IBM Spectrum Protect
Version 8.1.0

Verwendung der Anwendungsprogrammierschnittstelle

IBM Spectrum Protect
Version 8.1.0

Verwendung der Anwendungsprogrammierschnittstelle

Hinweis

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die Informationen unter „Bemerkungen" auf Seite 227 gelesen werden.

Diese Ausgabe bezieht sich auf Version 8, Release 1, Modifikation 0 von IBM Spectrum Protect (Produktnummern 5725-W98, 5725-W99 und 5725-X15) und auf alle nachfolgenden Releases und Modifikationen, sofern in neuen Ausgaben nicht anders angegeben.

Diese Veröffentlichung ist eine Übersetzung des Handbuchs
IBM Spectrum Protect Version 8.1.0, Using the Application Programming Interface,
herausgegeben von International Business Machines Corporation, USA

Herausgegeben von:
TSC Germany
Kst. 2877
Dezember 2016

Inhaltsverzeichnis

Zu dieser Veröffentlichung v
Zielgruppe v
Veröffentlichungen v
Konventionen in dieser Veröffentlichung v

Neuerungen in Version 8.1.0 vii

Kapitel 1. Übersicht über die API 1
Erläuterungen zu Konfigurations- und Optionsdateien 1
API-Umgebung konfigurieren 4

Kapitel 2. API-Musteranwendung erstellen und ausführen 5
Quellendateien für die UNIX- oder Linux-Musteranwendung 5
UNIX- oder Linux-Musteranwendung erstellen . . 6
64-Bit-Musteranwendung von Windows 7

Kapitel 3. Hinweise zum Entwerfen einer Anwendung 9
Größenbeschränkungen bestimmen 12
API-Versionssteuerung verwalten 12
Multithreading verwenden 14
Signale und Signalhandler 15
Sitzung starten oder beenden 15
Sitzungssicherheit 17
Option `passwordaccess` ohne TCA auf `generate` setzen 20
Benutzer mit Verwaltungsaufgaben mit Clienteignerberechtigung erstellen 21
Objektnamen und -IDs 22
Dateibereichsname 22
Übergeordnete und untergeordnete Namen. . . 23
Objekttyp 23
Zugriff auf Objekte als Sitzungseigner 24
Knoten- und eignerübergreifender Zugriff auf Objekte 25
Dateibereiche verwalten 25
Objekte Verwaltungsklassen zuordnen 28
Verwaltungsklassen abfragen 30
Verfall/Löschen anhalten und freigeben 30
Aufbewahrungsschutz für Archivierungsdaten. . 31
IBM Spectrum Protect-System abfragen 33
Beispiel für das Abfragen des Systems 35
Servereffizienz 36
Daten an einen Server senden 37
Das Transaktionsmodell 37
Dateizusammenfassung 38
LAN-unabhängige Datenübertragung. 38
Gleichzeitiges Schreiben 39
API-Leistung verbessern 39
API für das Senden von Leistungsdaten an die Clientleistungsüberwachung konfigurieren 40
Optionen für die Clientleistungsüberwachung konfigurieren 40
Objekte an den Server senden 42
Erläuterungen zu Sicherungs- und Archivierungsobjekten 43
Komprimierung 44
Pufferkopieneliminierung. 46
API-Verschlüsselung 49
Datendeduplizierung 52
Clientseitige Datendeduplizierung der API . . . 55
Serverseitige Datendeduplizierung. 59
Anwendungsübernahme 59
Übernahmestatusinformationen. 60
Beispielablaufdiagramme für Sicherung und Archivierung. 62
Codebeispiel für API-Funktionen, mit denen Daten zum IBM Spectrum Protect-Speicher gesendet werden 65
Dateigruppierung 67
Daten von einem Server empfangen 70
Zurückschreibung oder Abruf von Teilobjekten 70
Daten zurückschreiben oder abrufen 71
Beispielablaufdiagramme für Zurückschreibung und Abruf 75
Codebeispiel für das Empfangen von Daten von einem Server 77
Objekte auf dem Server aktualisieren und löschen 78
Objekte vom Server löschen 79
Ereignisse protokollieren 79
Zustandsdiagrammzusammenfassung für die IBM Spectrum Protect-API 80

Kapitel 4. Erläuterungen zur Interoperabilität 83
Interoperabilität des Clients für Sichern/Archivieren 83
Benennung von API-Objekten 83
Für die API verwendbare Befehle des Clients für Sichern/Archivieren 85
Interoperabilität zwischen Betriebssystemen . . . 86
Mehrere Knoten mit Clientknoten-Proxy-Unterstützung sichern 87

Kapitel 5. Verwendung der API mit Unicode. 89
Situationen für die Verwendung von Unicode . . . 89
Unicode konfigurieren. 89

Kapitel 6. API-Funktionsaufrufe 93
dsmBeginGetData 96
dsmBeginQuery 97
dsmBeginTxn 102
dsmBindMC. 103
dsmChangePW. 105
dsmCleanUp 106
dsmDeleteAccess 106

dsmDeleteFS 107
dsmDeleteObj 108
dsmEndGetData 109
dsmEndGetDataEx 110
dsmEndGetObj 110
dsmEndQuery 111
dsmEndSendObj 111
dsmEndSendObjEx 112
dsmEndTxn 113
dsmEndTxnEx 114
dsmGetData 116
dsmGetBufferData 117
dsmGetNextQObj 118
dsmGetObj 121
dsmGroupHandler 122
dsmInit 123
dsmInitEx 127
dsmLogEvent 131
dsmLogEventEx 132
dsmQueryAccess 133
dsmQueryApiVersion. 134
dsmQueryApiVersionEx 135
dsmQueryCliOptions 135
dsmQuerySessInfo. 136
dsmQuerySessOptions 137
dsmRCMsg 138
dsmRegisterFS 139
dsmReleaseBuffer 140
dsmRenameObj. 141
dsmRequestBuffer 142
dsmRetentionEvent 143
dsmSendBufferData 144
dsmSendData 145
dsmSendObj. 146
dsmSetAccess 150
dsmSetUp 151
dsmTerminate 153
dsmUpdateFS 153
dsmUpdateObj 154
dsmUpdateObjEx 156

Anhang A. Quellendatei mit API-Rückkehrcodes: `dsmrc.h` 159

Anhang B. Quellendateien für API-Typdefinitionen 171

Anhang C. Quellendatei mit den API-Funktionsdefinitionen 215

Anhang D. Funktionen zur behindertengerechten Bedienung für die IBM Spectrum Protect-Produktfamilie . . . 225

Bemerkungen 227

Glossar 231

Index 233

Zu dieser Veröffentlichung

Diese Veröffentlichung stellt Informationen zur Verfügung, die Ihnen bei der Ausführung folgender Tasks helfen sollen:

- Aufrufe für die IBM Spectrum Protect-Anwendungsprogrammierschnittstelle vorhandenen Anwendungen hinzufügen
- Programme mit allgemeinen Programmierschnittstellen erstellen, die die Services von IBM Spectrum Protect anfordern

Zusätzlich zur Anwendungsprogrammierschnittstelle (API) sind die folgenden Programme für verschiedene Betriebssysteme enthalten:

- Ein Clientprogramm für Sichern/Archivieren sichert und archiviert Dateien von Workstations oder Dateiservern im Speicher und schreibt Sicherungsversionen und Archivierungskopien der Dateien in die lokalen Dateisysteme zurück bzw. ruft diese dorthin ab.
- Ein Web-Client für Sichern/Archivieren, mit dem ein berechtigter Administrator, Unterstützungsmitarbeiter oder Endbenutzer Sicherungs-, Zurückschreibungs-, Archivierungs- und Abrufservices mit Hilfe eines Web-Browsers auf einem fernen System ausführen kann.
- Ein Verwaltungsclientprogramm, auf das von einem Web-Browser oder von der Befehlszeile aus zugegriffen werden kann. Ein Administrator steuert und überwacht Serveraktivitäten, definiert Speicherverwaltungsmaßnahmen für Sichern, Archivieren und Speicherverwaltungsservices und erstellt Zeitpläne, um diese Services in regelmäßigen Abständen auszuführen.

Zielgruppe

Diese Veröffentlichung enthält Anweisungen, mit denen Sie API-Aufrufe einer vorhandenen Anwendung hinzufügen können. Sie sollten mit der Programmiersprache C und den IBM Spectrum Protect-Funktionen vertraut sein.

Veröffentlichungen

Die IBM Spectrum Protect-Produktfamilie umfasst IBM Spectrum Protect Snapshot, IBM Spectrum Protect for Space Management, IBM Spectrum Protect for Databases und verschiedene andere Speicherverwaltungsprodukte von IBM®.

Die IBM Produktdokumentation finden Sie unter IBM Knowledge Center.

Konventionen in dieser Veröffentlichung

In dieser Veröffentlichung werden die folgenden Konventionen für die Schreibweise verwendet:

Beispiel	Beschreibung
autoexec.ncf hsmgui.exe	Eine Reihe von Kleinbuchstaben mit einer Erweiterung gibt Namen von Programmdateien an.
DSMI_DIR	Eine Reihe von Großbuchstaben gibt Rückkehrcodes und andere Werte an.

Beispiel	Beschreibung
dsmQuerySessInfo	Mit Fettschrift ist Folgendes gekennzeichnet: ein Befehl, den Sie in einer Befehlszeile eingeben, der Name eines Funktionsaufrufs, der Name einer Struktur, ein Feld innerhalb einer Struktur oder ein Parameter.
timeformat	Mit kursiver Fettschrift ist eine Option des Clients für Sichern/Archivieren gekennzeichnet. Bei der Einführung der Option oder in einem Beispiel wird Fettschrift verwendet.
dateformat	Kursivschrift wird für eine Option, den Wert einer Option, einen neuen Begriff, einen Platzhalter für von Ihnen anzugebende Informationen oder zur Hervorhebung im Text verwendet.
`maxcmdretries`	Monospaceschrift wird für Fragmente eines Programms oder am Bildschirm angezeigte Informationen, wie ein Befehlsbeispiel, verwendet.
Pluszeichen (+)	Ein Pluszeichen zwischen zwei Tasten gibt an, dass beide Tasten gleichzeitig gedrückt werden.

Neuerungen in Version 8.1.0

In IBM Spectrum Protect Version 8.1.0 werden neue Funktionen und Aktualisierungen eingeführt.

Eine Liste der neuen Funktionen und der Aktualisierungen in diesem Release finden Sie in API-Aktualisierungen.

Kapitel 1. Übersicht über die API

Die Anwendungsprogrammierschnittstelle (API) von IBM Spectrum Protect ermöglicht einem Anwendungsclient die Verwendung von Speicherverwaltungsfunktionen.

