baseObjInfolen;             /* length of base object-dependent info*/
   dsUint8_t       baseObjInfo[DSM_MAX_OBJINFO_LENGTH];  /* base object-dependent info */
   dsUint160_t     baseRestoreOrder;                          /* restore order */
   dsUint32_t     fsID;
   dsUint8_t       compressType;
   dsmBool_t       isGroupLeader;
   dsmBool_t       isOpenGroup;
   dsUint8_t       reserved1;               /* for future use */
   dsmBool_t       reserved2;               /* for future use */
   dsUint16_t      reserved3;               /* for future use */
   reservedInfo_t  *reserved4;              /* for future use */
   dsUint8_t       encryptionType;                    /* type of encryption */
   dsmBool_t       clientDeduplicated;             /* obj deduplicated by API*/
   char            objInfo[DSM_MAX_EXT_OBJINFO_LENGTH];  /*object-dependent info */
   char            compressAlg[DSM_MAX_COMPRESSTYPE_LENGTH + 1];  /* compression algorithm name */
}qryRespBackupData;

#define qryRespBackupDataVersion  8

/*------------------------------------------------------------------------+
|  Type definition for Active Backup queryBuffer on dsmBeginQuery()
|
|  Notes:  For the active backup query, only the fs (filespace) and objType
|          fields of objName need be set.  objType can only be set to
|          DSM_OBJ_FILE or DSM_OBJ_DIRECTORY.  DSM_OBJ_ANY_TYPE will not
|          find a match on the query.
+------------------------------------------------------------------------*/
typedef struct S_qryABackupData
{
   dsUint16_t      stVersion;            /* structure version   */
   dsmObjName      *objName;                    /* Only fs and objtype used */
}qryABackupData;

#define qryABackupDataVersion  1

/*------------------------------------------------------------------------+
| Type definition for Query Active Backup response on dsmGetNextQObj()     |
+------------------------------------------------------------------------*/
typedef struct S_qryARespBackupData
{
   dsUint16_t      stVersion;            /* structure version   */
   dsmObjName  objName;                       /* full dsm name of object */
   dsUint32_t  copyGroup;                           /* copy group number */
   char        mcName[DSM_MAX_MC_NAME_LENGTH + 1];/*management class name*/
   char            owner[DSM_MAX_OWNER_LENGTH + 1];            /* owner name */
   dsmDate     insDate;                           /* backup insertion date */
   dsUint16_t      objInfolen;             /* length of object-dependent info*/
   char        reservedObjInfo[DSM_MAX_OBJINFO_LENGTH];  /*object-dependent info */
   char        objInfo[DSM_MAX_EXT_OBJINFO_LENGTH];  /*object-dependent info */
}qryARespBackupData;

#define qryARespBackupDataVersion  2

/*------------------------------------------------------------------------+
```

```
|  Type definition for Backup queryBuffer on dsmBeginQuery()               |
+--------------------------------------------------------------------------*/
typedef struct qryBackupGroups
{
   dsUint16_t      stVersion;           /* structure version   */
   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


/*--------------------------------------------------------------------------+
|  Type definition for proxynode queryBuffer on dsmBeginQuery()             |
+--------------------------------------------------------------------------*/
typedef struct qryProxyNodeData
{
   dsUint16_t      stVersion;           /* structure version   */
   char        *targetNodeName;            /* target node name  */
}qryProxyNodeData;

#define qryProxyNodeDataVersion  1

/*--------------------------------------------------------------------------+
| Type definition for qryRespProxyNodeData  parameter used on dsmGetNextQObj()|
+--------------------------------------------------------------------------*/

typedef struct
{
   dsUint16_t       stVersion ;                   /* structure version      */
   char            targetNodeName[DSM_MAX_ID_LENGTH+1]; /* target node name  */
   char            peerNodeName[DSM_MAX_ID_LENGTH+1];   /* Peer node name    */
   char            hlAddress[DSM_MAX_ID_LENGTH+1];      /* peer hlAddress    */
   char            llAddress[DSM_MAX_ID_LENGTH+1];      /* peer hlAddress    */
}qryRespProxyNodeData;

#define qryRespProxyNodeDataVersion  1


/*--------------------------------------------------------------------------+
|  Type definition for WINNT and OS/2 Filespace attributes                  |
+--------------------------------------------------------------------------*/
typedef struct
{
   char         driveLetter ;          /* drive letter for filespace    */
   dsUint16_t   fsInfoLength;          /* fsInfo length used            */
   char         fsInfo[DSM_MAX_FSINFO_LENGTH];/*caller-determined data  */
}dsmDosFSAttrib ;

/*--------------------------------------------------------------------------+
|  Type definition for UNIX Filespace attributes                            |
+--------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t   fsInfoLength;          /* fsInfo length used            */
   char         fsInfo[DSM_MAX_FSINFO_LENGTH];/*caller-determined data  */
}dsmUnixFSAttrib ;

/*--------------------------------------------------------------------------+
|  Type definition for NetWare Filespace attributes                         |
+--------------------------------------------------------------------------*/
typedef dsmUnixFSAttrib dsmNetwareFSAttrib;

/*--------------------------------------------------------------------------+
```

```
|  Type definition for Filespace attributes on all Filespace calls         |
+--------------------------------------------------------------------------*/
typedef union
{
   dsmNetwareFSAttrib  netwareFSAttr;
   dsmUnixFSAttrib     unixFSAttr ;
   dsmDosFSAttrib      dosFSAttr ;
}dsmFSAttr ;

/*--------------------------------------------------------------------------+
|  Type definition for fsUpd parameter on dsmUpdateFS()
+--------------------------------------------------------------------------*/
typedef struct S_dsmFSUpd
{
   dsUint16_t       stVersion ;                    /* structure version      */
   char            *fsType ;              /* filespace type                 */
   dsStruct64_t    occupancy ;            /* occupancy estimate             */
   dsStruct64_t    capacity  ;            /* capacity estimate              */
   dsmFSAttr       fsAttr ;                   /* platform specific attributes */
}dsmFSUpd ;

#define dsmFSUpdVersion  1

/*--------------------------------------------------------------------------+
|  Type definition for Filespace queryBuffer on dsmBeginQuery()            |
+--------------------------------------------------------------------------*/
typedef struct S_qryFSData
{
   dsUint16_t      stVersion;            /* structure version   */
   char        *fsName;                   /* File space name */
}qryFSData;

#define qryFSDataVersion  1

/*--------------------------------------------------------------------------+
| Type definition for Query Filespace response on dsmGetNextQObj()         |
+--------------------------------------------------------------------------*/
typedef struct S_qryRespFSData
{
   dsUint16_t      stVersion;            /* structure version   */
   char           fsName[DSM_MAX_FSNAME_LENGTH + 1];         /* Filespace name */
   char           fsType[DSM_MAX_FSTYPE_LENGTH + 1] ;        /* Filespace type */
   dsStruct64_t   occupancy;                        /* Occupancy est. in bytes.*/
   dsStruct64_t    capacity;                     /* Capacity est. in bytes. */
   dsmFSAttr       fsAttr ;                    /* platform specific attributes */
   dsmDate        backStartDate;           /* start backup date             */
   dsmDate        backCompleteDate;           /* end backup Date               */
   dsmDate        reserved1;                   /* For future use               */
   dsmDate        lastReplStartDate;        /* The last time replication was started */
   dsmDate        lastReplCmpltDate;        /* The last time replication completed   */
                                            /*   (could have had a failure,          */
                                            /*    but it still completes)            */
   dsmDate        lastBackOpDateFromServer; /* The last store time stamp the client  */
                                            /*    saved on the server               */
   dsmDate        lastArchOpDateFromServer; /* The last store time stamp the client  */
                                            /*    saved on the server               */
   dsmDate        lastSpMgOpDateFromServer; /* The last store time stamp the client  */
                                            /*    saved on the server               */
   dsmDate        lastBackOpDateFromLocal;  /* The last store time stamp the client  */
                                            /*    saved on the Local                */
   dsmDate        lastArchOpDateFromLocal;  /* The last store time stamp the client  */
                                            /*    saved on the Local                */
   dsmDate        lastSpMgOpDateFromLocal;  /* The last store time stamp the client  */
                                            /*    saved on the Local                */
   dsInt32_t      failOverWriteDelay;       /* Minutes for client to wait before  allowed  */
                                            /* to store to this Repl srvr, Specail codes:  */
                                            /* NO_ACCESS(-1), ACCESS_RDONLY (-2)           */
}qryRespFSData;

#define qryRespFSDataVersion 4
```

```
/*-------------------------------------------------------------------------+
|  Type definition for regFilespace parameter on dsmRegisterFS()
+-------------------------------------------------------------------------*/
typedef struct S_regFSData
{
   dsUint16_t      stVersion;           /* structure version   */
   char            *fsName;                    /* Filespace name */
   char            *fsType;                    /* Filespace type */
   dsStruct64_t    occupancy;                /* Occupancy est. in bytes. */
   dsStruct64_t    capacity;                  /* Capacity est. in bytes. */
   dsmFSAttr       fsAttr ;                  /* platform specific attributes */
}regFSData;

#define regFSDataVersion  1

/*-------------------------------------------------------------------------+
| Type definition for dedupType used in apisessInfo                        |
+-------------------------------------------------------------------------*/
typedef enum
{
   dedupServerOnly= 0x00,             /* dedup only done on server    */
   dedupClientOrServer               /* dedup can be done on client or server */
}dsmDedupType ;

/*-------------------------------------------------------------------------+
 | Type definition for fail over configuration and status
 -------------------------------------------------------------------------*/
typedef enum
{
   failOvrNotConfigured = 0x00,
   failOvrConfigured,
   failOvrConnectedToReplServer
}dsmFailOvrCfgType ;

/*-------------------------------------------------------------------------+
|  Type definition for session info response on dsmQuerySessionInfo()      |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t     stVersion;           /* Structure version               */
      /*-----------------------------------------------------------------*/
      /*           Server information                                    */
      /*-----------------------------------------------------------------*/
   char           serverHost[DSM_MAX_SERVERNAME_LENGTH+1];
                                  /* Network host name of DSM server  */
   dsUint16_t     serverPort;             /* Server comm port on host         */
   dsmDate        serverDate;             /* Server's date/time               */
   char           serverType[DSM_MAX_SERVERTYPE_LENGTH+1];
                                 /* Server's execution platform      */
   dsUint16_t     serverVer;              /* Server's version number          */
   dsUint16_t         serverRel;     /* Server's release number          */
   dsUint16_t     serverLev;              /* Server's level number            */
   dsUint16_t     serverSubLev;           /* Server's sublevel number         */
      /*-----------------------------------------------------------------*/
      /*           Client Defaults                                       */
      /*-----------------------------------------------------------------*/
   char           nodeType[DSM_MAX_PLATFORM_LENGTH+1]; /*node/application type*/
   char           fsdelim;                /* File space delimiter             */
   char           hldelim;                /* Delimiter betw highlev & lowlev  */
   dsUint8_t      compression;            /* Compression flag                 */
   dsUint8_t      archDel;                /* Archive delete permission        */
   dsUint8_t      backDel;                /* Backup  delete permission        */
   dsUint32_t     maxBytesPerTxn;         /* for future use                   */
   dsUint16_t     maxObjPerTxn;           /* The max objects allowed in a txn */
      /*-----------------------------------------------------------------*/
      /*           Session Information                                   */
      /*-----------------------------------------------------------------*/
   char       id[DSM_MAX_ID_LENGTH+1];    /* Sign-in id node name         */
   char       owner[DSM_MAX_OWNER_LENGTH+1]; /* Sign-in owner             */
```

```
                                    /*   (for multi-user platforms)     */
   char       confFile[DSM_PATH_MAX + DSM_NAME_MAX +1];
                                   /* len is platform dep             */
                                   /* dsInit name of appl config file  */
   dsUint8_t  opNoTrace;             /* dsInit option - NoTrace = 1      */
      /*---------------------------------------------------------------*/
      /*           Policy Data                                         */
      /*---------------------------------------------------------------*/
   char           domainName[DSM_MAX_DOMAIN_LENGTH+1]; /* Domain name     */
   char           policySetName[DSM_MAX_PS_NAME_LENGTH+1];
                                    /* Active policy set name           */
   dsmDate        polActDate;             /* Policy set activation date       */
   char           dfltMCName[DSM_MAX_MC_NAME_LENGTH+1];/* Default Mgmt Class  */
   dsUint16_t     gpBackRetn;             /* Grace-period backup retention    */
   dsUint16_t     gpArchRetn;             /* Grace-period archive retention   */
   char           adsmServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* adsm server name */
   dsmBool_t      archiveRetentionProtection; /* is server Retention protection enabled */
   dsStruct64_t   maxBytesPerTxn_64;         /* for future use                */
   dsmBool_t      lanFreeEnabled;            /* lan free option is set        */
   dsmDedupType   dedupType;                 /* server or clientOrServer      */
   char           accessNode[DSM_MAX_ID_LENGTH+1];    /* as node node name         */

   /*---------------------------------------------------------------*/
   /*           Replication and fail over information               */
   /*---------------------------------------------------------------*/
   dsmFailOvrCfgType failOverCfgType; /* status of fail over */
   char           replServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* repl server name */
   char               homeServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* home server name */
   char           replServerHost[DSM_MAX_SERVERNAME_LENGTH+1]; /* Network host name of DSM server  */
   dsInt32_t      replServerPort;                             /* Server comm port on host        */

}ApiSessInfo;

#define ApiSessInfoVersion  6



/*-------------------------------------------------------------------------+
| Type definition for Query options response on dsmQueryCliOptions()       |
|      and dsmQuerySessOptions()                                           |
+-------------------------------------------------------------------------*/

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 ;

/*-------------------------------------------------------------------------+
| Type definition for LogType used in logInfo                              |
+-------------------------------------------------------------------------*/
typedef enum
{
   logServer = 0x00,            /* log msg only to server    */
   logLocal,                    /* log msg only to local error log */
   logBoth,                      /* log msg to server and to local error log */
   logNone
}dsmLogType ;

/*-------------------------------------------------------------------------+
| Type definition for logInfo parameter used on dsmLogEvent()              |
+-------------------------------------------------------------------------*/
```

```
typedef struct
{
   char        *message;   /* text of message to be logged */
   dsmLogType  logType;    /* log type : local, server, both */
}logInfo;