Die API umfasst Funktionsaufrufe, die Sie in einer Anwendung zur Ausführung der folgenden Operationen verwenden können:
- Sitzung starten oder beenden
- Objekten Verwaltungsklassen zuordnen, bevor sie auf einem Server gespeichert werden
- Objekte auf einem Server sichern oder archivieren
- Objekte von einem Server zurückschreiben oder abrufen
- Server nach Informationen zu gespeicherten Objekten abfragen
- Dateibereiche verwalten
- Aufbewahrungsereignisse senden

Wenn Sie als Anwendungsentwickler die API installieren, erhalten Sie die Dateien, die ein Endbenutzer einer Anwendung benötigt:
- die gemeinsam genutzte Bibliothek der API
- die Nachrichtendatei
- die Musterclientoptionsdateien
- den Quellcode für die API-Headerdateien, die Ihre Anwendung benötigt
- den Quellcode für eine Musteranwendung und die Makefile zum Erstellen dieser Anwendung
- die Datei dsmtca (nur UNIX und Linux)

Bei 64-Bit-Anwendungen sollten alle Kompilierungen unter Verwendung von Compileroptionen ausgeführt werden, die die 64-Bit-Unterstützung ermöglichen. Beispielsweise sollte '-q64' verwendet werden, wenn API-Anwendungen unter AIX erstellt werden, und '-m64' sollte unter Linux verwendet werden. Die Mustermakefiles enthalten weitere Informationen.

Wichtig: Wenn Sie die API installieren, stellen Sie sicher, dass alle Dateien dieselbe Version haben.

Informationen zur Installation der API finden Sie in Installation der IBM Spectrum Protect-Clients für Sichern/Archivieren.

Verweise auf UNIX und Linux schließen die Betriebssysteme AIX, HP-UX, Linux, Mac OS X und Oracle Solaris ein.

Erläuterungen zu Konfigurations- und Optionsdateien

Konfigurations- und Optionsdateien definieren die Bedingungen und Einschränkungen, unter denen die Sitzung ausgeführt wird.

Als Administrator oder Endbenutzer können Sie Optionswerte für Folgendes definieren:
- Konfiguration der Verbindung zu einem Server
- Steuerung, welche Objekte an den Server gesendet werden, und Festlegung der Verwaltungsklasse, der sie zugeordnet sind

Sie definieren Optionen in einer oder zwei Dateien, wenn Sie die API auf Ihrer Workstation installieren.

Unter den Betriebssystemen UNIX und Linux sind die Optionen in zwei Optionsdateien gespeichert:
- dsm.opt - die Clientoptionsdatei
- dsm.sys - die Clientsystemoptionsdatei

Unter anderen Betriebssystemen enthält die Clientoptionsdatei (dsm.opt) alle Optionen.

Einschränkung: Die API unterstützt nicht die folgenden Optionen des Clients für Sichern/Archivieren:
- `autofsrename`
- `changingretries`
- `domain`
- `eventlogging`
- `groups`
- `subdir`
- `users`
- `virtualmountpoint`

Sie können auch Optionen im Funktionsaufruf **`dsmInitEx`** angeben. Verwenden Sie den Parameter für die Optionszeichenfolge oder den Parameter für die API-Konfigurationsdatei.

Dieselbe Option kann aus mehreren Konfigurationsquellen stammen. In diesem Fall hat die Quelle mit der höchsten Priorität Vorrang. In Tabelle 1 ist die Prioritätsreihenfolge aufgelistet.

Tabelle 1. Konfigurationsquellen in absteigender Prioritätsreihenfolge

Priorität	UNIX und Linux	Windows	Beschreibung
1	dsm.sys (Datei) (Clientsystemoptionen)	nicht zutreffend	Diese Datei enthält Optionen, die ein Systemadministrator ausschließlich für UNIX und Linux definiert. **Tipp:** Wenn Ihre Datei dsm.sys Serverzeilengruppen enthält, stellen Sie sicher, dass in allen Zeilengruppen für die Option `passwordaccess` derselbe Wert angegeben ist (entweder `prompt` oder `generate`).

Tabelle 1. Konfigurationsquellen in absteigender Prioritätsreihenfolge (Forts.)

Priorität	UNIX und Linux	Windows	Beschreibung
2	Optionszeichenfolge (Clientoptionen)	Optionszeichenfolge (alle Optionen)	Eine dieser Optionen wird wirksam, wenn sie als Parameter an einen Aufruf **dsmInitEx** übergeben wird. Die Liste kann Clientoptionen wie `compressalways`, `servername` (nur UNIX und Linux) oder `tcpserveraddr` (nicht UNIX) enthalten. Mithilfe der API-Optionszeichenfolge kann ein Anwendungsclient Änderungen an den Optionswerten in der API-Konfigurationsdatei und der Clientoptionsdatei durchführen. Beispielsweise kann Ihre Anwendung den Endbenutzer abfragen, ob Komprimierung erforderlich ist. Abhängig von den Benutzeraktionen können Sie eine API-Optionszeichenfolge mit dieser Option erstellen und in den Aufruf **dsmInitEx** übergeben. Informationen zum Format der API-Optionszeichenfolge finden Sie in „**dsmInitEx**" auf Seite 127. Sie können diesen Parameter auch auf `NULL` setzen. Dies gibt an, dass für diese Sitzung keine API-Optionszeichenfolge vorhanden ist.
3	API-Konfigurationsdatei (Clientoptionen)	API-Konfigurationsdatei (alle Optionen)	Die Werte, die Sie in der API-Konfigurationsdatei definieren, überschreiben die Werte, die Sie in der Clientoptionsdatei definieren. Definieren Sie die Optionen in der API-Konfigurationsdatei mit geeigneten Werten für die IBM Spectrum Protect-Sitzung des Benutzers. Die Werte werden wirksam, wenn der Name der API-Konfigurationsdatei als Parameter in dem Aufruf **dsmInitEx** übergeben wird. Sie können diesen Parameter auch auf NULL setzen. Dies gibt an, das für diese Sitzung keine API-Konfigurationsdatei vorhanden ist.
4	Datei dsm.opt (Clientoptionen)	Datei dsm.opt (alle Optionen)	Unter den Betriebssystemen UNIX und Linux enthält die Datei dsm.opt nur die Benutzeroptionen. Unter anderen Betriebssystemen enthält die Datei dsm.opt alle Optionen. Um die Optionen in diesen Dateien zu überschreiben, führen Sie die in dieser Tabelle beschriebenen Verfahren aus.

Zugehörige Konzepte:

Verarbeitungsoptionen

API-Umgebung konfigurieren

Die API verwendet eindeutige Umgebungsvariablen, um Dateien zu lokalisieren. Sie können für API-Anwendungen andere Dateien verwenden als die, die vom Client für Sichern/Archivieren verwendet werden. Anwendungen können mithilfe des Funktionsaufrufs **dsmSetup** die Werte überschreiben, die in den Umgebungsvariablen definiert sind.

Tipp: Unter Windows ist das Standardinstallationsverzeichnis `%SystemDrive%\Programme\Common Files\Tivoli\TSM\api`.

In Tabelle 2 sind die API-Umgebungsvariablen nach Betriebssystem aufgelistet.

Tabelle 2. API-Umgebungsvariablen

Variablen	UNIX und Linux	Windows
DSMI_CONFIG	Der vollständig qualifizierte Name für die Clientoptionsdatei (dsm.opt).	Der vollständig qualifizierte Name für die Clientoptionsdatei (dsm.opt).
DSMI_DIR	Verweist auf den Pfad, der dsm.sys, dsmtca, das Unterverzeichnis en_US und alle anderen Sprachen für die Unterstützung in der Landessprache enthält. Das Unterverzeichnis en_US muss dsmclientV3.cat enthalten.	Verweist auf den Pfad, der dscenu.txt und alle Nachrichtendateien für die Unterstützung in der Landessprache enthält.
DSMI_LOG	Verweist auf den Pfad für die Datei dsierror.log.	Verweist auf den Pfad für die Datei dsierror.log. Wenn die Clientoption **errorlogname** (Fehlerprotokollname) definiert ist, überschreibt die durch diese Option angegebene Position das mit DSMI_LOG angegebene Verzeichnis.

Kapitel 2. API-Musteranwendung erstellen und ausführen

Das API-Paket enthält Musteranwendungen, die die API-Funktionsaufrufe im Kontext veranschaulichen. Installieren Sie eine Musteranwendung und schauen Sie sich den Quellcode an, um sich mit der Verwendung der Funktionsaufrufe vertraut zu machen.

Wählen Sie eines der folgenden API-Musteranwendungspakete aus:
- Paket für interaktive Einzelthread-Anwendung (dapi*)
- Paket für Multithread-Anwendung (callmt*)
- Testanwendung für logische Objektgruppierung (dsmgrp*)
- Musteranwendung für ereignisgesteuerte Aufbewahrungsdauer (callevnt)
- Musteranwendung für Status 'Löschen unzulässig' (callhold)
- Musteranwendung für Aufbewahrungsschutz für Daten (callret)
- Musterprogramm für IBM Spectrum Protect-Datenpuffer (callbuff)

Schauen Sie sich als Einstieg die Prozedur zum Erstellen der Musteranwendung dapismp für Ihre Plattform an:
- Für UNIX- und Linux-Anwendungen siehe „Quellendateien für die UNIX- oder Linux-Musteranwendung".
- Für Windows-Anwendungen siehe „64-Bit-Musteranwendung von Windows" auf Seite 7.

Die Musteranwendung dapismp erstellt eigene Datenströme, wenn Objekte gesichert oder archiviert werden. Es werden keine Objekte aus dem Dateisystem auf der lokalen Festplatte gelesen oder in dieses Dateisystem geschrieben. Der Objektname entspricht keiner Datei auf Ihrer Workstation. Mit der „Seedzeichenfolge", die Sie ausgeben, wird ein Muster generiert, das verifiziert werden kann, wenn das Objekt zurückgeschrieben oder abgerufen wird. Befolgen Sie, nachdem Sie die Musteranwendung kompiliert und **dapismp** zum Starten der Musteranwendung ausgeführt haben, die Anweisungen, die am Bildschirm angezeigt werden.

Quellendateien für die UNIX- oder Linux-Musteranwendung

Um die UNIX- oder Linux-Musteranwendung zu erstellen und auszuführen, müssen Sie sicherstellen, dass bestimmte Quellendateien vorhanden sind. Sobald Sie die Musteranwendung erstellt haben, können Sie die Anwendung kompilieren und ausführen.

Die Dateien, die in Tabelle 3 aufgelistet sind, umfassen die Quellendateien und anderen Dateien, die Sie zum Erstellen der Musteranwendung benötigen, die Teil des API-Pakets ist.

Tabelle 3. Erforderliche Dateien für das Erstellen der UNIX- oder Linux-API-Musteranwendung

Dateinamen	Beschreibung
README_api_enu	Readme-Datei
dsmrc.h	Headerdatei für Rückkehrcodes
dsmapitd.h	Headerdatei für allgemeine Typdefinitionen
dsmapips.h	Headerdatei für betriebssystemspezifische Typdefinitionen
dsmapifp.h	Headerdatei für Funktionsprototypen
release.h	Headerdatei für Releasewerte

Tabelle 3. Erforderliche Dateien für das Erstellen der UNIX- oder Linux-API-Musteranwendung (Forts.)

Dateinamen		Beschreibung
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	Module für die befehlszeilengesteuerte Musteranwendung
makesmp[64].*xxx*		Makefile zum Erstellen von dapismp für Ihr Betriebssystem. *xxx* gibt das Betriebssystem an.
callmt1.c callmt2.c		Multithread-Musterdateien
callmtu1.c callmtu2.c		Multithread-Unicode-Musterdateien
libApiDS.xx libApiDS64.xx oder libApiTSM64.xx		Gemeinsam genutzte Bibliothek (das Suffix ist plattformabhängig)
dsmgrp.c callevnt.c callhold.c callret.c callbuff.c dpsthread.c		Musterdateien für Gruppierung Musterquellcode für Maßnahme für ereignisgesteuerte Aufbewahrungsdauer Musterquellcode für Status 'Löschen unzulässig' Musterquellcode für Aufbewahrungsschutz für Daten

UNIX- oder Linux-Musteranwendung erstellen

Sie erstellen die API-Musteranwendung **dapismp** unter Verwendung eines Compilers für Ihr Betriebssystem.

Zum Erstellen der API-Musteranwendung für UNIX oder Linux müssen Sie die folgenden Compiler installieren:

- IBM AIX - IBM Visual Age Compiler Version 6 oder höher
- HP-IA64 - aCC-Compiler A.05.50 oder höher
- Linux - GCC-Compiler Version 3.3.3 oder höher
- Mac OS X - GCC-Compiler Version 4.0 oder höher
- Oracle Solaris - Oracle Studio C++-Compiler Version 11 oder höher

1. Führen Sie den folgenden Befehl aus, um die API-Muster zu erstellen:

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

   xxx gibt das Betriebssystem an.
2. Nachdem Sie die Muster erstellt haben, definieren Sie die Umgebungsvariablen, einschließlich `DSMI_DIR`, und die Optionsdateien. Weitere Informationen finden Sie in „Erläuterungen zu Konfigurations- und Optionsdateien" auf Seite 1.
3. Melden Sie sich beim erstmaligen Anmelden als Rootbenutzer an, um Ihr Kennwort zu registrieren.

   Tipp: Wird die Option `compressalways` auf `no` gesetzt, wird ein Objekt möglicherweise nicht dekomprimiert zurückgesendet. Dieses Verhalten ist von der Anwendungsfunktionalität abhängig.

Zum Angeben der Shared-Memory-Übertragungsmethode unter AIX muss der IBM Spectrum Protect-API-Clientbenutzer eine der folgenden Bedingungen erfüllen:

- Er muss als Rootbenutzer angemeldet sein.
- Er muss dieselbe Benutzer-ID (UID) haben wie der Prozess, der den IBM Spectrum Protect-Server ausführt.

Diese Einschränkung gilt nicht, wenn die Option `passwordaccess` in der Clientsystemoptionsdatei `dsm.sys` auf `generate` gesetzt ist und TCA verwendet wird oder wenn Sie die Dateiberechtigungen für Ihr Anwendungsprogramm mithilfe der folgenden Befehle ändern:

```
chown root.system Ihr_API-Programm
chown u+s Ihr-API-Programm
```

Weitere Informationen enthält die Dokumentation zum Anwendungsprogramm.

4. Führen Sie den Befehl **dapismp** aus, um die Anwendung zu starten.
5. Wählen Sie aus der Liste der angezeigten Optionen aus. Führen Sie die Anmeldeaktion aus, bevor Sie andere Aktionen ausführen.

   Voraussetzung: Geben Sie vor dem Dateibereichsnamen, dem übergeordneten und dem untergeordneten Namen bei der Eingabe immer den korrekten Pfadbegrenzer (/) als Präfix an, beispielsweise `/eigenerDateibereich`. Sie müssen dieses Präfix auch verwenden, wenn Sie den Stern (*) als Platzhalterzeichen angeben.

Zugehörige Konzepte:

Umgebungsvariablen (UNIX- und Linux-Systeme)

64-Bit-Musteranwendung von Windows

Um die Musteranwendung für Microsoft Windows-64-Bit-Systeme zu erstellen und auszuführen, müssen Sie die IBM Spectrum Protect-API installieren und sicherstellen, dass bestimmte Quellendateien vorhanden sind.

Einschränkungen:

- Die besten Ergebnisse werden bei Verwendung von dynamischem Laden erzielt. Nehmen Sie als Beispiel die Datei `dynaload.c` und die Implementierung im Mustercode.
- Dateien für die Musteranwendung befinden sich in den folgenden Verzeichnissen:

  api64\obj
  Enthält die Objektdateien des API-Musterprogramms.

  api64\samprun
  Enthält das Musterprogramm **dapismp**. Das Musterprogramm enthält das Ausführungsverzeichnis.
- Die DLL-Datei tsmapi64.dll ist eine 64-Bit-DLL-Datei.
- Verwenden Sie Microsoft C/C++ Compiler Version 15 und die Makefile `makesmp64.mak`, um die API-Musteranwendung **dapismp** zu kompilieren. Möglicherweise müssen Sie die Makefiles an Ihre Umgebung anpassen, insbesondere die Bibliothek oder die Einschlussverzeichnisse.
- Führen Sie nach der Kompilierung der Anwendung die Musteranwendung durch Ausgabe des Befehls **dapismp** im Verzeichnis `api64\samprun` aus.

- Wählen Sie aus der Liste der angezeigten Optionen aus. Führen Sie die Anmeldeaktion aus, bevor Sie andere Aktionen ausführen.
- Geben Sie vor dem Dateibereichsnamen, dem übergeordneten und dem untergeordneten Namen bei der Eingabe immer den korrekten Pfadbegrenzer (\) als Präfix an, z. B. `\eigenerDateibereich`. Sie müssen dieses Präfix auch verwenden, wenn Sie den Stern (*) als Platzhalterzeichen angeben.

Für Windows-Betriebssysteme sind die für das Erstellen der Musteranwendung erforderlichen Quellendateien in Tabelle 4 aufgelistet. Die Musteranwendung ist im API-Paket enthalten. Außerdem befindet sich dort eine vorkompilierte ausführbare Datei (dapismp.exe).

Tabelle 4. Dateien für das Erstellen der Windows-64-Bit-API-Musteranwendung

Dateinamen	Beschreibung
`api.txt`	Readme-Datei
`tsmapi64.dll`	API-DLLs

`dsmrc.h`
`dsmapitd.h`
`dsmapips.h`
`dsmapifp.h`
`dsmapidl.h`
`release.h`
	Headerdatei für Rückkehrcodes
	Headerdatei für allgemeine Typdefinitionen
	Headerdatei für betriebssystemspezifische Typdefinitionen
	Headerdatei für Funktionsprototypen
	Dynamisch geladene Headerdatei für Funktionsprototypen
	Headerdatei für Releasewerte

`dapidata.h`
`dapint64.h`
`dapitype.h`
`dapiutil.h`
	Headerdateien für Quellcode

`tsmapi64.lib`	Implizite Bibliothek

`dapibkup.c`
`dapiinit.c`
`dapint64.c`
`dapipref.c`
`dapiproc.c`
`dapiproc.h`
`dapipw.c`
`dapiqry.c`
`dapirc.c`
`dapismp64.c`
`dapiutil.c`
`dynaload.c`
	Quellcodedateien für dapismp.exe

`makesmpx64.mak` (Windows x64) `makesmp64.mak` (Windows IA64)	Makefile zum Erstellen von Musteranwendungen

`callmt1.c`
`callmt2.c`
`callmtu164.c`
`callmtu264.c`
 	
`dpsthread.c`
	Multithread-Musterdateien
	
	Musterdatei mit Quellcode

`callevnt.c`
`callhold.c`
`callret.c`
`callbuff.c`
	Quellcode für Maßnahme für ereignisgesteuerte Aufbewahrungsdauer
	Musterquellcode für Status 'Löschen unzulässig'
	Musterquellcode für Aufbewahrungsschutz für Daten
	Musterquellcode für gemeinsam genutzten Puffer (keine Kopie).

Kapitel 3. Hinweise zum Entwerfen einer Anwendung

Wenn Sie eine Anwendung entwerfen, müssen Sie über Kenntnisse über viele Aspekte der API verfügen.

Lesen Sie die folgenden Abschnitte, um ein Verständnis für die API zu entwickeln:

- „Größenbeschränkungen bestimmen" auf Seite 12
- „API-Versionssteuerung verwalten" auf Seite 12
- „Multithreading verwenden" auf Seite 14
- „Signale und Signalhandler" auf Seite 15
- „Sitzung starten oder beenden" auf Seite 15
- „Objektnamen und -IDs" auf Seite 22
- „Option `passwordaccess` ohne TCA auf `generate` setzen" auf Seite 20
- „Zugriff auf Objekte als Sitzungseigner" auf Seite 24
- „Knoten- und eignerübergreifender Zugriff auf Objekte" auf Seite 25
- „Dateibereiche verwalten" auf Seite 25
- „Objekte Verwaltungsklassen zuordnen" auf Seite 28
- „Verfall/Löschen anhalten und freigeben" auf Seite 30
- „IBM Spectrum Protect-System abfragen" auf Seite 33
- „Daten an einen Server senden" auf Seite 37
- „Beispielablaufdiagramme für Sicherung und Archivierung" auf Seite 62
- „Dateigruppierung" auf Seite 67
- „Zustandsdiagrammzusammenfassung für die IBM Spectrum Protect-API" auf Seite 80

Wenn Sie Ihre Anwendung entwerfen, lesen Sie die Hinweise in Tabelle 5. Startstrukturen mit **memset**-Feldern werden möglicherweise in nachfolgenden Releases geändert. Der Wert für `stVersion` erhöht sich mit jeder Produkterweiterung.

Tabelle 5. API: Hinweise zum Entwerfen von Anwendungen

Entwurfselement	Hinweise
Ländereinstellung definieren	Die Anwendung muss die Ländereinstellung festlegen, bevor die API aufgerufen wird. Wenn Sie den Standardwert für die Ländereinstellung verwenden möchten, fügen Sie der Anwendung den folgenden Code hinzu: `setlocale(LC_ALL,"");` Soll ein anderer Wert für die Ländereinstellung definiert werden, verwenden Sie denselben Aufruf mit der entsprechenden Ländereinstellung im zweiten Parameter. Überprüfen Sie die Spezifikationen in der Dokumentation für jedes von Ihnen verwendete Betriebssystem.