/*-------------------------------------------------------------------------+
| Type definition for qryRespAccessData  parameter used on dsmQueryAccess()|
+-------------------------------------------------------------------------*/

typedef struct
{
   dsUint16_t       stVersion ;                    /* structure version      */
   char              node[DSM_MAX_ID_LENGTH+1];     /* node name              */
   char              owner[DSM_MAX_OWNER_LENGTH+1]; /* owner                  */
   dsmObjName        objName ;                      /* object name            */
   dsmAccessType     accessType;                    /* archive or backup     */
   dsUint32_t        ruleNumber ;                   /* Access rule id         */
}qryRespAccessData;

#define qryRespAccessDataVersion  1

/*-------------------------------------------------------------------------+
|  Type definition for envSetUp parameter on dsmSetUp()
+-------------------------------------------------------------------------*/
typedef struct S_envSetUp
{
   dsUint16_t      stVersion;            /* structure version   */
   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;         /* for future use */
   dsmBool_t       reserved2;              /* for future use */
}envSetUp;

#define envSetUpVersion  4

/*-------------------------------------------------------------------------+
|  Type definition for dsmInitExIn_t
+-------------------------------------------------------------------------*/
typedef struct dsmInitExIn_t
{
   dsUint16_t      stVersion;            /* structure version   */
   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

/*-------------------------------------------------------------------------+
|  Type definition for dsmInitExOut_t
+-------------------------------------------------------------------------*/
```

```
typedef struct dsmInitExOut_t
{
   dsUint16_t      stVersion;            /* structure version   */
   dsInt16_t          userNameAuthorities;
   dsInt16_t          infoRC;        /* error return code if encountered */
   char               adsmServerName[DSM_MAX_SERVERNAME_LENGTH+1];
   dsUint16_t     serverVer;             /* Server's version number      */
   dsUint16_t         serverRel;     /* Server's release number          */
   dsUint16_t     serverLev;             /* Server's level number            */
   dsUint16_t     serverSubLev;          /* Server's sublevel number         */

   dsmBool_t          bIsFailOverMode; /* true if failover has occured */
   char           replServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* repl server name */
   char               homeServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* home server name */

}dsmInitExOut_t;

#define dsmInitExOutVersion 3

/*------------------------------------------------------------------------+
| Type definition for LogType used in logInfo                             |
+------------------------------------------------------------------------*/
typedef enum
{
   logSevInfo = 0x00,       /* information ANE4991 */
   logSevWarning,           /* warning     ANE4992 */
   logSevError,             /* Error       ANE4993 */
   logSevSevere,            /* severe      ANE4994 */
   logSevLicense,           /* License     ANE4995 */
   logSevTryBuy             /* try Buy     ANE4996 */
}dsmLogSeverity ;


/*------------------------------------------------------------------------+
|  Type definition for dsmLogExIn_t
+------------------------------------------------------------------------*/
typedef struct dsmLogExIn_t
{
   dsUint16_t      stVersion;            /* structure version   */
   dsmLogSeverity     severity;
   char               appMsgID[8];
   dsmLogType  logType;    /* log type : local, server, both */
   char        *message;   /* text of message to be logged */
   char               appName[DSM_MAX_PLATFORM_LENGTH];
   char               osPlatform[DSM_MAX_PLATFORM_LENGTH];
   char               appVersion[DSM_MAX_PLATFORM_LENGTH];
}dsmLogExIn_t;

#define dsmLogExInVersion 2

/*------------------------------------------------------------------------+
|  Type definition for dsmlogExOut_t
+------------------------------------------------------------------------*/
typedef struct dsmLogExOut_t
{
   dsUint16_t      stVersion;            /* structure version   */
}dsmLogExOut_t;

#define dsmLogExOutVersion 1

/*------------------------------------------------------------------------+
|  Type definition for dsmRenameIn_t
+------------------------------------------------------------------------*/
typedef struct dsmRenameIn_t
{
   dsUint16_t      stVersion;            /* structure version   */
   dsUint32_t       dsmHandle;                    /* handle for session */
   dsUint8_t        repository;                   /* Backup or Archive  */
   dsmObjName       *objNameP ;                   /* object name */
   char             newHl[DSM_MAX_HL_LENGTH + 1];  /* new High level name */
```

```
   char             newLl[DSM_MAX_LL_LENGTH + 1];  /* new Low level name */
   dsmBool_t        merge;                        /* merge into existing name*/
   ObjID            objId;                         /* objId for Archive */
}dsmRenameIn_t;

#define dsmRenameInVersion 1

/*--------------------------------------------------------------------------+
|  Type definition for dsmRenameOut_t
+--------------------------------------------------------------------------*/
typedef struct dsmRenameOut_t
{
   dsUint16_t      stVersion;            /* structure version   */
}dsmRenameOut_t;

#define dsmRenameOutVersion 1

/*--------------------------------------------------------------------------+
|  Type definition for dsmEndSendObjExIn_t
+--------------------------------------------------------------------------*/
typedef struct dsmEndSendObjExIn_t
{
   dsUint16_t      stVersion;            /* structure version   */
   dsUint32_t       dsmHandle;                    /* handle for session */
}dsmEndSendObjExIn_t;

#define dsmEndSendObjExInVersion 1

/*--------------------------------------------------------------------------+
|  Type definition for dsmEndSendObjExOut_t
+--------------------------------------------------------------------------*/
typedef struct dsmEndSendObjExOut_t
{
   dsUint16_t      stVersion;            /* structure version   */
   dsStruct64_t       totalBytesSent;      /* total bytes read from app */
   dsmBool_t          objCompressed;       /* was object compressed */
   dsStruct64_t       totalCompressSize;   /* total size after compress */
   dsStruct64_t       totalLFBytesSent;    /* total bytes sent Lan Free */
   dsUint8_t          encryptionType;      /* type of encryption used   */
   dsmBool_t          objDeduplicated;     /* was object processed for dist. data dedup */
   dsStruct64_t       totalDedupSize;      /* total size after de-dup */
}dsmEndSendObjExOut_t;

#define dsmEndSendObjExOutVersion 3
/*--------------------------------------------------------------------------+
|  Type definition for dsmGroupHandlerIn_t
+--------------------------------------------------------------------------*/
typedef struct dsmGroupHandlerIn_t
{
   dsUint16_t      stVersion;            /* structure version   */
   dsUint32_t       dsmHandle;                    /* handle for session */
   dsUint8_t        groupType;        /* Type of group                          */
   dsUint8_t        actionType;       /* Type of group operation                */
   dsUint8_t        memberType;       /* Type of member: Leader or member       */
   dsStruct64_t    leaderObjId;      /* OBJID of the groupleader when manipulating a member */
   char            *uniqueGroupTagP; /* Unique group identifier             */
   dsmObjName      *objNameP ;       /* group leader object name */
   dsmGetList       memberObjList;    /* list of objects to remove, assign       */
}dsmGroupHandlerIn_t;

#define dsmGroupHandlerInVersion 1

/*--------------------------------------------------------------------------+
|  Type definition for dsmGroupHandlerExOut_t
+--------------------------------------------------------------------------*/
typedef struct dsmGroupHandlerOut_t
{
   dsUint16_t      stVersion;            /* structure version   */
}dsmGroupHandlerOut_t;
```

```
#define dsmGroupHandlerOutVersion 1

/*-------------------------------------------------------------------------+
|  Type definition for dsmEndTxnExIn_t
+-------------------------------------------------------------------------*/
typedef struct dsmEndTxnExIn_t
{
   dsUint16_t      stVersion;            /* structure version   */
   dsUint32_t       dsmHandle;                    /* handle for session */
   dsUint8_t        vote;
}dsmEndTxnExIn_t;

#define dsmEndTxnExInVersion 1

/*-------------------------------------------------------------------------+
|  Type definition for dsmEndTxnExOut_t
+-------------------------------------------------------------------------*/
typedef struct dsmEndTxnExOut_t
{
   dsUint16_t      stVersion;            /* structure version   */
   dsUint16_t     reason;            /* reason code                   */
   dsStruct64_t       groupLeaderObjId;        /* groupLeader obj id returned on */
                                               /* DSM_ACTION_OPEN               */
   dsUint8_t          reserved1;               /* future use                    */
   dsUint16_t         reserved2;               /* future use                    */
}dsmEndTxnExOut_t;

#define dsmEndTxnExOutVersion 1

/*-------------------------------------------------------------------------+
|  Type definition for dsmEndGetDataExIn_t
+-------------------------------------------------------------------------*/
typedef struct dsmEndGetDataExIn_t
{
   dsUint16_t      stVersion;            /* structure version   */
   dsUint32_t       dsmHandle;                    /* handle for session */
}dsmEndGetDataExIn_t;

#define dsmEndGetDataExInVersion 1

/*-------------------------------------------------------------------------+
|  Type definition for dsmEndGetDataExOut_t
+-------------------------------------------------------------------------*/
typedef struct dsmEndGetDataExOut_t
{
   dsUint16_t      stVersion;            /* structure version   */
   dsUint16_t     reason;            /* reason code                   */
   dsStruct64_t   totalLFBytesRecv;  /* total lan free bytes recieved */
}dsmEndGetDataExOut_t;

#define dsmEndGetDataExOutVersion 1

/*-------------------------------------------------------------------------+
| Type definition for object list on dsmRetentionEvent()                   |
+-------------------------------------------------------------------------*/
typedef struct dsmObjList
{
   dsUint16_t      stVersion;            /* structure version   */
   dsUint32_t       numObjId;          /* number of object IDs in the list */
   ObjID            *objId;            /* list of object IDs to signal     */
}dsmObjList_t ;

#define dsmObjlistVersion 1

/*-------------------------------------------------------------------------+
| Type definition eventType used on dsmRetentionEvent                       |
+-------------------------------------------------------------------------*/
typedef enum
{
   eventRetentionActivate = 0x00,   /* signal the server that the event has occured */
```

```
   eventHoldObj,                 /* suspend delete/expire of the object      */
   eventReleaseObj               /* Resume normal delete/expire processing   */
}dsmEventType_t;

/*-------------------------------------------------------------------------+
|  Type definition for  on dsmRetentionEvent()                             |
+-------------------------------------------------------------------------*/
typedef struct dsmRetentionEventIn_t
{
   dsUint16_t      stVersion;           /* structure version   */
   dsUint32_t         dsmHandle;                  /* session Handle     */
   dsmEventType_t     eventType;                  /* Event type         */
   dsmObjList_t       objList;                    /* object ID          */
}dsmRetentionEventIn_t;

#define dsmRetentionEventInVersion 1

/*-------------------------------------------------------------------------+
|  Type definition for  on dsmRetentionEvent()                             |
+-------------------------------------------------------------------------*/
typedef struct dsmRetentionEventOut_t
{
   dsUint16_t       stVersion ;                 /* structure version      */
}dsmRetentionEventOut_t;

#define dsmRetentionEventOutVersion 1


/*-------------------------------------------------------------------------+
|  Type definition for  on dsmRequestBuffer()                                |
+-------------------------------------------------------------------------*/
typedef struct requestBufferIn_t
{
   dsUint16_t      stVersion;           /* structure version   */
   dsUint32_t         dsmHandle;                  /* session Handle     */
}requestBufferIn_t;

#define requestBufferInVersion 1

/*-------------------------------------------------------------------------+
|  Type definition for  on dsmRequestBuffer()                                |
+-------------------------------------------------------------------------*/
typedef struct requestBufferOut_t
{
   dsUint16_t       stVersion ;                 /* structure version      */
   dsUint8_t        tsmBufferHandle;          /* handle to tsm Data buffer */
   char             *dataPtr;                 /* Address to write data to */
   dsUint32_t       bufferLen;                /* Max length of data to be written */
}requestBufferOut_t;

#define requestBufferOutVersion 1

/*-------------------------------------------------------------------------+
|  Type definition for  on dsmReleaseBuffer()                              |
+-------------------------------------------------------------------------*/
typedef struct releaseBufferIn_t
{
   dsUint16_t      stVersion;           /* structure version   */
   dsUint32_t         dsmHandle;                  /* session Handle     */
   dsUint8_t        tsmBufferHandle;          /* handle to tsm Data buffer */
   char             *dataPtr;                 /* Address to write data to */
}releaseBufferIn_t;

#define releaseBufferInVersion 1

/*-------------------------------------------------------------------------+
|  Type definition for  on dsmReleaseBuffer()                              |
+-------------------------------------------------------------------------*/
typedef struct releaseBufferOut_t
{
```

```
   dsUint16_t       stVersion ;                   /* structure version      */
}releaseBufferOut_t;

#define releaseBufferOutVersion 1

/*-------------------------------------------------------------------------+
|  Type definition for  on dsmGetBufferData()                              |
+-------------------------------------------------------------------------*/
typedef struct getBufferDataIn_t
{
   dsUint16_t      stVersion;            /* structure version   */
   dsUint32_t         dsmHandle;                    /* session Handle     */
}getBufferDataIn_t;

#define getBufferDataInVersion 1

/*-------------------------------------------------------------------------+
|  Type definition for  on dsmGetBufferData()                              |
+-------------------------------------------------------------------------*/
typedef struct getBufferDataOut_t
{
   dsUint16_t       stVersion ;                   /* structure version      */
   dsUint8_t        tsmBufferHandle;           /* handle to tsm Data buffer */
   char             *dataPtr;              /* Address of actual data to read */
   dsUint32_t       numBytes;              /* Actual number of bytes to read from dataPtr*/
}getBufferDataOut_t;

#define getBufferDataOutVersion 1

/*-------------------------------------------------------------------------+
|  Type definition for  on dsmSendBufferData()                             |
+-------------------------------------------------------------------------*/
typedef struct sendBufferDataIn_t
{
   dsUint16_t      stVersion;            /* structure version   */
   dsUint32_t         dsmHandle;                    /* session Handle     */
   dsUint8_t        tsmBufferHandle;           /* handle to tsm Data buffer */
   char               *dataPtr;              /* Address of actual data to send */
   dsUint32_t         numBytes;              /* Actual number of bytes to send from dataPtr*/
}sendBufferDataIn_t;

#define sendBufferDataInVersion 1

/*-------------------------------------------------------------------------+
|  Type definition for  on dsmSendBufferData()                             |
+-------------------------------------------------------------------------*/
typedef struct sendBufferDataOut_t
{
   dsUint16_t       stVersion ;                   /* structure version      */
}sendBufferDataOut_t;

#define sendBufferDataOutVersion 1

/*-------------------------------------------------------------------------+
|  Type definition for dsmUpdateObjExIn_t
+-------------------------------------------------------------------------*/
typedef struct dsmUpdateObjExIn_t
{
   dsUint16_t      stVersion;            /* structure version   */
   dsUint32_t         dsmHandle;                    /* session Handle     */
   dsmSendType     sendType;             /* send type back/arch */
   char            *descrP;              /* archive description */
   dsmObjName      *objNameP;            /* objName             */
   ObjAttr         *objAttrPtr;          /* attribute           */
   dsUint32_t      objUpdAct;            /* update action       */
   ObjID           archObjId;            /* objId for archive   */
}dsmUpdateObjExIn_t;