Tabelle 5. API: Hinweise zum Entwerfen von Anwendungen (Forts.)

Entwurfselement	Hinweise

Sitzungs-steuerung	Wenden Sie die folgenden Richtlinien für die Sitzungssteuerung an:
	• Weisen Sie jedem von Ihnen verwendeten Produkt des IBM Spectrum Protect-Clients für Sichern/Archivieren und des IBM Spectrum Protect-API-Clients einen eindeutigen Knotennamen zu. Die folgenden Produkte sind Beispiele für diese Clients:
	– IBM Spectrum Protect for Mail
	– oder IBM Spectrum Protect HSM for Windows
	• Verwenden Sie während einer Sicherungs- und Zurückschreibungsprozedur einen konsistenten Eignernamen.
	• Verwenden Sie zum Steuern des Zugriffs auf die geschützte Kennwortdatei die Option `passwordaccess`. Nur unter UNIX und Linux wirkt sich diese Option auf den untergeordneten TCA-Prozess bei Knotennamen, Sitzungseignernamen und Kennwortmanagement aus.
	• Stellen Sie sicher, dass Datenversetzungssitzungen enden, wenn die Task abgeschlossen ist, sodass Einheiten auf dem Server für die Verwendung durch andere Sitzungen freigegeben werden.
	• Verwenden Sie den Funktionsaufruf **dsmSetup** mit aktiviertem Flag `multithread`, um LAN-unabhängige Datenübertragung zu gestatten.
	• Wenn Sie unter AIX Multithread-Anwendungen oder LAN-unabhängige Operationen verwenden, insbesondere auf Multiprozessormaschinen, setzen Sie in der Umgebung die Umgebungsvariable `AIXTHREAD_SCOPE` vor dem Starten der Anwendung auf `S`, um Leistung und Zeitplanung zu verbessern. Beispiel:
	`EXPORT AIXTHREAD_SCOPE=S`
	Wenn `AIXTHREAD_SCOPE` auf `S` gesetzt ist, werden mit Standardattributen erstellte Benutzerthreads in den systemweiten Zuteilungsbereich gestellt. Wenn ein Benutzerthread mit systemweitem Zuteilungsbereich erstellt wird, wird er an einen Kernel-Thread gebunden und vom Kernel geplant. Der zugrunde liegende Kernel-Thread wird mit keinem anderen Benutzerthread gemeinsam genutzt. Weitere Informationen zu dieser Umgebungsvariablen finden Sie hier:
	„Multithreading verwenden" auf Seite 14
	• Stellen Sie sicher, dass eine API-Funktion nicht gleichzeitig von mehreren Threads in einer Sitzung aufgerufen wird. Anwendungen, die mehrere Threads mit derselben Sitzungskennung verwenden, müssen die API-Aufrufe synchronisieren. Verwenden Sie beispielsweise einen **mutex**, um API-Aufrufe zu synchronisieren:
	getTSMMutex()
	TSM-API-Aufruf ausgeben
	releaseTSMMutex()
	Verwenden Sie diese Methode nur, wenn die Threads eine Kennung gemeinsam nutzen. Sie können parallele API-Funktionsaufrufe verwenden, wenn die Aufrufe unterschiedliche Sitzungskennungen verwenden.
	• Implementieren Sie für das Versetzen von Daten ein Konsumenten-/Produzentenmodell mit Threads. Die API-Aufrufe sind synchron und die Aufrufe der Funktionen **dsmGetData** und **dsmSendData** blockieren, bis sie beendet sind. Bei Verwendung eines Konsumenten-/Produzentenmodells kann die Anwendung den nächsten Puffer lesen, während sie auf das Netz wartet. Die Entkopplung von Datenlese-/-schreiboperationen und Netzoperationen verbessert außerdem die Leistung, wenn ein Netzengpass oder Verzögerungen vorliegen. Im Allgemeinen gilt:
	`Data thread <---> shared queue of buffers <---> communication thread (issue calls to the IBM Spectrum Protect API)`
	• Verwenden Sie dieselbe Sitzung für mehrere Operationen, um einen erhöhten Systemaufwand zu vermeiden. Implementieren Sie für Anwendungen, die viele kleine Objekte bearbeiten, die Anwendung von Sitzungspools, sodass dieselbe Sitzung von mehreren kleinen Operationen verwendet werden kann. Systemaufwand ist mit dem Öffnen und Schließen einer Sitzung mit dem IBM Spectrum Protect-Server verbunden. Der Aufruf `dsmInit/dsmInitEX` ist serialisiert, sodass auch in einer Multithread-Anwendung nicht mehrere Threads gleichzeitig angemeldet sein können. Außerdem sendet die API während der Anmeldung eine Reihe von einmaligen Abfragen an den Server, sodass der Server alle Operationen ausführen kann. Diese Abfragen umfassen Maßnahmen, Optionen, Dateibereiche und die lokale Konfiguration.

Tabelle 5. API: Hinweise zum Entwerfen von Anwendungen (Forts.)

Entwurfselement	Hinweise
Operationsfolge	Der IBM Spectrum Protect-Server sperrt Datenbankeinträge im Dateibereich während einiger Operationen. Für das Entwerfen von IBM Spectrum Protect-API-Anwendungen gelten folgende Regeln: • Abfragen sperren den Dateibereich während der gesamten Transaktion. • Die Abfragesperre kann mit anderen Abfrageoperationen gemeinsam genutzt werden, so dass mehrere Abfrageoperationen in demselben Dateibereich gleichzeitig ausgeführt werden können. • Die folgenden Operationen werden zum Ändern der IBM Spectrum Protect-Serverdatenbank verwendet (**DB Chg**): Senden, Abrufen, Umbenennen, Aktualisieren und Löschen. • Die Beendigung einer **DB Chg**-Operation erfordert eine Dateibereichssperre während der Datenbankänderung am Ende der Transaktion. • Es können mehrere **DB Chg**-Operationen in demselben Dateibereich gleichzeitig ausgeführt werden. Es kann eine Verzögerung beim Warten auf die Sperre am Ende der Transaktion auftreten. • Die Abfragesperre kann nicht mit **DB Chg**-Operationen gemeinsam genutzt werden. Eine **DB Chg**-Operation verzögert den Anfang einer Abfrage in demselben Dateibereich. Daher sollten Sie Ihre Anwendungen so gestalten, dass Abfragen von **DB Chg**-Operationen in demselben Dateibereich getrennt und serialisiert werden.
Objektbenennung	Beachten Sie beim Benennen von Objekten die folgenden Faktoren: • Die spezifischen Objektnamen sind die übergeordneten und untergeordneten Objektnamen. Enthält der Name eine eindeutige Kennung, z. B. eine Datumszeitmarke, sind Sicherungsobjekte immer aktiv. Die Objekte verfallen nur, wenn sie durch den Funktionsaufruf **dsmDeleteObj** als inaktiv markiert werden. • Durch die Zurückschreibungsmethode für Objekte wird die Art der Namensformatierung für einfache Abfragen festgelegt. Wenn Sie eine Zurückschreibung von Teilobjekten durchführen wollen, können Sie keine Komprimierung verwenden. Verwenden Sie die Funktion **dsmSendObj objAttr objCompressed=bTrue** zum Unterdrücken der Komprimierung.
Objektgruppierung	Gruppieren Sie Objekte logisch durch die Verwendung von Dateibereichen. Ein Dateibereich ist ein Container auf dem Server, der eine Gruppierungskategorie für die Objekte darstellt. Die API fragt während der ersten Anmeldung und auch während der Abfragen alle Dateibereiche ab, sodass die Anzahl der Dateibereiche eingeschränkt werden muss. Es ist eine realistische Annahme, dass eine Anwendung 20 bis 100 Dateibereiche pro Knoten aufbaut. Die API kann mehr Dateibereiche verarbeiten, aber jeder Dateibereich stellt einen Aufwand für das System dar. Verwenden Sie das Objekt `Verzeichnis` in der Anwendung, um eine differenziertere Trennung zu erreichen.
Objektbearbeitung	Speichern Sie keine `objectID`-Werte zur Verwendung in späteren Zurückschreibungen. Die Persistenz dieser Werte ist während der Lebensdauer des Objekts nicht garantiert. Achten Sie während einer Zurückschreibung besonders auf die Zurückschreibungsreihenfolge. Sortieren Sie nach der Abfrage anhand dieses Werts, bevor die Zurückschreibung ausgeführt wird. Wenn Sie mehrere Typen serieller Datenträger verwenden, greifen Sie in separaten Sitzungen auf die verschiedenen Datenträgertypen zu. Weitere Informationen finden Sie im folgenden Thema: „Objekte nach Zurückschreibungsreihenfolge auswählen und sortieren" auf Seite 72
Verwaltungsklasse	Überlegen Sie, wie viel Kontrolle die Anwendung über die Verwaltungsklasse haben muss, die den Anwendungsobjekten zugeordnet ist. Sie können Einschlussanweisungen definieren oder einen Namen im Funktionsaufruf **dsmSendObj** angeben.
Objektgröße	IBM Spectrum Protect benötigt eine Größenschätzung für jedes Objekt. Überlegen Sie, wie Ihre Anwendung die Größe eines Objekts schätzt. Eine zu hohe Schätzung der Objektgröße ist besser als eine zu niedrige.

Größenbeschränkungen bestimmen

Bestimmte Datenstrukturen oder Felder in der API unterliegen Größenbeschränkungen. Bei diesen Strukturen handelt es sich häufig um Namen oder andere Textfelder, die eine vordefinierte Länge nicht überschreiten dürfen.

Die folgenden Felder sind Beispiele für Datenstrukturen mit Größenbeschränkungen:
- Anwendungstyp
- Archivierungsbeschreibung
- Ziel für Kopiengruppe
- Kopiengruppenname
- Dateibereichsinformationen
- Verwaltungsklassenname
- Objekteignername
- Kennwort

Diese Grenzwerte sind als Konstanten in der Headerdatei `dsmapitd.h` definiert. Jede Speicherzuordnung basiert auf diesen Konstanten und nicht auf von Ihnen eingegebenen Werten. Weitere Informationen finden Sie in Anhang B, „Quellendateien für API-Typdefinitionen", auf Seite 171.

API-Versionssteuerung verwalten

Für alle APIs gibt es eine Form der Versionssteuerung. Die von Ihnen in Ihrer Anwendung verwendete API-Version muss mit der Version der API-Bibliothek kompatibel sein, die auf der Benutzerworkstation installiert ist.

`dsmQueryApiVersionEx` sollte der erste API-Aufruf sein, den Sie bei Verwendung der API eingeben. Dieser Aufruf führt die folgenden Tasks aus:
- bestätigt, dass die API-Bibliothek auf dem System des Endbenutzers installiert und verfügbar ist.
- gibt den Versionsstand der API-Bibliothek zurück, auf die die Anwendung zugreift.

Die API ist auf Aufwärtskompatibilität angelegt. Für ältere Versionen oder Releases der API-Bibliothek geschriebene Anwendungen funktionieren ordnungsgemäß, wenn Sie eine höhere Version ausführen.

Die Bestimmung des Releases der API-Bibliothek ist sehr wichtig, weil einige Releases einen anderen Speicherbedarf und andere Datenstrukturdefinitionen haben können. Abwärtskompatibilität ist unwahrscheinlich. Informationen zu Ihrer Plattform finden Sie in Tabelle 6.

Tabelle 6. Plattformkompatibilitätsinformationen

Plattform	Beschreibung
Windows	Die Nachrichtendateien müssen dieselbe Version wie die Bibliothek (DLL) haben. Das Trusted Communication Agent-Modul (dsmtca) wird nicht verwendet.
UNIX oder Linux	Die API-Bibliothek, das Trusted Communication Agent-Modul (dsmtca) und die Nachrichtendateien müssen dieselbe Version haben.

Der Aufruf **dsmQueryApiVersionEx** gibt die Version der API-Bibliothek zurück, die auf der Endbenutzerworkstation installiert ist. Dann können Sie den zurückgegebenen Wert mit der Version der API vergleichen, die der Anwendungsclient verwendet.

Die API-Versionsnummer des Anwendungsclients wird als Gruppe von vier in dsmapitd.h definierten Konstanten in den kompilierten Objektcode eingegeben:

```
DSM_API_VERSION
DSM_API_RELEASE
DSM_API_LEVEL
DSM_API_SUB_LEVEL
```

Siehe Anhang B, „Quellendateien für API-Typdefinitionen", auf Seite 171.

Die API-Version des Anwendungsclients sollte kleiner-gleich der Version der API-Bibliothek sein, die auf dem Benutzersystem installiert ist. Beachten Sie alle anderen Bedingungen. Sie können den Aufruf **dsmQueryApiVersionEx** jederzeit eingeben, unabhängig davon, ob die API-Sitzung gestartet wurde oder nicht.

Von der API verwendete Datenstrukturen enthalten ebenfalls Versionssteuerungsinformationen. Strukturen enthalten Versionsinformationen als erstes Feld. Wenn Erweiterungen an Strukturen vorgenommen werden, wird die Versionsnummer erhöht. Wenn Sie das Versionsfelds initialisieren, verwenden Sie den definierten Strukturversionswert in dsmapitd.h.