#define dsmUpdateObjExInVersion 1
```

```
/*-------------------------------------------------------------------------+
|  Type definition for dsmUpdateObjExOut_t
+-------------------------------------------------------------------------*/
typedef struct dsmUpdateObjExOut_t
{
    dsUint16_t      stVersion;            /* structure version   */
}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 */
/*************************************************************************
 * Tivoli Storage Manager                                                *
 * API Client Component                                                  *
 *                                                                       *
 * (C) Copyright IBM Corporation 1993,2010                               *
 *************************************************************************/

/*************************************************************************
 * Header File Name: tsmapitd.h
 *
 * Environment:     ************************************************
 *                  ** This is a platform-independent source file **
 *
 *                  ************************************************
 *
 * Design Notes:    This file contains basic data types and constants
 *                  includable by all client source files. The constants
 *                  within this file should be set properly for the
 *                  particular machine and operating system on which the
 *                  client software is to be run.
 *
 *                  Platform specific definitions are included in dsmapips.h
 *
 * Descriptive-name: Definitions for Tivoli Storage manager API constants
 *-------------------------------------------------------------------------*/

#ifndef _H_TSMAPITD
#define _H_TSMAPITD


/*=== set the structure alignment to pack the structures ===*/
#if _OPSYS_TYPE == DS_WINNT
#ifdef _WIN64
#pragma pack(8)
#else
#pragma pack(1)
#endif
#endif

#ifdef _MAC
#pragma options align = packed
#endif

/*============================================================
 Win32 applications using the tsm interface must use the
 -DUNICODE flag during compilation.
 ============================================================*/
#if _OPSYS_TYPE == DS_WINNT && !defined(DSMAPILIB)
#ifndef UNICODE
```

```
#error "Win32 applications using the TSM interface MUST be compiled with the -DUNICODE flag"
#endif
#endif

/*============================================================
 Mac OS X applications using the tsm interface must use the
 -DUNICODE flag during compilation.
 ============================================================*/
#if _OPSYS_TYPE == DS_MACOS && !defined(DSMAPILIB)
#ifndef UNICODE
#error "Mac OS X applications using the TSM interface MUST be compiled with the -DUNICODE flag"
#endif
#endif

/*-------------------------------------------------------------------------+
 | Type definition for dsmGetType parameter on tsmBeginGetData()           |
 +-------------------------------------------------------------------------*/
typedef enum
{
   gtTsmBackup = 0x00,                    /* Backup processing type    */
   gtTsmArchive                           /* Archive processing type   */
} tsmGetType ;

/*-------------------------------------------------------------------------+
 |  Type definition for dsmQueryType parameter on tsmBeginQuery()          |
 +-------------------------------------------------------------------------*/
typedef enum
{
   qtTsmArchive = 0x00,                    /* Archive query type        */
   qtTsmBackup,                            /* Backup query type         */
   qtTsmBackupActive,                      /* Fast query for active backup files */
   qtTsmFilespace,                         /* Filespace query type      */
   qtTsmMC,                                /* Mgmt. class query type    */
    qtTsmReserved1,                         /* future use                */
    qtTsmReserved2,                         /* future use                */
    qtTsmReserved3,                         /* future use                */
    qtTsmReserved4,                         /* future use                */
   qtTsmBackupGroups,                      /* All group leaders in a specific filespace */
   qtTsmOpenGroups,                        /* All group members associated with a leader */
    qtTsmReserved5,                         /* future use                */
   qtTsmProxyNodeAuth,                     /* nodes that this node can proxy to   */
    qtTsmProxyNodePeer,                     /* peer nodes under this target node  */
    qtTsmReserved6,                         /* future use                */
    qtTsmReserved7,                         /* future use                */
    qtTsmReserved8                          /* future use                */
} tsmQueryType ;

/*-------------------------------------------------------------------------+
 |  Type definition sendType parameter on tsmBindMC() and tsmSendObj()     |
 +-------------------------------------------------------------------------*/
typedef enum
{
   stTsmBackup = 0x00,                        /* Backup processing type    */
   stTsmArchive,                              /* Archive processing type   */
   stTsmBackupMountWait,           /* Backup processing with mountwait on   */
   stTsmArchiveMountWait           /* Archive processing with mountwait on  */
} tsmSendType ;

/*-------------------------------------------------------------------------+
 |  Type definition for delType parameter on tsmDeleteObj()                |
 +-------------------------------------------------------------------------*/
typedef enum
{
   dtTsmArchive = 0x00,                             /* Archive delete type */
   dtTsmBackup,                           /* Backup delete (deactivate) type */
   dtTsmBackupID                         /* Backup delete (remove)     type */
} tsmDelType ;
```

```
/*-------------------------------------------------------------------------+
 |  Type definition sendType parameter on tsmSetAccess()                   |
 +-------------------------------------------------------------------------*/
typedef enum
{
   atTsmBackup = 0x00,                          /* Backup processing type    */
   atTsmArchive                                /* Archive processing type   */
}tsmAccessType;

/*-------------------------------------------------------------------------+
 |  Type definition for Overwrite  parameter on tsmSendObj()               |
 +-------------------------------------------------------------------------*/
typedef enum
{
   owIGNORE = 0x00,
   owYES,
   owNO
}tsmOwType;


/*-------------------------------------------------------------------------+
 |  Type definition for API Version on tsmInit() and tsmQueryApiVersion()  |
 +-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t     stVersion;           /* Structure version               */
   dsUint16_t version;       /* API version                       */
   dsUint16_t release;       /* API release                       */
   dsUint16_t level;         /* API level                         */
   dsUint16_t subLevel;      /* API sub level                     */
   dsmBool_t  unicode;       /* API unicode?                      */
} tsmApiVersionEx;

#define tsmApiVersionExVer       2

/*-------------------------------------------------------------------------+
|  Type definition for Application Version on tsmInit()                    |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t     stVersion;           /* Structure version               */
   dsUint16_t   applicationVersion;   /* application version number   */
   dsUint16_t   applicationRelease;   /* application release number   */
   dsUint16_t   applicationLevel;     /* application level number     */
   dsUint16_t   applicationSubLevel;  /* application sub level number */
} tsmAppVersion;

#define tsmAppVersionVer      1


/*-------------------------------------------------------------------------+
 |  Type definition for object name used on BindMC, Send, Delete, Query    |
 +-------------------------------------------------------------------------*/

typedef struct tsmObjName
{
   dsChar_t   fs[DSM_MAX_FSNAME_LENGTH + 1] ;              /* Filespace name */
   dsChar_t   hl[DSM_MAX_HL_LENGTH + 1] ;                 /* High level name */
   dsChar_t   ll[DSM_MAX_LL_LENGTH + 1] ;                  /* Low level name */
   dsUint8_t  objType;          /* for object type values, see defines above */
   dsChar_t           dirDelimiter;
} tsmObjName;

/*-------------------------------------------------------------------------+
 |  Type definition for Backup delete info on dsmDeleteObj()               |
 +-------------------------------------------------------------------------*/
```

```
typedef struct tsmDelBack
{
   dsUint16_t       stVersion ;                   /* structure version      */
   tsmObjName       *objNameP ;                   /* object name */
   dsUint32_t       copyGroup ;                   /* copy group             */
} tsmDelBack ;

#define  tsmDelBackVersion    1

/*-------------------------------------------------------------------------+
 |  Type definition for Archive delete info on dsmDeleteObj()              |
 +-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t       stVersion ;                   /* structure version      */
   dsStruct64_t       objId ;                       /* object ID              */
} tsmDelArch ;

#define  tsmDelArchVersion  1

/*-------------------------------------------------------------------------+
 |  Type definition for Backup ID delete info on dsmDeleteObj()            |
 +-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t       stVersion ;                   /* structure version      */
   dsStruct64_t       objId ;                       /* object ID              */
} tsmDelBackID;

#define  tsmDelBackIDVersion  1


/*-------------------------------------------------------------------------+
 |  Type definition for delete info on dsmDeleteObj()                      |
 +-------------------------------------------------------------------------*/
typedef union
{
   tsmDelBack   backInfo ;
   tsmDelArch   archInfo ;
   tsmDelBackID backIDInfo;
} tsmDelInfo ;

/*-------------------------------------------------------------------------+
 |  Type definition for Object Attribute parameter on dsmSendObj()         |
 +-------------------------------------------------------------------------*/
typedef struct tsmObjAttr
{
   dsUint16_t     stVersion;           /* Structure version               */
   dsChar_t     owner[DSM_MAX_OWNER_LENGTH + 1];      /* object owner         */
   dsStruct64_t sizeEstimate;             /* Size estimate in bytes of the object */
   dsmBool_t    objCompressed;            /* Is object already compressed?     */
   dsUint16_t   objInfoLength;            /* length of object-dependent info   */
   char         *objInfo;                 /* object-dependent info byte buffer */
   dsChar_t     *mcNameP;                 /* mgmnt class name for override     */
   tsmOwType    reserved1;                /* for future use                    */
   tsmOwType    reserved2;                /* for future use                    */
   dsmBool_t    disableDeduplication;         /* force no dedup for this object    */
   dsmBool_t    useExtObjInfo;             /* use ext objinfo up to 1536        */
} tsmObjAttr;

#define tsmObjAttrVersion 5

/*-------------------------------------------------------------------------+
 | Type definition for mcBindKey returned on dsmBindMC()                   |
 +-------------------------------------------------------------------------*/
typedef struct tsmMcBindKey
{
```

```
   dsUint16_t      stVersion;            /* structure version   */
   dsChar_t    mcName[DSM_MAX_MC_NAME_LENGTH + 1];
   /* Name of mc bound to object. */
   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];
   /* Backup copy dest. name */
   dsChar_t    archive_copy_dest[DSM_MAX_CG_DEST_LENGTH + 1];
   /* Arch copy dest.name */
} tsmMcBindKey;

#define tsmMcBindKeyVersion  1


/*-------------------------------------------------------------------------+
 |  Type definition for Mgmt Class queryBuffer on dsmBeginQuery()          |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryMCData
{
   dsUint16_t      stVersion;            /* structure version   */
   dsChar_t     *mcName;                     /* Mgmt class name */
   /* single name to get one or empty string to get all*/
   dsmBool_t    mcDetail;                          /* Want details or not? */
} tsmQryMCData;

#define tsmQryMCDataVersion  1


/*-------------------------------------------------------------------------+
 |  Type definition for Archive Copy Group details on Query MC response    |
 +-------------------------------------------------------------------------*/
typedef struct tsmArchDetailCG
{
   dsChar_t     cgName[DSM_MAX_CG_NAME_LENGTH + 1];       /* Copy group name */
   dsUint16_t   frequency;                    /* Copy (archive) frequency */
   dsUint16_t   retainVers;                             /* Retain version */
   dsUint8_t    copySer;       /* for copy serialization values, see defines */
   dsUint8_t    copyMode;       /* for copy mode values, see defines above */
   dsChar_t     destName[DSM_MAX_CG_DEST_LENGTH + 1];      /* Copy dest name */
   dsmBool_t    bLanFreeDest;         /* Destination has lan free path?    */
   dsmBool_t    reserved;                          /* Not currently used */
   dsUint8_t    retainInit;             /* possible values see dsmapitd.h  */
   dsUint16_t   retainMin;              /* if retInit is EVENT num of days */
   dsmBool_t    bDeduplicate;            /* destination has dedup enabled    */
}tsmArchDetailCG;

/*-------------------------------------------------------------------------+
 |  Type definition for Backup Copy Group details on Query MC response     |
 +-------------------------------------------------------------------------*/
typedef struct tsmBackupDetailCG
{
   dsChar_t     cgName[DSM_MAX_CG_NAME_LENGTH + 1];       /* Copy group name */
   dsUint16_t   frequency;                              /* Backup frequency */
   dsUint16_t   verDataExst;                         /* Versions data exists */
   dsUint16_t   verDataDltd;                        /* Versions data deleted */
   dsUint16_t   retXtraVers;                        /* Retain extra versions */
   dsUint16_t   retOnlyVers;                         /* Retain only versions */
   dsUint8_t    copySer;       /* for copy serialization values, see defines */
   dsUint8_t    copyMode;       /* for copy mode values, see defines above */
   dsChar_t     destName[DSM_MAX_CG_DEST_LENGTH + 1];      /* Copy dest name */
   dsmBool_t    bLanFreeDest;           /* Destination has lan free path?    */
   dsmBool_t    reserved;                          /* Not currently used */
   dsmBool_t    bDeduplicate;            /* destination has dedup enabled    */
}tsmBackupDetailCG;


/*-------------------------------------------------------------------------+
```

```
 |  Type definition for Query Mgmt Class detail response on dsmGetNextQObj()|
 +-------------------------------------------------------------------------*/
typedef struct tsmQryRespMCDetailData
{
   dsUint16_t      stVersion;            /* structure version   */
   dsChar_t        mcName[DSM_MAX_MC_NAME_LENGTH + 1];        /* mc name */
   dsChar_t        mcDesc[DSM_MAX_MC_DESCR_LENGTH + 1]; /*mc description */
   archDetailCG    archDet;                /* Archive copy group detail */
   backupDetailCG  backupDet;               /* Backup copy group detail */
} tsmQryRespMCDetailData;

#define tsmQryRespMCDetailDataVersion  4

/*-------------------------------------------------------------------------+
 | Type definition for Query Mgmt Class summary response on dsmGetNextQObj()|
 +-------------------------------------------------------------------------*/
typedef struct tsmQryRespMCData
{
   dsUint16_t      stVersion;            /* structure version   */
   dsChar_t        mcName[DSM_MAX_MC_NAME_LENGTH + 1];           /* mc name */
   dsChar_t     mcDesc[DSM_MAX_MC_DESCR_LENGTH + 1];       /* mc description */
}tsmQryRespMCData;

#define tsmQryRespMCDataVersion  1

/*-------------------------------------------------------------------------+
 |  Type definition for Archive queryBuffer on tsmBeginQuery()              |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryArchiveData
{
   dsUint16_t      stVersion;            /* structure version   */
   tsmObjName   *objName;                    /* Full dsm name of object */
   dsChar_t     *owner;                                /* owner name */
   /* for maximum date boundaries, see defines above */
   dsmDate      insDateLowerBound;      /* low bound archive insert date */
   dsmDate      insDateUpperBound;       /* hi bound archive insert date */
   dsmDate      expDateLowerBound;          /* low bound expiration date */
   dsmDate      expDateUpperBound;           /* hi bound expiration date */
   dsChar_t     *descr;               /* archive description */
} tsmQryArchiveData;

#define tsmQryArchiveDataVersion  1

/*-------------------------------------------------------------------------+
| Type definition for Query Archive response on dsmGetNextQObj()           |
+-------------------------------------------------------------------------*/
typedef struct tsmQryRespArchiveData
{
    dsUint16_t      stVersion;            /* structure version   */
    tsmObjName      objName;                      /* Filespace name qualifier */
    dsUint32_t  copyGroup;                         /* copy group number */
    dsChar_t        mcName[DSM_MAX_MC_NAME_LENGTH + 1];           /* mc name */
    dsChar_t    owner[DSM_MAX_OWNER_LENGTH + 1];           /* owner name */
    dsStruct64_t    objId;                                 /* Unique copy id */
    dsStruct64_t    reserved;                        /* backward compatability */
    dsUint8_t       mediaClass;                         /* media access class */
    dsmDate         insDate;                       /* archive insertion date */
    dsmDate         expDate;                     /* expiration date for object */
    dsChar_t        descr[DSM_MAX_DESCR_LENGTH + 1];   /* archive description */
    dsUint16_t      objInfolen;             /* length of object-dependent info*/
    dsUint8_t       reservedObjInfo[DSM_MAX_OBJINFO_LENGTH];  /*object-dependent info */
    dsUint160_t     restoreOrderExt;                        /* restore order */
    dsStruct64_t    sizeEstimate;              /* size estimate stored by user*/
    dsUint8_t       compressType;                         /* Compression flag */
    dsUint8_t       retentionInitiated;   /* object waiting on retention event*/
    dsUint8_t       objHeld; /* object is on "hold" see dsmapitd.h for values */
    dsUint8_t       encryptionType;                    /* type of encryption */
```

```
    dsmBool_t       clientDeduplicated;             /* obj deduplicated by API*/
    dsUint8_t       objInfo[DSM_MAX_EXT_OBJINFO_LENGTH];  /*object-dependent info */
    dsChar_t        compressAlg[DSM_MAX_COMPRESSTYPE_LENGTH + 1];  /* compression algorithm name */
} tsmQryRespArchiveData;

#define tsmQryRespArchiveDataVersion  7

/*-------------------------------------------------------------------------+
 |  Type definition for Archive sendBuff parameter on dsmSendObj()         |
 +-------------------------------------------------------------------------*/
typedef struct tsmSndArchiveData
{
   dsUint16_t      stVersion;            /* structure version   */
   dsChar_t     *descr;                /* archive description */
} tsmSndArchiveData;

#define tsmSndArchiveDataVersion  1

/*-------------------------------------------------------------------------+
 |  Type definition for Backup queryBuffer on dsmBeginQuery()              |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryBackupData
{
   dsUint16_t      stVersion;            /* structure version   */
   tsmObjName  *objName;          /* full dsm name of object */
   dsChar_t     *owner;                                    /* owner name */
   dsUint8_t   objState;          /* object state selector */
   dsmDate     pitDate;           /* Date value for point in time restore */
   /* for possible values, see defines above */
    dsUint32_t  reserved1;
    dsUint32_t  reserved2;
} tsmQryBackupData;

#define tsmQryBackupDataVersion  3

/*-------------------------------------------------------------------------+
 | Type definition for Query Backup response on dsmGetNextQObj()           |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryRespBackupData
{
   dsUint16_t      stVersion;            /* structure version   */
   tsmObjName  objName;                       /* full dsm name of object */
   dsUint32_t  copyGroup;                          /* copy group number */
   dsChar_t        mcName[DSM_MAX_MC_NAME_LENGTH + 1];           /* mc name */
   dsChar_t    owner[DSM_MAX_OWNER_LENGTH + 1];            /* owner name */
   dsStruct64_t    objId;                                /* Unique object id */
   dsStruct64_t    reserved;                       /* backward compatability */
   dsUint8_t       mediaClass;                         /* media access class */
   dsUint8_t       objState;                      /* Obj state, active, etc. */
   dsmDate     insDate;                        /* backup insertion date */
   dsmDate         expDate;                    /* expiration date for object */
   dsUint16_t      objInfolen;             /* length of object-dependent info*/
   dsUint8_t       reservedObjInfo[DSM_MAX_OBJINFO_LENGTH];  /*object-dependent info */
   dsUint160_t     restoreOrderExt;                          /* restore order */
   dsStruct64_t    sizeEstimate;             /* size estimate stored by user */
   dsStruct64_t    baseObjId;
   dsUint16_t      baseObjInfolen;             /* length of base object-dependent info*/
   dsUint8_t       baseObjInfo[DSM_MAX_OBJINFO_LENGTH];  /* base object-dependent info */
   dsUint160_t     baseRestoreOrder;                          /* restore order */
   dsUint32_t     fsID;
   dsUint8_t       compressType;
   dsmBool_t       isGroupLeader;
   dsmBool_t       isOpenGroup;
   dsUint8_t       reserved1;               /* for future use */
   dsmBool_t       reserved2;               /* for future use */
   dsUint16_t      reserved3;               /* for future use */
   reservedInfo_t  *reserved4;              /* for future use */
```

```
   dsUint8_t       encryptionType;                    /* type of encryption */
   dsmBool_t       clientDeduplicated;             /* obj deduplicated by API*/
   dsUint8_t       objInfo[DSM_MAX_EXT_OBJINFO_LENGTH];  /*object-dependent info */
   dsChar_t        compressAlg[DSM_MAX_COMPRESSTYPE_LENGTH + 1];  /* compression algorithm name */
} tsmQryRespBackupData;

#define tsmQryRespBackupDataVersion  8
/*-------------------------------------------------------------------------+
 |  Type definition for Active Backup queryBuffer on dsmBeginQuery()
 |
 |  Notes:  For the active backup query, only the fs (filespace) and objType
 |          fields of objName need be set.  objType can only be set to
 |          DSM_OBJ_FILE or DSM_OBJ_DIRECTORY.  DSM_OBJ_ANY_TYPE will not
 |          find a match on the query.
 +-------------------------------------------------------------------------*/
typedef struct tsmQryABackupData
{
   dsUint16_t      stVersion;            /* structure version   */
   tsmObjName      *objName;                     /* Only fs and objtype used */
} tsmQryABackupData;

#define tsmQryABackupDataVersion  1

/*-------------------------------------------------------------------------+
 | Type definition for Query Active Backup response on dsmGetNextQObj()    |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryARespBackupData
{
   dsUint16_t      stVersion;            /* structure version   */
   tsmObjName  objName;                       /* full dsm name of object */
   dsUint32_t  copyGroup;                          /* copy group number */
   dsChar_t    mcName[DSM_MAX_MC_NAME_LENGTH + 1];/*management class name*/
   dsChar_t    owner[DSM_MAX_OWNER_LENGTH + 1];            /* owner name */
   dsmDate     insDate;                         /* backup insertion date */
   dsUint16_t      objInfolen;             /* length of object-dependent info*/
   dsUint8_t   reservedObjInfo[DSM_MAX_OBJINFO_LENGTH];  /*object-dependent info */
   dsUint8_t  objInfo[DSM_MAX_EXT_OBJINFO_LENGTH];  /*object-dependent info */
} tsmQryARespBackupData;

#define tsmQryARespBackupDataVersion  2

/*-------------------------------------------------------------------------+
 |  Type definition for Backup queryBuffer on dsmBeginQuery()               |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryBackupGroups
{
   dsUint16_t      stVersion;            /* structure version   */
   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

/*-------------------------------------------------------------------------+
 |  Type definition for proxynode queryBuffer on tsmBeginQuery()            |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryProxyNodeData
{
   dsUint16_t      stVersion;            /* structure version   */
```

```
   dsChar_t    *targetNodeName;            /* target node name      */
}tsmQryProxyNodeData;

#define tsmQryProxyNodeDataVersion  1

/*-------------------------------------------------------------------------+
 | Type definition for qryRespProxyNodeData  parameter used on tsmGetNextQObj()|
 +-------------------------------------------------------------------------*/

typedef struct tsmQryRespProxyNodeData
{
   dsUint16_t       stVersion ;                   /* structure version      */
   dsChar_t          targetNodeName[DSM_MAX_ID_LENGTH+1];  /* target node name  */
   dsChar_t          peerNodeName[DSM_MAX_ID_LENGTH+1];    /* peer node name    */
   dsChar_t          hlAddress[DSM_MAX_ID_LENGTH+1];       /* peer hlAddress    */
   dsChar_t          llAddress[DSM_MAX_ID_LENGTH+1];       /* peer llAddress    */
}tsmQryRespProxyNodeData;

#define tsmQryRespProxyNodeDataVersion  1

/*-------------------------------------------------------------------------+
 |  Type definition for WINNT and OS/2 Filespace attributes                |
 +-------------------------------------------------------------------------*/
typedef struct tsmDosFSAttrib
{
   osChar_t     driveLetter ;          /* drive letter for filespace    */
   dsUint16_t   fsInfoLength;          /* fsInfo length used            */
   osChar_t     fsInfo[DSM_MAX_FSINFO_LENGTH];/*caller-determined data  */
} tsmDosFSAttrib ;

/*-------------------------------------------------------------------------+
 |  Type definition for UNIX Filespace attributes                          |
 +-------------------------------------------------------------------------*/
typedef struct tsmUnixFSAttrib
{
   dsUint16_t   fsInfoLength;          /* fsInfo length used            */
   osChar_t     fsInfo[DSM_MAX_FSINFO_LENGTH];/*caller-determined data  */
} tsmUnixFSAttrib ;

/*-------------------------------------------------------------------------+
 |  Type definition for NetWare Filespace attributes                       |
 +-------------------------------------------------------------------------*/
typedef tsmUnixFSAttrib tsmNetwareFSAttrib;

/*-------------------------------------------------------------------------+
 |  Type definition for Filespace attributes on all Filespace calls        |
 +-------------------------------------------------------------------------*/
typedef union
{
   tsmNetwareFSAttrib  netwareFSAttr;
   tsmUnixFSAttrib     unixFSAttr ;
   tsmDosFSAttrib      dosFSAttr ;
} tsmFSAttr ;

/*-------------------------------------------------------------------------+
 |  Type definition for fsUpd parameter on dsmUpdateFS()
 +-------------------------------------------------------------------------*/
typedef struct   tsmFSUpd
{
   dsUint16_t       stVersion ;                  /* structure version      */
   dsChar_t        *fsType ;              /* filespace type                */
   dsStruct64_t    occupancy ;            /* occupancy estimate            */
   dsStruct64_t    capacity  ;            /* capacity estimate             */
   tsmFSAttr       fsAttr ;                /* platform specific attributes */
} tsmFSUpd ;

#define tsmFSUpdVersion  1
```

```
/*-------------------------------------------------------------------------+
 |  Type definition for Filespace queryBuffer on dsmBeginQuery()           |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryFSData
{
   dsUint16_t      stVersion;            /* structure version   */
   dsChar_t    *fsName;                   /* File space name */
} tsmQryFSData;

#define tsmQryFSDataVersion  1

/*-------------------------------------------------------------------------+
 | Type definition for Query Filespace response on dsmGetNextQObj()        |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryRespFSData
{
   dsUint16_t      stVersion;            /* structure version   */
   dsChar_t       fsName[DSM_MAX_FSNAME_LENGTH + 1];     /* Filespace name */
   dsChar_t       fsType[DSM_MAX_FSTYPE_LENGTH + 1] ;     /* Filespace type */
   dsStruct64_t    occupancy;                  /* Occupancy est. in bytes.             */
   dsStruct64_t    capacity;                       /* Capacity est. in bytes.              */
   tsmFSAttr       fsAttr ;                /* platform specific attributes */
   dsmDate        backStartDate;           /* start backup date            */
   dsmDate        backCompleteDate;          /* end backup Date              */
    dsmDate        reserved1    ;          /* For future use               */
   dsmBool_t      bIsUnicode;
   dsUint32_t     fsID;
   dsmDate        lastReplStartDate;        /* The last time replication was started */
   dsmDate        lastReplCmpltDate;        /* The last time replication completed   */
                                            /*   (could have had a failure,          */
                                            /*    but it still completes)            */
   dsmDate        lastBackOpDateFromServer; /* The last store time stamp the client  */
                                            /*    saved on the server               */
   dsmDate        lastArchOpDateFromServer; /* The last store time stamp the client  */
                                            /*    saved on the server               */
   dsmDate        lastSpMgOpDateFromServer; /* The last store time stamp the client  */
                                            /*    saved on the server               */
   dsmDate        lastBackOpDateFromLocal;  /* The last store time stamp the client  */
                                            /*    saved on the Local                */
   dsmDate        lastArchOpDateFromLocal;  /* The last store time stamp the client  */
                                            /*    saved on the Local                */
   dsmDate        lastSpMgOpDateFromLocal;  /* The last store time stamp the client  */
                                            /*    saved on the Local                */
   dsInt32_t      failOverWriteDelay;       /* Minutes for client to wait before  allowed  */
                                            /* to store to this Repl srvr, Specail codes:  */
                                            /* NO_ACCESS(-1), ACCESS_RDONLY (-2)           */
} tsmQryRespFSData;

#define tsmQryRespFSDataVersion 5

/*-------------------------------------------------------------------------+
 |  Type definition for regFilespace parameter on dsmRegisterFS()
 +-------------------------------------------------------------------------*/
typedef struct tsmRegFSData
{
   dsUint16_t      stVersion;            /* structure version   */
   dsChar_t        *fsName;                        /* Filespace name */
   dsChar_t        *fsType;                        /* Filespace type */
   dsStruct64_t    occupancy;                  /* Occupancy est. in bytes. */
   dsStruct64_t    capacity;                    /* Capacity est. in bytes. */
   tsmFSAttr       fsAttr ;                /* platform specific attributes */
} tsmRegFSData;