Abb. 1 auf Seite 14 zeigt die Typdefinition der Struktur **dsmApiVersionEx** aus der Headerdatei dsmapitd.h. In dem Beispiel wird dann eine globale Variable mit dem Namen ***apiLibVer*** definiert. Außerdem wird die Verwendung in einem Aufruf an **dsmQueryApiVersionEx** gezeigt, mit dem die Version der API-Bibliothek des Endbenutzers zurückgegeben wird. Schließlich wird der zurückgegebene Wert mit der API-Versionsnummer des Anwendungsclients verglichen.

```
typedef struct
{
        dsUint16_t     stVersion;              /* Strukturversion              */
        dsUint16_t version;       /* API-Version                       */
        dsUint16_t release;       /* API-Release                       */
        dsUint16_t level;         /* API-Stand                         */
        dsUint16_t subLevel;      /* API-Unterstufe                    */
} dsmApiVersionEx;

dsmApiVersionEx apiLibVer;

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

/* Kompatibilitätsproblemprüfung */
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("Die IBM Spectrum Protect-Bibliothek ist niedriger als die Anwendungsversion.\n");
      printf("Installieren Sie die aktuelle Bibliotheksversion.\n");
      printf("************************************************************\n");
      return 0;
   }

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

Abbildung 1. Beispiel für das Abrufen des Versionsstands der API

Multithreading verwenden

Die Multithread-API ermöglicht Anwendungen das Erstellen mehrerer Sitzungen mit dem IBM Spectrum Protect-Server innerhalb desselben Prozesses. Die API kann erneut aufgerufen werden. Alle Aufrufe können in unterschiedlichen Threads parallel ausgeführt werden.

Tipp: Wenn Sie Anwendungen ausführen, die eine Multithread-API voraussetzen, verwenden Sie den Aufruf **dsmQueryAPIVersionEx**.

Um die API im Multithread-Modus auszuführen, setzen Sie den Wert *mtflag* im Aufruf **dsmSetUp** auf DSM_MULTITHREAD. Der Aufruf **dsmSetUp** muss der erste Aufruf nach dem Aufruf **dsmQueryAPIVersionEx** sein. Dieser Aufruf muss zurückkehren, bevor ein Thread den Aufruf **dsmInitEx** aufruft. Nachdem die Ausführung aller Threads beendet ist, geben Sie einen Aufruf **dsmCleanUp** ein. Der primäre Prozess darf nicht enden, bevor nicht die Verarbeitung aller Threads abgeschlossen ist. Siehe callmt1.c in der Musteranwendung.

Einschränkung: Der Standardmodus für die API ist der Einzelthread-Modus. Wenn eine Anwendung **dsmSetUp** nicht mit der Angabe DSM_MULTITHREAD für den Wert *mtflag* aufruft, erlaubt die API für jeden Prozess nur eine einzige Sitzung.

Sobald der Aufruf **dsmSetUp** erfolgreich ausgeführt wurde, kann die Anwendung mehrere Threads starten und mehrere Aufrufe **dsmInitEx** ausführen. Jeder Aufruf **dsmInitEx** gibt eine Kennung für diese Sitzung zurück. Alle nachfolgenden Aufrufe in diesem Thread für diese Sitzung müssen diesen Wert für die Kennung verwenden. Bei bestimmten Werten handelt es sich um prozessweite Umgebungsvariablen (Werte, die im Aufruf **dsmSetUp** definiert werden). Bei jedem Aufruf **dsmInitEx** werden die Optionen erneut syntaktisch analysiert. Jeder Thread kann mit unterschied-

lichen Optionen ausgeführt werden, indem eine Überschreibungsdatei oder eine Optionszeichenfolge im Aufruf **dsmInitEx** angegeben wird. Damit können unterschiedliche Threads auf verschiedenen Servern ausgeführt werden oder verschiedene Knotennamen verwenden.

Empfehlung: Definieren Sie bei HP den Thread-Stack mit 64 KB oder höher. Der Standardwert für den Thread-Stack (32 KB) ist möglicherweise nicht hoch genug.

Um Anwendungsbenutzern die Verwendung einer LAN-unabhängigen Sitzung zu ermöglichen, geben Sie **dsmSetUp** *mtFlag DSM_MULTITHREAD* in Ihrer Anwendung an. Dies ist selbst dann erforderlich, wenn es sich bei der Anwendung um eine Einzelthread-Anwendung handelt. Mit diesem Flag wird das für die LAN-unabhängige Schnittstelle von IBM Spectrum Protect erforderliche Threading aktiviert.

Signale und Signalhandler

Die Anwendung verarbeitet Signale vom Benutzer oder vom Betriebssystem. Wenn der Benutzer die Tastenanschlagfolge **Strg+C** drückt, muss die Anwendung das Signal abfangen und Aufrufe **dsmTerminate** für jeden aktiven Thread senden. Rufen Sie dann **dsmCleanUp** auf, um die Anwendung zu beenden. Wenn Sitzungen nicht korrekt geschlossen werden, können auf dem Server nicht erwartete Ergebnisse auftreten.

Die Anwendung erfordert Signalhandler, wie beispielsweise SIGPIPE und SIGUSR1, für Signale, die die Beendigung der Anwendung zur Folge haben. Die Anwendung empfängt dann den Rückkehrcode von der API. Um beispielsweise SIGPIPE zu ignorieren, fügen Sie Ihrer Anwendung die folgende Anweisung hinzu: `signal(SIGPIPE, SIG_IGN)`. Nachdem diese Angabe hinzugefügt wurde, wird die Anwendung bei einer unterbrochenen Pipe nicht mehr beendet, sondern der korrekte Rückkehrcode zurückgegeben.

Sie können den untergeordneten Prozess Trusted Communication Agent (TCA) verwenden, wenn die Option `passwordaccess` auf `generate` gesetzt ist. Wenn Sie den TCA-Prozess verwenden, verwendet IBM Spectrum Protect das SIGCLD-Signal. Wenn Ihre Anwendung das SIGCLD-Signal verwendet, beachten Sie mögliche Interferenzen von IBM Spectrum Protect-Prozessen und der Art und Weise, wie SIGCLD verwendet wird. Weitere Informationen zur Verwendung von TCA finden Sie in „Sitzungssicherheit" auf Seite 17.

Sitzung starten oder beenden

IBM Spectrum Protect ist ein sitzungsbasiertes Produkt und alle Aktivitäten müssen innerhalb einer IBM Spectrum Protect-Sitzung ausgeführt werden. Zum Starten einer Sitzung startet die Anwendung den Aufruf **dsmInitEx**. Dieser Aufruf muss vor jedem anderen API-Aufruf ausgeführt werden, der nicht **dsmQueryApiVersionEx**, **dsmQueryCliOptions** oder **dsmSetUp** ist.

Die Funktion **dsmQueryCliOptions** kann nur vor dem Aufruf **dsmInitEx** aufgerufen werden. Die Funktion gibt die Werte wichtiger Optionen zurück, beispielsweise Optionsdateien, Komprimierungseinstellungen und Kommunikationsparameter. Der Aufruf **dsmInitEx** richtet eine Sitzung mit dem Server gemäß den Angaben in den Parametern ein, die in dem Aufruf übergeben werden oder die in der Optionsdatei definiert sind.

Der Clientknotenname, der Eignername und die Kennwortparameter werden an den Aufruf **dsmInitEx** übergeben. Beim Eignernamen muss die Groß-/

Kleinschreibung beachtet werden, beim Knotennamen und Kennwort nicht. Die Anwendungsclientknoten müssen im Server registriert werden, bevor eine Sitzung gestartet wird.

Jedes Mal, wenn ein API-Anwendungsclient eine Sitzung mit dem Server startet, wird der Clientanwendungstyp im Server registriert. Geben Sie als Wert für den Anwendungstyp immer eine Abkürzung für das Betriebssystem an, da dieser Wert in das Plattformfeld auf dem Server eingegeben wird. Die maximale Länge für die Zeichenfolge ist DSM_MAX_PLATFORM_LENGTH.

Der Funktionsaufruf **dsmInitEx** richtet die IBM Spectrum Protect-Sitzung mit der API-Konfigurationsdatei und -Optionsliste des Anwendungsclients ein. Der Anwendungsclient kann mit der API-Konfigurationsdatei und -Optionsliste eine Reihe von IBM Spectrum Protect-Optionen definieren. Diese Werte überschreiben die während der Installation in den Benutzerkonfigurationsdateien definierten Werte. Benutzer können die vom Administrator definierten Optionen nicht ändern. Wenn der Anwendungsclient keine bestimmte Konfigurationsdatei und Optionsliste hat, können Sie für diese beiden Parameter NULL angeben. Weitere Informationen zu Konfigurationsdateien finden Sie hier:

„Erläuterungen zu Konfigurations- und Optionsdateien" auf Seite 1

Der Funktionsaufruf **dsmInitEx** richtet die IBM Spectrum Protect-Sitzung ein. Dabei werden Parameter verwendet, die eine erweiterte Überprüfung ermöglichen.

Überprüfen Sie den Funktionsaufruf **dsmInitEx** und den Informationsrückkehrcode **dsmInitExOut**. Der Administrator hat die letzte Sitzung abgebrochen, wenn der Rückkehrcode ok ist (RC=ok) und der Informationsrückkehrcode (infoRC) DSM_RC_REJECT_LASTSESS_CANCELED lautet. Soll die aktuelle Sitzung sofort beendet werden, rufen Sie **dsmTerminate** auf.

Der Aufruf **dsmQuerySessOptions** gibt dieselben Felder wie der Aufruf **dsmQueryCliOptions** zurück. Der Aufruf kann nur in einer Sitzung gesendet werden. Die Werte geben die Clientoptionen wieder, die während dieser Sitzung gültig sind (Werte aus Optionsdateien und alle Überschreibungswerte aus dem Aufruf **dsmInitEx**).

Nach dem Start einer Sitzung kann die Anwendung einen Aufruf **dsmQuerySessInfo** senden, um die für diese Sitzung definierten Serverparameter zu bestimmen. Mit diesem Aufruf werden Elemente wie die Maßnahmendomäne und Transaktionsgrenzwerte an die Anwendung zurückgegeben.

Beenden Sie Sitzungen mit einem Aufruf **dsmTerminate**. Alle Verbindungen zum Server werden beendet und alle Ressourcen, die dieser Sitzung zugeordnet sind, werden freigegeben.

Ein Beispiel für das Starten und Beenden einer Sitzung finden Sie hier:

Abb. 2 auf Seite 19

In dem Beispiel wird eine Reihe von globalen und lokalen Variablen definiert, die in Aufrufen **dsmInitEx** und **dsmTerminate** verwendet werden. Im Aufruf **dsmInitEx** wird ein Verweis auf `dsmHandle` als Parameter verwendet, während im Aufruf **dsmTerminate** `dsmHandle` als Parameter verwendet wird. Das Beispiel in Abb. 3 auf Seite 19 zeigt die Details von **rcApiOut** an. Die Funktion **rcApiOut** ruft die API-Funktion **dsmRCMsg** auf, die einen Rückkehrcode in eine Nachricht umsetzt. Der

Aufruf **rcApiOut** gibt dann die Nachricht für den Benutzer aus. Eine Version von **rcApiOut** ist in der API-Musteranwendung enthalten. Die Funktion **dsmApiVersion** ist eine Typdefinition, die sich in der Headerdatei dsmapitd.h befindet.

Sitzungssicherheit

Das sitzungsbasierte System IBM Spectrum Protect verfügt über Sicherheitskomponenten, die Anwendungen das Starten von Sitzungen auf sichere Art und Weise ermöglichen. Diese Sicherheitsmaßnahmen verhindern den unbefugten Zugriff auf den Server und unterstützen die Gewährleistung der Systemintegrität.

Jede Sitzung, die mit dem Server gestartet wird, muss einen Anmeldeprozess durchlaufen und erfordert ein Kennwort. Wenn das Kennwort zusammen mit dem Knotennamen des Clients angegeben wird, ist beim Herstellen der Verbindung zum Server die korrekte Berechtigung gewährleistet. Der Anwendungsclient stellt der API dieses Kennwort zum Starten der Sitzung zur Verfügung.

Es gibt zwei Methoden der Kennwortverarbeitung: *passwordaccess=prompt* und *passwordaccess=generate*. Wenn Sie die Option *passwordaccess=prompt* verwenden, müssen Sie den Kennwortwert bei jedem Aufruf von **dsmInitEx** angeben. Sie können auch den Knotennamen und Eignernamen im **dsmInitEx**-Aufruf angeben.

Jedem Kennwort ist eine Ablaufzeit zugeordnet. Wenn ein **dsmInitEx**-Aufruf mit einem Rückkehrcode wegen eines abgelaufenen Kennworts (DSM_RC_REJECT_VERIFIER_EXPIRED) fehlschlägt, muss der Anwendungsclient den Aufruf **dsmChangePW** mit der Kennung eingeben, die von **dsmInitEx** zurückgegeben wird. Dadurch wird das Kennwort aktualisiert, bevor die Sitzung erfolgreich eingerichtet werden kann. Das Beispiel in Abb. 4 auf Seite 20 zeigt die Vorgehensweise bei der Änderung eines Kennworts mithilfe von **dsmChangePW**. Der Anmeldeeigner muss eine Rootbenutzer-ID oder eine ID des berechtigten Benutzers verwenden, um das Kennwort ändern zu können.

Bei der zweiten Methode, *passwordaccess=generate*, wird der Kennwortwert in einer Datei verschlüsselt und gespeichert. Der Knotenname und Eignername können nicht im Aufruf **dsmInitEx** angegeben werden und die Systemstandardwerte werden verwendet. Dadurch wird die Sicherheit der Kennwortdatei gewährleistet. Wenn das Kennwort abläuft, erstellt der Parameter generate ein neues und aktualisiert die Kennwortdatei automatisch.

Hinweise:

1. Wenn zwei verschiedene physische Maschinen über denselben IBM Spectrum Protect-Knotennamen verfügen oder wenn auf einem Knoten mehrere Pfade mit mehreren Serverzeilengruppen definiert sind, funktioniert *passwordaccess=generate* möglicherweise nur für die Zeilengruppe, die nach dem Ablauf der Kennwortgültigkeit als Erstes verwendet wird. Während des ersten Client/Server-Kontakts muss der Benutzer dasselbe Kennwort für jede Serverzeilengruppe separat angeben und für jede Zeilengruppe wird eine Kopie des Kennworts separat gespeichert. Wenn das Kennwort abläuft, wird ein neues Kennwort für die Zeilengruppe generiert, die eine Verbindung des ersten Client/Server-Kontakts herstellt. Alle nachfolgenden Versuche, eine Verbindung über andere Serverzeilengruppen herzustellen, schlagen fehl, weil es keine logische Verbindung zwischen den jeweiligen Kopien des alten Kennworts und der aktualisierten Kopie gibt, die von der Zeilengruppe generiert wird, die nach dem Ablauf der Kennwortgültigkeit als Erstes verwendet wird. In diesem Fall müssen Sie die Kennwörter vor dem Ablauf oder nach dem Ablauf als Wiederherstellungsmaßnahme dieser Situation wie folgt aktualisieren:

a. Führen Sie **dsmadmc** aus und aktualisieren Sie das Kennwort auf dem Server.
b. Führen Sie **dsmc -servername=stanza1** aus und verwenden Sie das neue Kennwort, um einen ordnungsgemäßen Eintrag zu generieren.
c. Führen Sie **dsmc -servername=stanza2** aus und verwenden Sie das neue Kennwort, um einen ordnungsgemäßen Eintrag zu generieren.

2. Für UNIX oder Linux: Nur der Rootbenutzer oder ein berechtigter Benutzer kann das Kennwort ändern, wenn *passwordaccess=prompt* verwendet wird. Nur der Rootbenutzer oder ein berechtigter Benutzer kann die Kennwortdatei starten, wenn *passwordaccess=generate* verwendet wird. Sie können den untergeordneten Prozess von Trusted Communication Agent (TCA) für die Kennwortverarbeitung verwenden. Die Anwendung muss dies berücksichtigen, weil ein untergeordneter Prozess und das SIGCLD-Signal verwendet werden. Der TCA wird in folgenden Situation nicht verwendet:
   - Die Option *passwordaccess* ist auf *prompt* gesetzt.
   - Der Anmeldebenutzer ist Root.
   - Der Aufrufende der Funktion muss ein berechtigter Benutzer sein.

   Einschränkung: Die Optionen `users` und `groups` werden nicht erkannt.

Eine Anwendung kann den Benutzerzugriff mit anderen Mitteln beschränken, z. B. durch Zugriffsfilter.

Anwendungen, die mehrere IP-Verbindungen zu einem einzelnen IBM Spectrum Protect-Server verwenden, sollten denselben Knotennamen und dasselbe IBM Spectrum Protect-Clientkennwort für jede Sitzung verwenden. Gehen Sie wie folgt vor, um diese Unterstützung zu aktivieren:

1. Definieren Sie eine IBM Spectrum Protect-Serverzeilengruppe in der Datei dsm.sys.
2. Geben Sie für die Verbindungen, die nicht die IP-Standardadresse verwenden, die Optionswerte für die *TCPserver*-Adresse und *TCPport* im Aufruf **dsmInitEx** an.

Diese Werte überschreiben die IP-Verbindungsdaten, aber die Sitzung verwendet weiterhin dieselben Knoten- und Kennwortinformationen der Zeilengruppe in dsm.sys.

Anmerkung: Knoten in einem Cluster verwenden ein Kennwort gemeinsam.

```
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);
}
```

Abbildung 2. Beispiel für das Starten und Beenden einer Sitzung

```
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;
}
```

Abbildung 3. Details für 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;
```

Abbildung 4. Beispiel für eine Kennwortänderung

Option passwordaccess ohne TCA auf generate setzen

Der Trusted Communication Agent (TCA) ist ein untergeordneter Prozess, der normalerweise den Zugriff auf die geschützte Kennwortdatei steuert. Auf UNIX- und Linux-Systemen können Sie sich als berechtigter Benutzer anmelden und die Option passwordaccess auf generate setzen, ohne den TCA zu starten.

Führen Sie die folgenden Schritte durch, um passwordaccess ohne TCA auf generate zu setzen:

1. Geben Sie beim Schreiben der Anwendung einen Aufruf **dsmSetUp** an, der *argv[0]* übergibt. *Argv[0]* enthält den Namen der Anwendung, mit der die API aufgerufen wird. Die Anwendung kann durch einen berechtigten Benutzer ausgeführt werden, der Administrator muss jedoch den Anmeldenamen für den berechtigten Benutzer festlegen.
2. Setzen Sie das Bit für die aktuelle Benutzer-ID (Bit S) für die ausführbare Anwendung auf Ein. Der Eigner der ausführbaren Datei der Anwendung kann dann zu einem berechtigten Benutzer werden und eine Kennwortdatei erstellen, Kennwörter aktualisieren und Anwendungen ausführen. Der Eigner der ausführbaren Datei der Anwendung muss mit der Benutzer-ID übereinstimmen, die das Programm ausführt. In dem folgenden Beispiel ist user1 der *Benutzer* und applA der Name der ausführbaren Datei der Anwendung; user1 hat Schreib-/Lesezugriff auf das Verzeichnis /home/user1. Die ausführbare Datei applA hat die folgenden Berechtigungen:
   ```
   -rwsr-xr-x user1    group1   applA
   ```
3. Weisen Sie die Benutzer der Anwendung an, für die Anmeldung den Namen des berechtigten Benutzers zu verwenden. IBM Spectrum Protect prüft, ob die Anmelde-ID mit dem Eigner der ausführbaren Anwendung übereinstimmt, bevor der Zugriff auf die geschützte Kennwortdatei erlaubt wird.
4. Setzen Sie die Option passworddir in der Datei dsm.sys so, dass sie auf ein Verzeichnis verweist, für das dieser Benutzer über Schreib-/Lesezugriff verfügt. Geben Sie beispielsweise die folgende Zeile in der Serverzeilengruppe der Datei dsm.sys ein:
   ```
   passworddir /home/user1
   ```
5. Erstellen Sie die Kennwortdatei und stellen Sie sicher, dass der berechtigte Benutzer Eigner der Datei ist.
6. Melden Sie sich als user1 an und führen Sie app1A aus.
7. Rufen Sie **dsmSetUp** auf und übergeben Sie *argv*.

Benutzer mit Verwaltungsaufgaben mit Clienteignerberechtigung erstellen

Ein Benutzer mit Verwaltungsaufgaben mit Clienteignerberechtigung kann Parameter im Funktionsaufruf **dsmInitEx** definieren, um Sitzungen zu starten. Dieser Benutzer kann als „Benutzer mit Verwaltungsaufgaben" mit Sicherungs- und Zurückschreibungsberechtigung für die definierten Knoten arbeiten.

Führen Sie auf dem Server die folgenden Schritte aus, um die Clienteignerberechtigung zu erhalten:

1. Definieren Sie den Benutzer mit Verwaltungsaufgaben:

   ```
   REGister Admin Administratorname Kennwort
   ```

   Dabei gilt Folgendes:

   - *Administratorname* ist der Name des Benutzers mit Verwaltungsaufgaben.
   - *Kennwort* ist das Administratorkennwort.

2. Definieren Sie die Berechtigungsstufe. Benutzer mit System- oder Maßnahmenberechtigung verfügen auch über Clienteignerberechtigung.

   ```
   Grant Authority Administratorname Klassen Berechtigung node
   ```

   Dabei gilt Folgendes:

   - *Administratorname* ist der Benutzer mit Verwaltungsaufgaben.
   - *Klassen* ist der Knoten.
   - *Berechtigung* hat eine der folgenden Berechtigungsstufen:
     - `owner`: vollständige Sicherungs- und Zurückschreibungsberechtigung für den Knoten
     - `node`: einzelner Knoten
     - `domain`: Gruppe von Knoten

3. Definieren Sie den Zugriff auf einen einzelnen Knoten.

   ```
   Register Node Knotenname Kennwort
   userid=Benutzer-ID
   ```

   Dabei gilt Folgendes:

   - *Knotenname* ist der Clientbenutzerknoten
   - *Kennwort* ist das Kennwort für den Clientbenutzerknoten
   - *Benutzer-ID* ist der Name des Benutzers mit Verwaltungsaufgaben

Wenn die Anwendung den Benutzer mit Verwaltungsaufgaben verwendet, wird die Funktion **dsmInitEx** mit den Parametern `userName` und `userNamePswd` aufgerufen.

```
dsmInitEx
              clientNodeName = NULL
              clientOwnerName = NULL
      clientPassword = NULL
      userName = 'Name des Benutzers mit Verwaltungsaufgaben'
      userNamePswd = 'Kennwort des Benutzers mit Verwaltungsaufgaben'
```

Sie können Option `passwordaccess` auf `generate` oder `prompt` setzen. Mit jedem der Parameter wird über den Wert `userNamePswd` die Sitzung gestartet. Wenn die Sitzung gestartet wird, kann ein Sicherungs- oder Zurückschreibungsprozess für diesen Knoten ausgeführt werden.

Objektnamen und -IDs

Der IBM Spectrum Protect-Server ist ein Objektspeicherserver, dessen Funktion in erster Linie darin besteht, benannte Objekte effizient zu speichern und abzurufen. Die Objekt-ID ist für jedes Objekt eindeutig und bleibt dem Objekt während der gesamten Lebensdauer zugeordnet, *es sei denn*, das Objekt wird exportiert oder importiert.

Um diese Voraussetzung zu erfüllen, verfügt IBM Spectrum Protect über zwei Hauptspeicherbereiche, Datenbank und Datenspeicher.

- Die Datenbank enthält alle Metadaten wie den Namen und die Attribute, die den Objekten zugeordnet sind.
- Der Datenspeicher enthält die Objektdaten. Der Datenspeicher ist eigentlich eine Speicherhierarchie, die der Systemadministrator definiert. Daten werden abhängig von Kosten und Zugriffsanforderungen entweder effizient auf Onlinedatenträgern oder auf Offlinedatenträgern gespeichert und verwaltet.

Jedem Objekt, das auf dem Server gespeichert ist, ist ein Name zugeordnet. Der Client steuert die folgenden Schlüsselkomponenten dieses Namens:

- Dateibereichsname
- Übergeordneter Name
- Untergeordneter Name
- Objekttyp

Beim Treffen von Entscheidungen in Bezug auf die Benennung von Objekten für eine Anwendung muss unter Umständen ein externer Name für die vollständigen Objektnamen beim Endbenutzer verwendet werden. Der Endbenutzer muss unter Umständen das Objekt in einer Ein- oder Ausschlussanweisung angeben, wenn die Anwendung ausgeführt wird. Die genaue Syntax des Objektnamens in diesen Anweisungen ist plattformabhängig. Unter dem Betriebssystem Windows wird der Laufwerkbuchstabe, der dem Dateibereich zugeordnet ist und nicht der Dateibereichsname selbst in der Ein- oder Ausschlussanweisung verwendet.

Der Wert für die Objekt-ID, der beim Erstellen des Objekts zugeordnet wurde, ist möglichweise nicht derselbe wie bei der Ausführung eines Zurückschreibungsprozesses. Anwendungen sollten den Objektnamen speichern und dann abfragen, um die aktuelle Objekt-ID abzurufen, bevor eine Zurückschreibung ausgeführt wird.

Dateibereichsname

Der Dateibereichsname ist eine der wichtigsten Speicherkomponenten. Dabei kann es sich um den Namen eines Dateisystems, eines Plattenlaufwerks oder eines anderen übergeordneten Qualifikationsmerkmals handeln, mit dem zusammengehörige Daten gruppiert werden.

IBM Spectrum Protect identifiziert mithilfe des Dateibereichs das Dateisystem oder das Plattenlaufwerk, auf dem sich die Daten befinden. So ist es möglich, Aktionen für alle Entitäten in einem Dateibereich auszuführen, z. B. Abfragen aller Objekte in einem bestimmten Dateibereich. Da der Dateibereich eine derart wichtige Komponente der Namenskonvention von IBM Spectrum Protect ist, verwenden Sie spezielle Aufrufe zum Registrieren, Aktualisieren, Abfragen und Löschen von Dateibereichen.

Der Server verfügt auch über Verwaltungsbefehle, um die Dateibereiche in jedem Knoten in IBM Spectrum Protect-Speicher abzufragen und, falls erforderlich, zu lö-

schen. Allen Daten, die vom Anwendungsclient gespeichert werden, muss ein Dateibereichsname zugeordnet sein. Wählen Sie den Namen mit Bedacht, um ähnliche Daten im System zu gruppieren.

Um mögliche Kollisionen zu verhindern, dürfen die Dateibereichsnamen, die ein Anwendungsclient auswählt, nicht mit den Namen übereinstimmen, die ein Client für Sichern/Archivieren verwenden würde. Der Anwendungsclient muss seine Dateibereichsnamen publizieren, damit Endbenutzer die Objekte für Einschluss-/Ausschlussanweisungen, falls erforderlich, identifizieren können.

Anmerkung: Auf Windows-Plattformen ist einem Dateibereich ein Laufwerkbuchstabe zugeordnet. Wenn Sie einen Dateibereich registrieren oder aktualisieren, müssen Sie den Laufwerkbuchstaben angeben. Da die Einschluss-/Ausschlussliste auf den Laufwerkbuchstaben Bezug nimmt, müssen Sie jeden Laufwerkbuchstaben mit seinem zugehörigen Dateibereich protokollieren. In dem Musterprogramm dapismp ist der Laufwerkbuchstabe standardmäßig auf "G" gesetzt.

Weitere Informationen zu den Musterprogrammen finden Sie in Kapitel 2, „API-Musteranwendung erstellen und ausführen", auf Seite 5.

Übergeordnete und untergeordnete Namen

Zwei weitere Komponenten des Objektnamens sind das Qualifikationsmerkmal für den übergeordneten Namen und das Qualifikationsmerkmal für den untergeordneten Namen. Das Qualifikationsmerkmal für den übergeordneten Namen ist der Verzeichnispfad, in den das Objekt gehört; das Qualifikationsmerkmal für den untergeordneten Namen ist der eigentliche Name des Objekts in diesem Verzeichnispfad.

Wenn der Dateibereichsname, der übergeordnete Name und der untergeordnete Name verknüpft werden, müssen sie einen syntaktisch korrekten Namen auf dem Betriebssystem bilden, auf dem der Client ausgeführt wird. Es ist nicht erforderlich, dass der Name als ein Objekt auf dem System vorhanden ist oder den tatsächlichen Daten im lokalen Dateisystem ähnelt. Der Name muss jedoch den Standardbenennungsregeln entsprechen, damit er von den Aufrufen **dsmBindMC** korrekt verarbeitet werden kann. Überlegungen zur Benennung in Bezug auf die Maßnahmenverwaltung finden Sie in „Erläuterungen zu Sicherungs- und Archivierungsobjekten" auf Seite 43.

Objekttyp

Der Objekttyp identifiziert das Objekt entweder als eine Datei oder als ein Verzeichnis. Eine Datei ist ein Objekt, das sowohl Attribute als auch Binärdaten enthält; ein Verzeichnis ist ein Objekt, das nur Attribute enthält.

Tabelle 7 zeigt den Anwendungsclient-Code für Objektnamen nach Plattform.