#define tsmRegFSDataVersion  1

/*-------------------------------------------------------------------------+
```

```
 |  Type definition for session info response on dsmQuerySessionInfo()      |
 +-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t     stVersion;           /* Structure version                */
   /*-----------------------------------------------------------------*/
   /*           Server information                                    */
   /*-----------------------------------------------------------------*/
   dsChar_t       serverHost[DSM_MAX_SERVERNAME_LENGTH+1];
   /* Network host name of DSM server  */
   dsUint16_t     serverPort;             /* Server comm port on host         */
   dsmDate        serverDate;             /* Server's date/time               */
   dsChar_t       serverType[DSM_MAX_SERVERTYPE_LENGTH+1];
   /* Server's execution platform      */
   dsUint16_t     serverVer;              /* Server's version number          */
   dsUint16_t         serverRel;     /* Server's release number          */
   dsUint16_t     serverLev;              /* Server's level number            */
   dsUint16_t     serverSubLev;           /* Server's sublevel number         */
   /*-----------------------------------------------------------------*/
   /*           Client Defaults                                       */
   /*-----------------------------------------------------------------*/
   dsChar_t       nodeType[DSM_MAX_PLATFORM_LENGTH+1]; /*node/application type*/
   dsChar_t       fsdelim;                /* File space delimiter             */
   dsChar_t       hldelim;                /* Delimiter betw highlev & lowlev  */
   dsUint8_t      compression;            /* Compression flag                 */
   dsUint8_t      archDel;                /* Archive delete permission        */
   dsUint8_t      backDel;                /* Backup  delete permission        */
   dsUint32_t     maxBytesPerTxn;         /* for future use                   */
   dsUint16_t     maxObjPerTxn;           /* The max objects allowed in a txn */
   /*-----------------------------------------------------------------*/
   /*           Session Information                                   */
   /*-----------------------------------------------------------------*/
   dsChar_t       id[DSM_MAX_ID_LENGTH+1];    /* Sign-in id node name         */
   dsChar_t       owner[DSM_MAX_OWNER_LENGTH+1]; /* Sign-in owner             */
   /*   (for multi-user platforms)     */
   dsChar_t       confFile[DSM_PATH_MAX + DSM_NAME_MAX +1];
   /* len is platform dep              */
   /* dsInit name of appl config file  */
   dsUint8_t  opNoTrace;              /* dsInit option - NoTrace = 1      */
   /*-----------------------------------------------------------------*/
   /*           Policy Data                                           */
   /*-----------------------------------------------------------------*/
   dsChar_t       domainName[DSM_MAX_DOMAIN_LENGTH+1]; /* Domain name         */
   dsChar_t       policySetName[DSM_MAX_PS_NAME_LENGTH+1];
   /* Active policy set name           */
   dsmDate        polActDate;             /* Policy set activation date       */
   dsChar_t       dfltMCName[DSM_MAX_MC_NAME_LENGTH+1];/* Default Mgmt Class  */
   dsUint16_t     gpBackRetn;             /* Grace-period backup retention    */
   dsUint16_t     gpArchRetn;             /* Grace-period archive retention   */
   dsChar_t       adsmServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* adsm server name */
  dsmBool_t      archiveRetentionProtection; /* is server Retention protection enabled */
   dsUint64_t     maxBytesPerTxn_64;         /* for future use                  */
   dsmBool_t      lanFreeEnabled;            /* lan free option is set         */
   dsmDedupType   dedupType;                 /* server or clientOrServer       */
   dsChar_t       accessNode[DSM_MAX_ID_LENGTH+1];    /* as node node name         */

   /*-----------------------------------------------------------------*/
   /*           Replication and fail over information                 */
   /*-----------------------------------------------------------------*/
   dsmFailOvrCfgType failOverCfgType; /* status of fail over */
   dsChar_t           replServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* repl server name */
   dsChar_t           homeServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* home server name */
   dsChar_t        replServerHost[DSM_MAX_SERVERNAME_LENGTH+1]; /* Network host name of DSM server  */
   dsInt32_t      replServerPort;                              /* Server comm port on host         */

} tsmApiSessInfo;
```

```
#define tsmApiSessInfoVersion  6

/*------------------------------------------------------------------------+
 | Type definition for Query options response on dsmQueryCliOptions()     |
 |      and dsmQuerySessOptions()                                         |
 +------------------------------------------------------------------------*/

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

/*------------------------------------------------------------------------+
 | Type definition for qryRespAccessData  parameter used on dsmQueryAccess()|
 +------------------------------------------------------------------------*/

typedef struct
{
   dsUint16_t       stVersion ;                    /* structure version      */
   dsChar_t          node[DSM_MAX_ID_LENGTH+1];     /* node name              */
   dsChar_t          owner[DSM_MAX_OWNER_LENGTH+1]; /* owner                  */
   tsmObjName        objName ;                      /* object name            */
   dsmAccessType     accessType;                    /* archive or backup     */
   dsUint32_t        ruleNumber ;                   /* Access rule id         */

}tsmQryRespAccessData;

#define tsmQryRespAccessDataVersion  1

/*------------------------------------------------------------------------+
 |  Type definition for envSetUp parameter on dsmSetUp()
 +------------------------------------------------------------------------*/
typedef struct tsmEnvSetUp
{
   dsUint16_t      stVersion;            /* structure version   */
   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;         /* for future use */
   dsmBool_t       reserved2;              /* for future use */
} tsmEnvSetUp;

#define tsmEnvSetUpVersion  4

/*------------------------------------------------------------------------+
 |  Type definition for dsmInitExIn_t
 +------------------------------------------------------------------------*/
typedef struct tsmInitExIn_t
{
   dsUint16_t      stVersion;            /* structure version   */
   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

/*-------------------------------------------------------------------------+
 |  Type definition for dsmInitExOut_t
 +-------------------------------------------------------------------------*/
typedef struct tsmInitExOut_t
{
   dsUint16_t      stVersion;            /* structure version   */
   dsInt16_t          userNameAuthorities;
   dsInt16_t          infoRC;        /* error return code if encountered */
   /* adsm server name                 */
   dsChar_t           adsmServerName[DSM_MAX_SERVERNAME_LENGTH+1];
   dsUint16_t     serverVer;              /* Server's version number      */
   dsUint16_t         serverRel;     /* Server's release number          */
   dsUint16_t     serverLev;              /* Server's level number            */
   dsUint16_t     serverSubLev;           /* Server's sublevel number         */
   dsmBool_t          bIsFailOverMode; /* true if failover has occured */
   dsChar_t           replServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* repl server name */
   dsChar_t           homeServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* home server name */
} tsmInitExOut_t;

#define tsmInitExOutVersion 3

/*-------------------------------------------------------------------------+
 |  Type definition for dsmLogExIn_t
 +-------------------------------------------------------------------------*/
typedef struct tsmLogExIn_t
{
   dsUint16_t      stVersion;            /* structure version   */
   dsmLogSeverity     severity;
   dsChar_t           appMsgID[8];
   dsmLogType  logType;    /* log type : local, server, both */
   dsChar_t           *message;   /* text of message to be logged */
   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

/*-------------------------------------------------------------------------+
 |  Type definition for dsmlogExOut_t
 +-------------------------------------------------------------------------*/
typedef struct tsmLogExOut_t
{
   dsUint16_t      stVersion;            /* structure version   */
} tsmLogExOut_t;

#define tsmLogExOutVersion 1
```

```
/*-------------------------------------------------------------------------+
 |  Type definition for dsmRenameIn_t
 +-------------------------------------------------------------------------*/
typedef struct tsmRenameIn_t
{
   dsUint16_t      stVersion;            /* structure version   */
   dsUint32_t     tsmHandle;   /* handle for session */
   dsUint8_t        repository;                   /* Backup or Archive  */
   tsmObjName       *objNameP ;                   /* object name */
   dsChar_t         newHl[DSM_MAX_HL_LENGTH + 1];  /* new High level name */
   dsChar_t         newLl[DSM_MAX_LL_LENGTH + 1];  /* new Low level name */
   dsmBool_t        merge;                       /* merge into existing name*/
   ObjID            objId;                        /* objId for Archive */
} tsmRenameIn_t;

#define tsmRenameInVersion 1

/*-------------------------------------------------------------------------+
 |  Type definition for dsmRenameOut_t
 +-------------------------------------------------------------------------*/
typedef struct tsmRenameOut_t
{
   dsUint16_t      stVersion;            /* structure version   */
} tsmRenameOut_t;

#define tsmRenameOutVersion 1

/*-------------------------------------------------------------------------+
 |  Type definition for tsmEndSendObjExIn_t
 +-------------------------------------------------------------------------*/
typedef struct tsmEndSendObjExIn_t
{
   dsUint16_t      stVersion;            /* structure version   */
   dsUint32_t     tsmHandle;   /* handle for session */
} tsmEndSendObjExIn_t;

#define tsmEndSendObjExInVersion 1

/*-------------------------------------------------------------------------+
 |  Type definition for dsmEndSendObjExOut_t
 +-------------------------------------------------------------------------*/
typedef struct tsmEndSendObjExOut_t
{
   dsUint16_t      stVersion;            /* structure version   */
   dsStruct64_t       totalBytesSent;      /* total bytes read from app */
   dsmBool_t          objCompressed;       /* was object compressed */
   dsStruct64_t       totalCompressSize;   /* total size after compress */
   dsStruct64_t       totalLFBytesSent;    /* total bytes sent Lan Free */
   dsUint8_t          encryptionType;      /* type of encryption used   */
   dsmBool_t          objDeduplicated;     /* was object processed for dist. data dedup */
   dsStruct64_t       totalDedupSize;      /* total size after de-dup */
}tsmEndSendObjExOut_t;

#define tsmEndSendObjExOutVersion 3

/*-------------------------------------------------------------------------+
 |  Type definition for tsmGroupHandlerIn_t
 +-------------------------------------------------------------------------*/
typedef struct tsmGroupHandlerIn_t
{
   dsUint16_t      stVersion;            /* structure version   */
   dsUint32_t     tsmHandle;   /* handle for session */
   dsUint8_t        groupType;        /* Type of group                          */
   dsUint8_t        actionType;       /* Type of group operation                */
   dsUint8_t        memberType;       /* Type of member: Leader or member       */
   dsStruct64_t     leaderObjId;      /* OBJID of the groupleader               */
   dsChar_t         *uniqueGroupTagP; /* Unique group identifier                */
```

```
   tsmObjName       *objNameP ;       /* group leader object name               */
   dsmGetList       memberObjList;    /* list of objects to remove, assign      */
} tsmGroupHandlerIn_t;

#define tsmGroupHandlerInVersion 1

/*--------------------------------------------------------------------------+
 |  Type definition for tsmGroupHandlerExOut_t
 +--------------------------------------------------------------------------*/
typedef struct tsmGroupHandlerOut_t
{
   dsUint16_t      stVersion;            /* structure version   */
} tsmGroupHandlerOut_t;

#define tsmGroupHandlerOutVersion 1

/*--------------------------------------------------------------------------+
 |  Type definition for tsmEndTxnExIn_t
 +--------------------------------------------------------------------------*/
typedef struct tsmEndTxnExIn_t
{
   dsUint16_t      stVersion;            /* structure version   */
   dsUint32_t     tsmHandle;   /* handle for session */
   dsUint8_t        vote;
} tsmEndTxnExIn_t;

#define tsmEndTxnExInVersion 1

/*--------------------------------------------------------------------------+
 |  Type definition for tsmEndTxnExOut_t
 +--------------------------------------------------------------------------*/
typedef struct tsmEndTxnExOut_t
{
   dsUint16_t      stVersion;            /* structure version   */
   dsUint16_t     reason;            /* reason code                   */
   dsStruct64_t       groupLeaderObjId;        /* groupLeader obj id returned on */
   /* DSM_ACTION_OPEN                */
    dsUint8_t          reserved1;              /* future use                    */
    dsUint16_t         reserved2;              /* future use                    */
} tsmEndTxnExOut_t;

#define tsmEndTxnExOutVersion 1

/*--------------------------------------------------------------------------+
 |  Type definition for tsmEndGetDataExIn_t
 +--------------------------------------------------------------------------*/
typedef struct tsmEndGetDataExIn_t
{
   dsUint16_t      stVersion;            /* structure version   */
   dsUint32_t     tsmHandle;   /* handle for session */
}tsmEndGetDataExIn_t;

#define tsmEndGetDataExInVersion 1

/*--------------------------------------------------------------------------+
 |  Type definition for tsmEndGetDataExOut_t
 +--------------------------------------------------------------------------*/
typedef struct tsmEndGetDataExOut_t
{
   dsUint16_t      stVersion;            /* structure version   */
   dsUint16_t     reason;            /* reason code                   */
   dsStruct64_t   totalLFBytesRecv;  /* total lan free bytes recieved */
}tsmEndGetDataExOut_t;

#define tsmEndGetDataExOutVersion 1

/*--------------------------------------------------------------------------+
```

```
 |  Type definition for  on tsmRetentionEvent()                          |
 +-----------------------------------------------------------------------*/
typedef struct tsmRetentionEventIn_t
{
   dsUint16_t      stVersion;            /* structure version   */
   dsUint32_t         tsmHandle;                  /* session Handle     */
   dsmEventType_t     eventType;                  /* Event type         */
   dsmObjList_t       objList;                    /* object ID          */
}tsmRetentionEventIn_t;

#define tsmRetentionEventInVersion 1

/*-----------------------------------------------------------------------+
 |  Type definition for  on tsmRetentionEvent()                          |
 +-----------------------------------------------------------------------*/
typedef struct tsmRetentionEventOut_t
{
   dsUint16_t       stVersion ;                /* structure version      */
}tsmRetentionEventOut_t;

#define tsmRetentionEventOutVersion 1

/*-----------------------------------------------------------------------+
 |  Type definition for tsmUpdateObjExIn_t                               |
 +-----------------------------------------------------------------------*/
typedef struct tsmUpdateObjExIn_t
{
   dsUint16_t      stVersion;            /* structure version   */
   dsUint32_t         tsmHandle;                  /* session Handle     */
   tsmSendType     sendType;             /* send type back/arch */
   dsChar_t        *descrP;              /* archive description */
   tsmObjName      *objNameP;            /* objName             */
   tsmObjAttr      *objAttrPtr;          /* attribute           */
   dsUint32_t      objUpdAct;            /* update action       */
   ObjID           archObjId;            /* objId for archive   */
}tsmUpdateObjExIn_t;

#define tsmUpdateObjExInVersion 1

/*-----------------------------------------------------------------------+
 |  Type definition for tsmUpdateObjExOut_t                              |
 +-----------------------------------------------------------------------*/
typedef struct tsmUpdateObjExOut_t
{
   dsUint16_t      stVersion;            /* structure version   */
}tsmUpdateObjExOut_t;

#define tsmUpdateObjExOutVersion 1

#if _OPSYS_TYPE == DS_WINNT
#pragma pack()
#endif


#ifdef _MAC
#pragma options align = reset
#endif
#endif /* _H_TSMAPITD */
/*************************************************************************
* Tivoli Storage Manager                                                 *
* API Client Component                                                   *
*                                                                        *
* (C) Copyright IBM Corporation 1993,2010                                *
*************************************************************************/

/*************************************************************************
* Header File Name: dsmapips.h
```

```
*
* Environment:      ********************************************
*                   ** This is a platform-specific source file **
*                   ** versioned for Windows NT                **
*
*                   ********************************************
*
* Design Notes:     This file includes platform dependent definitions
*
* Descriptive-name: Definitions for Tivoli Storage Manager typedefs and LINKAGE
*-------------------------------------------------------------------------*/

#ifndef _H_DSMAPIPS
#define _H_DSMAPIPS


#ifndef _WIN64
#pragma pack(1)
#endif

/*<>*/
/*                    T Y P E D E F S                                   */
/*<>*/

/* new typedef file for Version 3   */


#if !defined(DSMAPILIB) || defined (XOPEN_BUILD)

/* support for linkage */
#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;

/*=== Character and string types ===*/
#ifdef UNICODE
  typedef wchar_t dsChar_t;
  #define dsTEXT(x)       L##x
#else
  typedef char dsChar_t;
  #define dsTEXT(x)       x
#endif  /* !UNICODE */

/*=== Common typedefs and defines derived from dsChar_t ===*/
typedef dsChar_t            *dsString_t;

  /* added for the extended restore order */
   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;        /* Most significant 32 bits. */
      dsUint32_t lo;        /* Least significant 32 bits. */
   } dsUint64_t;
#endif

/*-------------------------------------------------------------------------+
| Type definition for bool_t                                               |
+-------------------------------------------------------------------------*/
/*
 *  Had to create a Boolean type that didn't clash with any other predefined
 *  version in any operating system or windowing system.
 */
typedef enum
{
   dsmFalse = 0x00,
   dsmTrue  = 0x01
}dsmBool_t ;


/*===  for backward compatability  ===*/
#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;        /* Most significant 32 bits. */
   dsUint32_t lo;        /* Least significant 32 bits. */
}dsStruct64_t ;

#endif /* DSMAPILIB */



#ifndef _WIN64
#pragma pack()
#endif
#endif  /* _H_DSMAPIPS */
/*************************************************************************
* Tivoli Storage Manager                                                 *
* Common Source Component                                                *
*                                                                        *
* (C) Copyright IBM Corporation 1993,2016                                *
*************************************************************************/

/*************************************************************************
* Header File Name: release.h
*
* Environment:     ***********************************************
*                  ** This is a platform-independent source file **
*                  ***********************************************
*
```

```
* Design Notes:    This file contains the common information about
*                  the actual version.release.level.sublevel
*
* Descriptive-name: Definitions for Tivoli Storage manager version
*
* Note: This file should contain no LOG or CMVC information. It is
*       shipped with the API code.
*
*-----------------------------------------------------------------------*/

#ifndef _H_RELEASE
#define _H_RELEASE

#define COMMON_VERSION         8
#define COMMON_RELEASE         1
#define COMMON_LEVEL           0
#define COMMON_SUBLEVEL        0
#define COMMON_DRIVER  dsTEXT("")

#define COMMON_VERSIONTXT "8.1.0.0"

#define SHIPYEARTXT  "2016"
#define SHIPYEARTXTW dsTEXT("2016")
#define TSMPRODTXT  "IBM Tivoli Storage Manager"

/*=====================================================================
   The following string definitions are used for VERSION information
   and should not be converted to dsTEXT or osTEXT.  They are used
   only at link time.

   These are also used when the Jar file is built on Unix.  See the
   the perl script tools/unx/mzbuild/createReleaseJava
=====================================================================*/
#define COMMON_VERSION_STR    "8"
#define COMMON_RELEASE_STR    "1"
#define COMMON_LEVEL_STR      "0"
#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

/*=====================================================================
   Internal version, release, and level (build) version.  This
   should be unique for every version+release+ptf of a product.
   This information is recorded in the file attributes and data
   stream for diagnostic purposes.
   NOTE: DO NOT MODIFY THESE VALUES. YOU CAN ONLY ADD NEW ENTRIES!
=====================================================================*/
#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           COMMON_BUILD_TSM_810

/*=== define VRL as an Int for bitmap version compares ====*/
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_810 = 810;

#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 */
```

Apéndice C. Archivo de origen de las definiciones de la función de la API

Este apéndice contiene el archivo de encabezado `dsmapifp.h` para poder ver las definiciones de la función de API.

Nota: **DSMLINKAGE** se define de forma distinta en cada sistema operativo. Consulte en el archivo `dsmapips.h` las definiciones que se corresponden con su sistema operativo específico.

La información que se proporciona aquí contiene una copia puntual de los archivos que se distribuyen con la API. Vea los archivos en el paquete de distribución de la API para la última versión.

```
/*************************************************************************
* Tivoli Storage Manager                                                 *
* API Client Component                                                   *
*                                                                        *
* (C) Copyright IBM Corporation 1993,2002                                *
*************************************************************************/

/*************************************************************************/
/* Header File Name: dsmapifp.h                                          */
/*                                                                       */
/* Descriptive-name: Tivoli Storage Manager API function prototypes      */
/*************************************************************************/
#ifndef _H_DSMAPIFP
#define _H_DSMAPIFP

#if defined(__cplusplus)
extern "C" {
#endif

#ifdef DYNALOAD_DSMAPI

/* function will be dynamically loaded */
#include "dsmapidl.h"

#else

/* functions will be implicitly loaded from library */

/*************************************************************************/
/*              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 */
```

Esta sección contiene las definiciones de función de API. Es una copia del archivo de encabezado de `tsmapifp.h`.

Nota: DSMLINKAGE se define de forma distinta para cada sistema operativo. Consulte las definiciones en el archivo `tsmapips.h` para el sistema operativo específico.

```
/*************************************************************************
* Tivoli Storage Manager                                                 *
* API Client Component                                 *
*                                                      *
```

```
* (C) Copyright IBM Corporation 1993,2002              *
*********************************************************************/

/*********************************************************************/
/* Header File Name: tsmapifp.h                                      */
/*                                                                   */
/* Descriptive-name: Tivoli Storage Manager API function prototypes       */
/*********************************************************************/
#ifndef _H_TSMAPIFP
#define _H_TSMAPIFP

#if defined(__cplusplus)
extern "C" {
#endif

#ifdef DYNALOAD_DSMAPI

/* function will be dynamically loaded */
#include "dsmapidl.h"

#else

/* functions will be implicitly loaded from library */

/*===================================================================*/
/*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 */
```

Apéndice D. Funciones de accesibilidad para la familia de productos IBM Spectrum Protect

Las funciones de accesibilidad ayudan a aquellos usuarios que tienen una discapacidad, como, por ejemplo, movilidad reducida o poca visión, a utilizar productos tecnológicos de información de forma satisfactoria.

Visión general

La familia de productos de IBM Spectrum Protect incluye las siguientes funciones de accesibilidad mayores:

- Funcionamiento utilizando solo el teclado
- Operaciones que utilizan un lector de pantalla

La familia de productos de IBM Spectrum Protect utiliza el estándar W3C más reciente, WAI-ARIA 1.0 (www.w3.org/TR/wai-aria/), para asegurar la conformidad con US Section 508 (www.access-board.gov/guidelines-and-standards/communications-and-it/about-the-section-508-standards/section-508-standards) y Web Content Accessibility Guidelines (WCAG) 2.0 (www.w3.org/TR/WCAG20/). Para aprovechar las características de accesibilidad, utilice el release más reciente del lector de pantalla y el navegador web más reciente soportados por el producto.

La documentación del producto en IBM Knowledge Center está habilitada para la accesibilidad. Las funciones de accesibilidad del IBM Knowledge Center se describen en la Sección de accesibilidad de la ayuda del IBM Knowledge Center (www.ibm.com/support/knowledgecenter/about/releasenotes.html?view=kc#accessibility).

Navegación mediante teclado

Este producto utiliza teclas estándar de navegación.

Información sobre interfaces

Las interfaces de usuario no tienen contenido que se actualiza de 2 a 55 veces por segundo.

Las interfaces de usuarios web se basan en las hojas de estilo en cascada para representar el contenido correctamente y para proporcionar una experiencia que se pueda utilizar. La aplicación proporciona un método equivalente para usuarios con problemas de poca visión para utilizar los parámetros de visualización del sistema, incluido el modo de alto contraste. Puede controlar el tamaño de fuente utilizando los parámetros del dispositivo o del navegador web.

Las interfaces de usuarios web incluyen puntos de referencia de navegación WAI-ARIA que puede utilizar para navegar rápidamente a áreas funcionales de la aplicación.

Software del proveedor

La familia de productos IBM Spectrum Protect incluye cierto software del proveedor que no está cubierto por el acuerdo de licencia de IBM. IBM no es responsable de las características de accesibilidad de estos productos. Póngase en contacto con el proveedor para obtener información sobre accesibilidad relacionada con sus productos.

Información de accesibilidad relacionada

Además del centro de atención al cliente de IBM y de los sitios web de soporte estándar, IBM dispone de un servicio telefónico TTY que permite a clientes sordos o con dificultades auditivas acceder a los servicios de ventas y asistencia técnica.

Servicio TTY
800-IBM-3383 (800-426-3383)
(en América del Norte)

Para obtener más información acerca del compromiso que IBM tiene con la accesibilidad, consulte IBM Accessibility(www.ibm.com/able).

Avisos

Esta información se ha desarrollado para productos y servicios que se ofrecen en EE.UU. Es posible que este material esté disponible en otros idiomas en IBM. Sin embargo, es posible que tenga obligación de tener una copia del producto o de la versión del producto en dicho idioma para acceder a él.

IBM puede no ofrecer los productos, servicios o funcionalidades tratados en este documento en otros países. Consulte al representante local de IBM para obtener más información sobre los productos y servicios que están disponibles actualmente en su zona. Las referencias a programas, productos o servicios de IBM no pretenden establecer ni implicar que sólo puedan utilizarse dichos productos, programas o servicios de IBM. También se puede utilizar otro producto, programa o servicio que tenga la misma función y no infrinja el derecho de propiedad intelectual de IBM. No obstante, es responsabilidad del usuario evaluar y verificar el funcionamiento de cualquier producto, programa o servicio que no sea de IBM.

IBM puede tener patentes o solicitudes de patentes pendientes que cubran la materia descrita en este documento. El presente documento no le confiere ningún derecho sobre estas patentes. Puede enviar por escrito consultas acerca de las licencias a :

IBM Director of Licensing
IBM Corporation
North
Castle Drive, MD-NC119
Armonk, NY 10504-1785
EE.UU.

Si desea realizar consultas acerca de la información de juegos de caracteres de doble byte (DBCS), puede ponerse en contacto con el Departamento de Propiedad Intelectual de IBM de su país o bien enviar las consultas por escrito a:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokio 103-8510, Japón

INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN "TAL CUAL" SIN GARANTÍA DE NINGUNA CLASE, NI EXPLÍCITA NI IMPLÍCITA, INCLUIDAS, PERO SIN LIMITARSE A, LAS GARANTÍAS O CONDICIONES IMPLÍCITAS DE NO VULNERACIÓN DE DERECHOS, COMERCIALIZACIÓN O IDONEIDAD PARA UN FIN DETERMINADO. Algunos estados no autorizan la exclusión de garantías explícitas o implícitas en determinadas transacciones, por lo que es posible que este aviso no sea aplicable en su caso.

Esta información puede contener imprecisiones técnicas o errores tipográficos. Periódicamente se efectúan cambios en la información aquí contenida; estos cambios se incorporarán en nuevas ediciones de la publicación. IBM puede realizar en cualquier momento mejoras o cambios en los productos o programas descritos en esta publicación sin previo aviso.

Las referencias contenidas en esta información a sitios web no IBM solo se proporcionan por comodidad y de ningún modo constituyen un aval de esos sitios web. Los materiales de estos sitios web no forman parte de los materiales para este producto IBM y el uso de estos sitios web es responsabilidad del usuario.

IBM puede utilizar o distribuir la información que se le proporcione, en la forma que crea conveniente, sin incurrir por ello en ninguna obligación con el remitente.

Los propietarios de licencia de este programa que deseen obtener información acerca del mismo con el fin de: (i) intercambiar la información entre programas creados independientemente y otros programas (incluido éste) y (ii) compartir la información que se ha intercambiado, deben ponerse en contacto con:

IBM Director of Licensing
IBM Corporation
North
Castle Drive, MD-NC119
Armonk, NY 10504-1785
EE.UU.

Esta información puede estar disponible, sujeta a los términos y condiciones adecuados, incluyendo en algunos casos el pago de una tarifa.

El programa bajo licencia descrito en este documento y todo el material bajo licencia disponible se proporcionan bajo los términos de IBM Customer Agreement, IBM International Program License Agreement o cualquier otro acuerdo equivalente entre IBM y el cliente.

Los datos de rendimiento aquí mencionados se han obtenido en condiciones de funcionamiento específicas. Los resultados reales pueden variar.

La información relativa a productos que no son de IBM se ha obtenido de los proveedores de estos productos, sus anuncios publicados y otras fuentes públicamente disponibles. IBM no ha realizado pruebas de estos productos y no puede confirmar la exactitud de la información con respecto a su rendimiento, compatibilidad u otros aspectos relacionados con los productos que no sean de IBM. Las preguntas relativas a las posibilidades de productos no IBM deben dirigirse a los suministradores de esos productos.

Esta información contiene ejemplos de datos e informes utilizados en operaciones comerciales cotidianas. Para ilustrarlos de la forma más completa posible, se han utilizado nombres de personas, empresas, marcas y productos. Todos estos nombres son ficticios y cualquier parecido con nombres y direcciones de una empresa real es pura coincidencia.

LICENCIA DE COPYRIGHT:

Esta información contiene programas de aplicación de ejemplo en código fuente, que ilustran técnicas de programación en diferentes plataformas operativas. Puede copiar, modificar y distribuir libremente estos programas de muestra, sin pagar costo alguno aIBM, con la finalidad de desarrollar, utilizar, comercializar o distribuir programas de aplicación destinados a la interfaz de programación de aplicaciones para la plataforma operativa para la que se han escrito los programas de muestra. Estos ejemplos no se han probado exhaustivamente bajo todas las condiciones. Por tanto, IBM, no puede garantizar ni implicar la fiabilidad, utilidad o función de estos programas. Los programas de ejemplo se proporcionan "TAL

CUAL" y sin garantía de ninguna clase. IBM no será responsable de ningún daño que surja del uso de los programas de muestra.

Cada copia o fragmento de estos programas de ejemplo o cualquier trabajo derivado deben incluir un aviso de copyright como el siguiente: © (nombre de su empresa) (año). Partes de este código derivan de programas de ejemplo de IBM Corp. © Copyright IBM Corp. _escriba el año o años_.

Marcas registradas

IBM, el logotipo de IBM y ibm.com son marcas registradas de International Business Machines Corp., que se encuentran registradas en un gran número de jurisdicciones en todo el mundo. Otros nombres de productos y servicios pueden ser marcas registradas de IBM o de otras empresas. Hay disponible una lista actual de marcas registradas de IBM en la web, en sección "Copyright and trademark information" de www.ibm.com/legal/copytrade.shtml.

Adobe es una marca registrada de Adobe Systems Incorporated en Estados Unidos o en otros países.

Linear Tape-Open, LTO y Ultrium son marcas registradas de HP, IBM Corp. y Quantum en EE.UU. y en otros países.

Intel y Itanium son marcas registradas de Intel Corporation o sus empresas filiales en Estados Unidos y otros países.

Linux es una marca registrada de Linus Torvalds en Estados Unidos o en otros países.

Microsoft, Windows y Windows NT son marcas registradas de Microsoft Corporation en Estados Unidos y/o en otros países.

Java™ y todas las marcas registradas y los logotipos basados en Java son marcas registradas de Oracle y/o sus filiales.

SoftLayer es una marca registrada de SoftLayer, Inc., una empresa de IBM.

UNIX es una marca registrada de The Open Group en Estados Unidos y en otros países.

Términos y condiciones de la documentación de producto

Los permisos para la utilización de estas publicaciones se otorgan sujetos a los siguientes términos y condiciones.

Ámbito de aplicación

Estos términos y condiciones completan los términos y condiciones de uso del sitio web de IBM.

Uso personal

Puede reproducir estas publicaciones para su uso personal, no comercial, siempre y cuando se conserven todos los avisos de propiedad. No podrá distribuir, mostrar, ni crear trabajo derivado de estas publicaciones, o cualquier parte de éstas, sin el consentimiento expreso de IBM.

Uso comercial

Puede reproducir, distribuir y visualizar estas publicaciones únicamente en

su empresa siempre que se conserven todos los avisos de propiedad. No puede realizar trabajos derivados de estas publicaciones ni reproducir, distribuir o visualizar estas publicaciones ni parte de las mismas fuera de la empresa sin el consentimiento expreso de IBM.

Derechos

Si no se indica lo contrario en este permiso, no se otorgan otros permisos, licencias o derechos, ya sea de forma expresa o implícita, a las publicaciones u otra información, datos, software u otra propiedad intelectual que contenga este documento.

IBM se reserva el derecho de retirar los permisos aquí concedidos cuando lo desee, siempre que el uso de las publicaciones vaya en detrimento de su interés o, según determine IBM, si no se cumplen correctamente las instrucciones anteriores.

Queda prohibido descargar, exportar o reexportar esta información si no se cumplen íntegramente todas las leyes aplicables y regulaciones, incluyendo las leyes y regulaciones de exportación de los Estados Unidos.

IBM NO EFECTÚA NINGÚN TIPO DE GARANTÍA SOBRE EL CONTENIDO DE ESTAS PUBLICACIONES. LAS PUBLICACIONES SE SUMINISTRAN "TAL CUAL", SIN GARANTÍAS DE NINGÚN TIPO, NI EXPLÍCITAS NI IMPLÍCITAS, INCLUIDAS LAS GARANTÍAS IMPLÍCITAS DE COMERCIALIZACIÓN O IDONEIDAD PARA UN PROPÓSITO DETERMINADO, SIN LIMITARSE A ELLAS.

Consideraciones sobre la política de privacidad

Los productos de IBM Software, incluido el software como soluciones de servicio, (“Ofertas de software”) podrían utilizar cookies u otras tecnologías para recopilar información del uso del producto para ayudar a mejorar la experiencia del usuario final, para adaptar las interacciones con el usuario final o para otros fines. En muchos casos, las Ofertas de software no recopilan información de identificación personal. Algunas de nuestras ofertas de software pueden ayudarle a recopilar información de identificación personal. Si esta oferta de software utiliza cookies para recopilar información de identificación personal, la información específica sobre la utilización de cookies de esta oferta se expone más adelante.

Esta oferta de software no utiliza cookies u otras tecnologías para recopilar información de identificación personal.

Si las configuraciones desplegadas para esta Oferta de software le ofrecen como cliente la posibilidad de recopilar información de identificación personal de los usuarios finales mediante cookies y otras tecnologías, debe buscar asesoramiento legal sobre las leyes aplicables a dicha recopilación de datos, incluidos los requisitos de aviso y consentimiento.

Para obtener más información sobre el uso de las distintas tecnologías, incluidas las cookies, para estos fines, consulte la Política de privacidad de IBM en http://www.ibm.com/privacy y la Declaración de privacidad en línea de IBM en http://www.ibm.com/privacy/details en la sección titulada “Cookies, Web Beacons and Other Technologies” e “IBM Software Products and Software-as-a-Service Privacy Statement” en http://www.ibm.com/software/info/product-privacy.

Glosario

Hay un glosario disponible con términos y definiciones para la familia de productos de IBM Spectrum Protect.

Consulte la publicación Glosario de IBM Spectrum Protect.

Para ver los glosarios de otros productos de IBM, consulte Terminología de IBM.

Índice

Números

64 bits
  compilar 1
  requisitos 1

A

acceder a objetos
  en diferentes nodos 26
acceso a objetos
  por usuario 25
agregación de archivo 38
agrupación de archivos 66
agrupaciones de almacenamiento
  operaciones de grabación simultánea 39
ajustar tamaño de objetos 42
anotación de eventos 78
API
  aplicaciones de ejemplo 5
  cadena de opción utilizada por dsmInitEx 2
  configuración del entorno 3
  dsmInitEx
    archivo de configuración utilizado por 3
  utilizar Unicode 87
  visión general 1
aplicación de ejemplo
  callmt1.c 16
aplicaciones API de ejemplo
  callbuff 5
  callbuff - almacenamiento intermedio de datos 5
  callevnt 5
  callevnt - retención basada en evento 5
  callhold 5
  callhold - retención de anulación 5
  callmt* 5
  callmt* - aplicaciones API de ejemplo de varias hebras 5
  callret 5
  callret - aplicaciones API de ejemplo de protección de retención de datos 5
  dapi* 5
  dapi* - interactiva, hebra única 5
  dsmgrp 5
  dsmgrp* - ejemplo de agrupación de objeto 5
aplicaciones de API de ejemplo
  UNIX o Linux 5
  Windows 64 bits 7
aplicaciones de la API de ejemplo
  callmtu1.c 87
  callmtu2.c 87
archivar objetos 43
archiveretentionprotection 32
archivo de configuración
  API 3
Archivo de configuración de la API
  utilizado por dsmInitEx 17
archivo de encabezado dsmapips.h 165
archivo de encabezado dsmapitd.h 165
archivo de encabezado release.h 165
archivo de encabezado tsmapifp.h 211
archivo de encabezado tsmapitd.h 165
archivos
  configuración 1
  opción 1
  tipo de objeto 25
archivos archivados
  cuánto tiempo retenidos 29
archivos de cabecera
  dsmapifp.h 207
  dsmrc.h 155
  tsmapifp.h 211
archivos de opciones
  usuario 3
archivos en la eliminación de datos duplicados
  excluir 58
  incluir 58
asnodename 85
autorización de propietario 22
autorización de propietario de cliente 22

B

basado en evento
  política de retención 33

C

cadena de opción
  API 2
cadena de opciones
  fromowner 26
calificador de alto rango 81
calificador de bajo rango 81
callbuff
  aplicaciones API de ejemplo de almacenamiento intermedio de datos de IBM Spectrum Protect 5
callevnt
  retención basada en eventos 5
callhold
  aplicaciones API de ejemplo de retención de anulación 5
callmt*
  aplicaciones API de ejemplo de varias hebras 5
callmt1.c
  ejemplo 16
callret
  aplicaciones API de ejemplo de protección de retención de datos 5
capacidad
  espacio de archivos 26
Centro de conocimientos v
cifrado
  aplicación gestionada 49
  configuración de autenticación 49
  interoperatividad 84
  transparente 51
cifrado y compresión que utiliza eliminación de copia de almacenamiento intermedio 48
clase de gestión
  consultar 30
  dsmBindMC, asignada por 29
  objetos asociados 29
  vincular y revincular a archivos 30
cliente de archivado y copia de seguridad
  interoperatividad 81
código de ejemplo
  dsmgrp.c 68
códigos de retorno
  archivo de cabecera de origen 155
  obtener a través de dsmRCMsg 135
compatibilidad
  entre versiones de la API 14
compilar
  Unicode 87
compresión 44, 69
compresión LZ4 45
compresión LZW 45
compressalways 2
  opción 7
configurar acceso 83
conjunto de caracteres de doble byte 87
conjuntos de caracteres 87
consideraciones acerca del rendimiento 39
  función dsmSendData 39
consulta
  actlog 128
  mandato 83
  nodos con autorización de nodo proxy de cliente 85
consultas, sistema 34
consultas de ruta de acceso rápida 96
consultas del sistema 34
control de versiones
  dsmQueryApiVersionEx, utilizando 14
  estructuras de datos de la API 15
  gestionar copias de seguridad 43
copia de seguridad
  utilizar el proxy de nodo de cliente 85
  varios nodos 85
copias de objetos activas 43
copias de objetos inactivas 43
CTRL+C 16

D

dapi*
  hebra única, aplicaciones API de ejemplo interactivas 5
DBCS 87
definiciones de función, API 207, 211
denominación de objetos
  alto rango
    nombre de objeto 24
  bajo rango
    nombre de objeto 24
  dsmBindMC 24
  ejemplos por OS 25
  interoperatividad 81
  nombre de espacio de archivo 24
  tipo de objeto 25
  visión general 23
desactivar objetos 77
descriptores de señales 16
desduplicación de datos 52
detener una sesión 17
diagrama de estado
  ejemplo de copia de seguridad y archivado 62
  restauración y recuperación 74
dir
  tipo de objeto 25
dirección TCPserver 19
discapacidad 217
dscenu.txt 3
dsierror.log 4
dsierror.log. 4
DSM_MAX_PLATFORM_LENGTH 17
dsm.opt 1
  asnodename, opción 85
  enablearchiveretentionprotection 33
  encryptkey 49
dsm.sys 1, 3, 19
  asnodename, opción 85
  enablearchiveretentionprotection 33
  encryptkey 49
dsmapifp.h
  archivo de cabecera 91, 207
dsmapitd.h 14, 15, 121, 124
  archivo de cabecera 131
dsmBeginGetData función
  diagrama de estado 78
dsmBeginQuery función
  diagrama de estado 78
dsmBeginTxn 26
dsmBeginTxn función
  diagrama de estado 78
dsmBindMC
  ejemplo 31
dsmBindMC función
  diagrama de estado 78
dsmChangePW
  descripción general 78
dsmChangePW función
  códigos de retorno 103
  diagrama de estado 78
  sintaxis 102
dsmclientV3.cat 3
dsmDeleteFS función
  diagrama de estado 78
dsmDeleteObj función
  diagrama de estado 78
dsmEndGetData
  detener el proceso 73
dsmEndGetData función
  diagrama de estado 78
dsmEndQuery 34
  descripción general 34
dsmEndQuery función
  diagrama de estado 78
dsmEndSendObj función
  diagrama de estado 78
dsmEndTxn función
  diagrama de estado 78
  modelo de transacción 37
dsmGetData 73
dsmGetData función
  diagrama de estado 78
dsmGetNextObj
  función dsmDeleteObj 105
dsmGetNextQObj 34
  función dsmEndQuery 108
dsmGetNextQObj función
  diagrama de estado 78
dsmGetNextQObj function
  función dsmRetentionEvent 140
dsmGetObj
  recibir objetos 73
dsmGetObj función
  diagrama de estado 78
dsmgrp*
  aplicaciones API de ejemplo que agrupan a objetos lógicos 5
dsmgrp.c 68
dsmHandle 133, 134
DSMI_CONFIG, variable de entorno 3
DSMI_DIR
  variable de entorno 7
DSMI_DIR, variable de entorno 3
DSMI_LOG, variable de entorno 4
dsmQuerySessInfo
  función dsmDeleteFS 104
dsmQuerySessInfo función
  diagrama de estado 78
dsmrc.h
  archivo de cabecera 155
dsmRegisterFS función
  diagrama de estado 78
dsmSendData función
  diagrama de estado 78
dsmSendObj
  política de retención 33
dsmSendObj función
  diagrama de estado 78
  ejemplo de código 64
  sintaxis 144
  visión general 143
dsmtca
  control de versiones 14
dsmTerminate 73
dsmUpdateFS función
  diagrama de estado 78

E

ejemplos de ruta
  por OS 25
eliminación de copia de almacenamiento intermedio
  restauración y recuperación 47
  visión general 46
eliminación de datos duplicados del lado del servidor 58
eliminación de duplicados de datos del lado del cliente 54
eliminar archivado 83
eliminar espacios de archivo 83
enablearchiveretentionprotection 33
  dsm.opt 33
  dsm.sys 33
encryptkey 49
entorno
  configurar una API 3
enviar datos
  a espacios de archivo que no son Unicode 88
enviar datos a un servidor 37
envSetUp 148
errorlogretention
  cuándo se debe utilizar 78
espacio de archivos
  capacidad 26
  gestionar 26
  registro 26
  supresión 26
espacios de archivo
  no Unicode 88
estado
  InSession 129
estado de réplica 59
estado InSession 128, 129
estimaciones de tamaño 42
estructura
  función qryRespFSData 26
  qryRespBackupData 34
estructura qMCData 35
estructura qryRespBackupData 34
estructuras
  límites de tamaño 14, 36
  qMCData 35
estructuras de datos
  control de versiones 15
  límites de tamaño 14, 36
evento
  eventRetentionActivate 33
evento eventRetentionActivate 33
excluir archivos de la eliminación de datos duplicados 58
excluir objetos 24

F

fin de una sesión 17
  con dsmTerminate 18
fromowner, opción 26
fuera de LAN
  función dsmEndGetDataEX 107
función dsmApiVersion
  sesión 17
función dsmBeginGetData 69, 73
  códigos de retorno 95
  diagrama de estado 74
  ejemplo de código 75
  en organigrama 74

función dsmBeginGetData *(continuación)*
función dsmEndGetData 107
función dsmTerminate 107
gestión de almacenamiento intermedio 47
sintaxis 94
visión general 94
función dsmBeginQuery
códigos de retorno 99
consultar 34
diagrama de estado 34
diagrama de flujo 34
ejemplo de consulta 35
enviar ejemplos de datos 37
recibir datos 70
sintaxis 95
visión general 95
función dsmBeginTxn
caducidad 31
códigos de retorno 101
ejemplo de código 64
eliminación 31
eliminación de copia de almacenamiento intermedio 46
eliminar objetos 77
función dsmEndTxn 110
función dsmRenameObj 137
función dsmRetentionEvent 140
modelo de transacción 37
política de retención 33
sintaxis 101
visión general 100
función dsmBindMC
clases de gestión 30
códigos de retorno 102
descripción general 63
ejemplo de código 64
eliminación de copia de almacenamiento intermedio 46
función dsmSendObj 143
información devuelta por 29
lista de inclusión/exclusión 29
nombre de objetos 24
sintaxis 101
visión general 101
función dsmChangePW
seguridad de sesión 18
visión general 102
función dsmCleanUp
función dsmSetUp 148
señales 16
sintaxis 103
varias hebras 16
visión general 103
función dsmDeleteAccess
acceder a objetos 26
sintaxis 104
visión general 104
función dsmDeleteFS
código de ejemplo 26
códigos de retorno 105
espacios de archivo 26
gestión de sistema de archivo 27
sintaxis 105
visión general 104
función dsmDeleteObj
códigos de retorno 106
función dsmDeleteObj *(continuación)*
denominación de objetos 11
eliminar objetos 77
función dsmEndTxn 110
función dsmSendObj
clase de gestión 11
objetos 43
sintaxis 106
visión general 105
función dsmEndGetData 69
diagrama de estado 74
ejemplo de código 75
en organigrama 74
gestión de almacenamiento intermedio 47
sin LAN 38
sintaxis 107
visión general 107
función dsmEndGetDataEx
sintaxis 107
visión general 107
función dsmEndGetObj 69
códigos de retorno 108
diagrama de estado 74, 78
ejemplo de código 75
en organigrama 74
función dsmBeginGetData 94
gestión de almacenamiento intermedio 47
sintaxis 108
visión general 108
función dsmEndQuery 35
consultar al servidor 70
diagrama de estado 34
diagrama de flujo 34
función dsmGetNextQObj 115
sintaxis 108
visión general 108
función dsmEndSendObj
códigos de retorno 109
diagrama de estado 62
ejemplo de código 64
enviar objetos 42
función dsmEndTxn 110
función dsmSendData 142
función dsmSendObj 143
organigrama 62
sintaxis 109
visión general 109
función dsmEndSendObjEx 46
cifrado 49
códigos de retorno 110
compresión 44
sin LAN 38
sintaxis 109
visión general 109
función dsmEndTxn 31, 140
agrupación de archivos 66
códigos de retorno 111
diagrama de estado 62
ejemplo de código 64
eliminación de copia de almacenamiento intermedio 46
eliminar objetos 77
función dsmEndTxnEx 111
función dsmRenameObj 137
función dsmRetentionEvent 140
función dsmEndTxn *(continuación)*
función dsmSendObj 143
operaciones de grabación simultánea 39
organigrama 62
sintaxis 110
visión general 110
función dsmEndTxnEx
agrupación de archivos 66
códigos de retorno 112
sintaxis 111
visión general 111
función dsmEventType
política de retención 33
función dsmGetBufferData 47
códigos de retorno 114
sintaxis 114
visión general 114
función dsmGetData
códigos de retorno 113
ejemplo de código 75
en el diagrama de estado 74
en organigrama 74
sintaxis 113
visión general 113
función dsmGetDataEx
función dsmReleaseBuffer 137
función dsmRequestBuffer 139
función dsmGetList
función dsmGetObj 118
función dsmGetNextQObj 32, 34, 59
códigos de retorno 117
diagrama de estado 34
diagrama de flujo 34
ejemplo de consulta 35
sintaxis 115
visión general 115
función dsmGetObj 69
códigos de retorno 119
diagrama de estado 74
ejemplo de código 75
en organigrama 74
función dsmBeginGetData 94
función dsmEndGetObj 108
función dsmGetData 113
sintaxis 119
visión general 118
función dsmGroupHandler
agrupación de archivos 66
códigos de retorno 120
función dsmEndTxnEx 111
sintaxis 119
visión general 119
función dsmHandle
sesión 17
función dsmInit
códigos de retorno 123
protección de retención 32
sintaxis 120
visión general 120
función dsmInitEx 26, 46
asnodename, opción 85
cadena de opción 2
cifrado 49
códigos de retorno 127
contraseña caducada 18
diagrama de estado 78

función dsmInitEx *(continuación)*
dsmQuerySessOptions 134
especificar opciones 2
función dsmChangePW 102
función dsmEndGetData 107
función dsmGetBufferData 114
función dsmGetNextQObj 115
función dsmLogEvent 128
función dsmQueryCliOptions 132
función dsmReleaseBuffer 137
función dsmSetUp 148
iniciar una sesión 17
interoperatividad 84
propietario de sesión, configurar 25
protección de retención 32
seguridad de sesión 18
sesión 17
sintaxis 124
usuario administrativo 22
varias hebras 16
visión general 124
función dsmIntitEx
función dsmQuerySessInfo 133
función dsmLogEvent
códigos de retorno 129
sintaxis 128
visión general 128
función dsmLogEventEx 78
códigos de retorno 130
sintaxis 129
visión general 129
función dsmQuery
varios nodos 85
función dsmQueryAccess 26
función dsmDeleteAccess 104
visión general 130
función dsmQueryApiVersion
diagrama de estado 78
sintaxis 131
visión general 131
función dsmQueryApiVersionEx
control de versiones 14
sintaxis 132
visión general 131
función dsmQueryAPIVersionEx
varias hebras 16
función dsmQueryCliOptions
dsmQuerySessOptions 134
sesión 17
sintaxis 132
visión general 132
función dsmQuerySessInfo
códigos de retorno 133
descripción general 18
función dsmRetentionEvent 140
modelo de transacción 37
sintaxis 133
visión general 133
función dsmQuerySessOptions
sintaxis 134
visión general 134
función dsmRCMsg
códigos de retorno 135
sintaxis 135
visión general 135
función dsmRegisterFS
código de ejemplo 26
función dsmRegisterFS *(continuación)*
códigos de retorno 136
espacios de archivo 26
sintaxis 136
visión general 136
función dsmReleaseBuffer 46, 47
códigos de retorno 137
función dsmGetBufferData 114
función dsmReleaseBuffer 137
función dsmRequestBuffer 139
función dsmSendBufferData 141
sintaxis 137
Función dsmReleaseBuffer
visión general 137
función dsmRenameObj
códigos de retorno 139
sintaxis 138
visión general 137
función dsmRequestBuffer
códigos de retorno 140
eliminación de copia de almacenamiento intermedio 46
sintaxis 139
visión general 139
función dsmRetentionEvent
caducidad 31
códigos de retorno 141
eliminación 31
política de retención 33
sintaxis 140
visión general 140
función dsmSendBufferData
códigos de retorno 142
eliminación de copia de almacenamiento intermedio 46
sintaxis 141
visión general 141
función dsmSendData
códigos de retorno 143
compresión 44
diagrama de estado 62
ejemplo de código 64
enviar objetos 42
función dsmEndSendObj 109
función dsmEndTxn 110
función dsmSendObj 143
organigrama 62
rendimiento 39
sintaxis 142
varias hebras 16
visión general 142
función dsmSendObj 34
acceder a objetos 25
compresión 44
denominación de objetos 11
eliminar objetos 77
en el diagrama de estado 62
enviar objetos 42
función dsmEndTxn 110
grupo de copias de seguridad 30
grupos de copia 29
organigrama 62
política de retención 33
función dsmSendType
actualizar objetos 76
función dsmSetAccess
acceder a objetos 26
función dsmSetAccess *(continuación)*
códigos de retorno 147
sintaxis 147
visión general 147
función dsmSetUp
passwordaccess 21
sin LAN 11, 38
sintaxis 148, 149
varias hebras 16, 38
visión general 148
función dsmTerminate
almacenamiento intermedio 46
descripción general 18
diagrama de estado 78
eliminación de copia de almacenamiento intermedio 46
función dsmInit 120
función dsmReleaseBuffer 137
función dsmRequestBuffer 139
función dsmSetUp 148
señales 16
sesión 17
sintaxis 150
visión general 150
función dsmUpdateFS
código de ejemplo 26
códigos de retorno 151
espacios de archivo 26
gestión de espacio de archivo 27
sintaxis 150
visión general 150
función dsmUpdateObj
cambiar clase de gestión 29
códigos de retorno 152
sintaxis 151
visión general 151
función dsmUpdateObject(Ex)
actualizar objetos 76
función dsmUpdateObjEx
cambiar clase de gestión 29
códigos de retorno 154
sintaxis 153
visión general 152
función rcApiOut
sesión 17
función smBeginQuery
clase de gestión 30
función dsmEndQuery 108
función dsmGetNextQObj 115
funciones de accesibilidad 217

G

gestión de espacio de archivo
dsmUpdateFS 27
gestión de sistema de archivo
dsmDeleteFS 27
grupo de copia 29
grupo de copias archivadas 29
grupo de copias de seguridad 29

I

IBM Knowledge Center v
id de objetos, visión general 23

incluir archivos en la eliminación de datos duplicados 58
incluir objetos 24
inclusión/exclusión
archivo 149
información de ruta
interoperatividad 81
iniciar una sesión 17
interoperatividad
acceso a objetos de la API 81
asignar nombres a los objetos de la API 81
cliente de archivado y copia de seguridad 81
convenios
UNIX o Linux 81
Windows 81
mandatos 83
sistema operativo 84
interoperatividad del sistema operativo 84

L

líder de grupo 66
límites de tamaño
estructuras de datos de la API 14, 36
lista de inclusión/exclusión 29, 89
lista de opciones
formato 122, 125
Lista de opciones de la API
utilizado por dsmInitEx 17
llamadas de función
descripciones breves 91

M

makemtu 87
mandatos
makemtu 87
mbcs 87
mensajes
función dsmRCMsg 135
metadatos
denominación de objetos 23
migración automática de cliente 59
modelo de transacción
función dsmBeginTxn 100

N

nodos
acceder a distintos propietarios 26
autorización 77
con soporte de proxy de cliente 85
consultar clases de gestión 30
nombres 11
nodos destino y nodos tradicionales 85
nombre de alto nivel
función dsmRenameObj 137
nombre de espacio de archivo
agregación de archivo 38
visión general 24
nombre de propietario 11, 25
NULO 25
nombres de nivel inferior
función dsmRenameObj 137
NULO
grupo archivado o copia de seguridad 29

O

objeto
control de versiones 43
objetos
actualización 76
ciclo de caducidad 77
copias activas 43
copias inactivas 43
desactivar 77
eliminar del servidor 77
reglas de acceso 25
supresión 76
objetos de archivado
caducidad 31
liberar 31
suspender 31
opciones
compressalways 2
configuradas por el administrador 2
enablearchiveretentionprotection 33
errorlogretention 78
fromnode 26
fromowner 26
no compatibles en API 1
passwordaccess 16, 120
servername 2
tcpbuffsize 39
tcpnodelay 39
tcpserveraddr 2
opciones de administrador 2
opciones del supervisor de rendimiento de cliente
PERFCOMMTIMEOUT 42
PERFMONTCPPORT 41
PERFMONTCPSERVERADDRESS 41
Operación DB Chg 11
operaciones de grabación simultánea
agrupaciones de almacenamiento 39
ordenar objetos
por orden de restauración 70
organigrama
ejemplo de copia de seguridad y archivado 62
restauración y recuperación 74
orígenes de configuración
secuencia de prioridad 2

P

páginas de código 87
passwordaccess
generate 149
opción 7, 11, 49
passwordaccess, opción
función dsmInit 120
generate 18
sin TCA 21
userNamePswd value 23
varias hebras 16
passwordaccess prompt 18
passworddir, opción
en dsm.sys 22
PERFMONCOMMTIMEOUT 42
PERFMONTCPPORT 41
PERFMONTCPSERVERADDRESS 41
pila de hebras HP 16
política
política de retención 33
políticas para almacenar datos 29
proceso de inicio de sesión 18
protección de datos 33
protección de retención 32
proxynode 85
publicaciones v

Q

qryRespArchiveData 32
qryRespBackupData
función dsmDeleteObj 105

R

rcApiOut
ejemplo, datos 18
realizar copias de seguridad a objetos 43
recibir los datos de un servidor
descripción general 68
procedimiento 69
recuperación o restauración de objeto parcial 69
recomendaciones
configurar la pila de hebra HP 16
dsmGetObject
grandes cantidades de datos 118
recomendaciones de diseño 11
recuperación o restauración de objeto parcial 69
recuperación tras error
información de estado 59
visión general 59
recuperar 83
objetos de un servidor 68
registrar espacios de archivo 26
registrar eventos 78
regla de autorización
función dsmDeleteAccess 104
replicación de nodo 59
restaurar 83
objetos de un servidor 68
restricciones
cifrado y compresión que utiliza eliminación de copia de almacenamiento intermedio 48
varias hebras 16
retención de datos 33
ruta rápida 34

S

seguridad 18
seleccionar objetos
para restaurar 70
señales, utilizar 16
servername 2

servidor
  eliminar objetos de 77
sesión
  contraseña
    sesión 18
  iniciar con dsmInitEx 17
  seguridad 18
sin LAN
  función dsmSetUp 11
  transferencia de datos 38
Soporte de cifrado AES de 128 bits 49
Soporte de cifrado AES de 256 bits 49
soporte de proxy de nodo de cliente 85
supervisor de rendimiento
  cliente 40
supervisor de rendimiento de cliente 40

T

TCA
  control de versiones 14
  seguridad de sesión 18
  señales 17
  sin passwordaccess 21
TCPport 19
tcpserveraddr 2
teclado 217
tipo de aplicación 17, 122, 125
tipo de compresión
  LZ4 45
  LZW 45
tipos de objeto 25
transferencia de datos
  sin LAN 38
Trusted Communication Agent
  passwordaccess 21
  seguridad de sesión 18
  señales 17
tsmapifp.h 87
tsmapitd.h 87

U

Unicode
  configuración 87
  espacios de archivo que no son
    Unicode 88
  mbcs 87
  Windows 87
UNIX o Linux
  aplicación de API de ejemplo 5
usuario
  intervención 16
usuario administrativo
  creación 22
usuario autorizado 21, 25

V

valores de objectID 11
variables de entorno
  DSMI_CONFIG 3
  DSMI_DIR 3
  DSMI_LOG 4
  por sistema operativo 3
varias hebras
  indicador 11
  opción de varias hebras 16
  restricciones 16
  valor mtflag 16
  visión general 16
versión activa
  supresión 77
versión de la aplicación vii
versiones
  archivos retenidos 29

W

Windows 64 bits
  aplicación de ejemplo 7

Número de Programa: 5725-W98
5725-W99
5725-X15

Impreso en España