Tabelle 7. Beispiele für Anwendungsobjektnamen nach Plattform

Plattform	Client-Code für Objektname
UNIX oder Linux	/myfs/highlev/lowlev

Tabelle 7. Beispiele für Anwendungsobjektnamen nach Plattform (Forts.)

Plattform	Client-Code für Objektname
Windows	"myvol\\highlev\\lowlev" **Anmerkung:** Auf einer Windows-Plattform wird ein doppelter umgekehrter Schrägstrich in einen einfachen umgekehrten Schrägstrich umgesetzt, da ein umgekehrter Schrägstrich das Escapezeichen ist. Auf der UNIX- oder Linux-Plattform beginnen Dateibereichsnamen mit einem Schrägstrich, während sie auf der Windows-Plattform nicht mit einem Schrägstrich beginnen dürfen.

Zugriff auf Objekte als Sitzungseigner

Jedem Objekt ist ein Eignername zugeordnet. Die Regeln, die festlegen, auf welche Objekte zugegriffen wird, sind von dem Eignernamen abhängig, der beim Starten einer Sitzung verwendet wird. Steuern Sie mithilfe dieses Werts für den Sitzungseigner den Zugriff auf das Objekt.

Der Sitzungseigner wird während des Aufrufs **dsmInitEx** im Parameter *clientOwnerNameP* festgelegt. Wenn Sie eine Sitzung mit **dsmInitEx** unter Angabe des Eignernamens *NULL* starten und *passwordaccess=prompt* verwenden, wird diesem Sitzungseigner Sitzungsberechtigung (Root- oder berechtigter Benutzer) zugeordnet. Dies trifft auch zu, wenn Sie sich mit einer Rootbenutzer-ID oder einer ID des berechtigten Benutzers anmelden und *passwordaccess=generate* verwenden. Während einer auf diese Weise gestarteten Sitzung können Sie jede Aktion für jedes Objekt ausführen, dessen Eigner dieser Knoten ist; dies ist unabhängig vom tatsächlichen Eigner dieses Objekts.

Wenn eine Sitzung mit einem bestimmten Eignernamen gestartet wird, kann die Sitzung nur Aktionen für Objekte ausführen, denen dieser Objekteignername zugeordnet ist. Allen Sicherungen oder Archivierungen in dem System muss dieser Eignername zugeordnet sein. Alle Abfragen, die ausgeführt werden, geben nur die Werte zurück, denen dieser Eignername zugeordnet ist. Der Wert für den Objekteigner wird während des Aufrufs **dsmSendObj** im Feld **owner** der Struktur **ObjAttr** festgelegt. Bei einem Eignernamen muss die Groß-/Kleinschreibung beachtet werden. In Tabelle 8 sind die Bedingungen zusammengefasst, unter denen ein Benutzer Zugriff auf ein Objekt hat.

Tabelle 8. Zusammenfassung des Benutzerzugriffs auf Objekte

Sitzungseigner	Objekteigner	Benutzerzugriff
NULL (Root, Systemeigner)	„ ” (leere Zeichenfolge)	Ja
NULL	Bestimmter Name	Ja
Bestimmter Name	„ ” (leere Zeichenfolge)	Nein
Bestimmter Name	Derselbe Name	Ja
Bestimmter Name	Anderer Name	Nein

Knoten- und eignerübergreifender Zugriff auf Objekte

Drei Funktionsaufrufe unterstützen knoten- und eignerübergreifenden Zugriff auf derselben Plattform: **dsmSetAccess**, **dsmDeleteAccess** und **dsmQueryAccess**. Diese Funktionen in Verbindung mit den Zeichenfolgeoptionen *-fromnode* und *-fromowner*, die in **dsmInitEx** übergeben werden, gestatten einen vollständigen knotenübergreifenden Abfrage-, Zurückschreibungs- und Abrufprozess über die API.

Benutzer A auf Knoten A verwendet beispielsweise den Funktionsaufruf **dsmSetAccess**, um Benutzer B auf Knoten B Zugriff auf seine Sicherungen unter dem Dateibereich /db zu geben. Die Zugriffsregel wird wie folgt angezeigt:

ID	Typ	Knoten	Benutzer	Pfad
1	Sicherung	Knoten B	Benutzer B	/db/*/*

Wenn sich Benutzer B bei Knoten B anmeldet, lautet die Optionszeichenfolge für **dsmInitEx**:

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

Diese Optionen werden für diese Sitzung definiert. Alle Abfragen greifen auf die Dateibereiche und Dateien des Knotens A zu. Sicherungen und Archivierungen sind nicht zulässig. Aus den Dateibereichen, auf die Benutzer B Zugriff hat, sind nur Abfrage-, Zurückschreibungs- und Abrufprozesse zulässig. Wenn die Anwendung versucht, bei einer Anmeldung mit einer definierten Option *-fromnode* oder *-fromowner* eine Operation mit **dsmBeginTxn** auszuführen (z. B. Sicherung oder Aktualisierung), schlägt **dsmBeginTxn** mit dem Rückkehrcode DSM_RC_ABORT_NODE_NOT_AUTHORIZED fehl. Weitere Informationen finden Sie bei den einzelnen Funktionsaufrufen und in „**dsmInitEx**" auf Seite 127.

Tipp: Unter UNIX und Linux können Sie *–fromowner=root* in der Optionszeichenfolge angeben, die im Funktionsaufruf **dsmInitEx** übergeben wird. Dadurch erhalten Benutzer ohne Rootberechtigung Zugriff auf Dateien des Root, wenn ein set access ausgeführt wurde.

Verwenden Sie die Option *asnodename* in der Optionszeichenfolge **dsmInitEx** mit der entsprechenden Funktion, um Daten unter dem Zielknotennamen auf dem IBM Spectrum Protect-Server zu sichern, zu archivieren, zurückzuschreiben, abzurufen, abzufragen oder zu löschen. Informationen zur Aktivierung dieser Option finden Sie in „Mehrere Knoten mit Clientknoten-Proxy-Unterstützung sichern" auf Seite 87.

Dateibereiche verwalten

Da Dateibereiche für den Betrieb des Systems wichtig sind, wird eine separate Gruppe von Aufrufen zum Registrieren, Aktualisieren und Löschen von Dateibereichs-IDs verwendet. Bevor Sie Objekte, die einem Dateibereich auf dem System zugeordnet sind, speichern können, müssen Sie den Dateibereich zunächst bei IBM Spectrum Protect registrieren.

Führen Sie diese Task mithilfe des Aufrufs **dsmRegisterFS** aus. Weitere Informationen zu Objektnamen und -IDs finden Sie in „Objektnamen und -IDs" auf Seite 22.

Die Dateibereichs-ID ist das Qualifikationsmerkmal der höchsten Ebene in einer dreiteiligen Namenshierarchie. Die Gruppierung zusammengehöriger Daten in einem Dateibereich erleichtert die Verwaltung dieser Daten erheblich. Beispielsweise

kann entweder der Anwendungsclient oder der IBM Spectrum Protect-Serveradministrator einen Dateibereich und alle Objekte in diesem Dateibereich löschen.

Dateibereiche ermöglichen es dem Anwendungsclient außerdem, dem Server Informationen zu dem Dateibereich bereitzustellen, die der Administrator dann abfragen kann. Diese Informationen werden bei der Abfrage in der Struktur **qryRespFSData** zurückgegeben und umfassen die folgenden Dateisysteminformationen:

Typ	Definition
fstype	Der Dateibereichstyp. Dieses Feld ist eine Zeichenfolge, die vom Anwendungsclient definiert wird.
fsAttr[Plattform].fsInfo	Ein Feld für Clientinformationen, das für clientspezifische Daten verwendet wird.
capacity	Der gesamte Speicherbereich in dem Dateibereich.
occupancy	Die Größe des Speicherbereichs, der momentan in dem Dateibereich belegt ist.
backStartDate	Die Zeitmarke, die angibt, wann die letzte Sicherung gestartet wurde (durch Senden eines Aufrufs **dsmUpdateFS** festgelegt).
backCompleteDate	Die Zeitmarke, die angibt, wann die letzte Sicherung beendet wurde (durch Senden eines Aufrufs **dsmUpdateFS** festgelegt).

Die Verwendung von 'capacity' und 'occupancy' ist vom Anwendungsclient abhängig. Einige Anwendungen benötigen möglicherweise keine Informationen zur Größe des Dateibereichs; in diesem Fall kann für diese Felder standardmäßig `0` angenommen werden. Weitere Informationen zum Abfragen von Dateibereichen finden Sie in „IBM Spectrum Protect-System abfragen" auf Seite 33.

Nachdem ein Dateibereich beim System registriert wurde, können Sie jederzeit Objekte sichern oder archivieren. Rufen Sie **dsmUpdateFS** auf, um die Felder 'occupancy' und 'capacity' des Dateibereichs nach einer Sicherungs- oder Archivierungsoperation zu aktualisieren. Dieser Aufruf stellt sicher, dass die Werte für Belegung (occupancy) und Kapazität (capacity) des Dateisystems aktuell sind. Sie können auch die Felder **fsinfo**, **backupstart** und **backupcomplete** aktualisieren.

Wenn Sie das Datum ihrer letzten Sicherungen überwachen möchten, geben Sie einen Aufruf **dsmUpdateFS** ein, bevor Sie die Sicherung starten. Setzen Sie die Aktualisierungsaktion auf DSM_FSUPD_BACKSTARTDATE. Dies zwingt den Server, das Feld **backStartDate** des Dateibereichs auf die aktuelle Uhrzeit zu setzen. Nachdem die Sicherung für diesen Dateibereich abgeschlossen ist, geben Sie einen Aufruf **dsmUpdateFS** mit der Angabe DSM_FSUPD_BACKCOMPLETEDATE für die Aktualisierungsaktion ein. Dieser Aufruf erstellt am Ende der Sicherung eine Zeitmarke.

Wenn ein Dateibereich nicht mehr benötigt wird, können Sie ihn mit dem Befehl **dsmDeleteFS** löschen. Auf einem UNIX- oder Linux-Betriebssystem können nur Rootbenutzer oder berechtigte Benutzer Dateibereiche löschen.

Die Beispiele in Abb. 5 auf Seite 27 veranschaulichen die Verwendung der drei Dateibereichsaufrufe für UNIX oder Linux. Ein Beispiel für die Verwendung der drei Dateibereichsaufrufe für Windows enthält der Code des auf Ihrem System installierten Beispielprogramms.

```
/* Den Dateibereich registrieren, sofern dies noch nicht erfolgt ist. */

dsInt16        rc;
regFSData    fsData;
char         fsName[DSM_MAX_FSNAME_LENGTH];
char         smpAPI[] = "Muster-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,"FsInfo für Muster-API");
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;  /* bereits ausgeführt */
if (rc)
{
   printf("Dateibereichsregistrierung fehlgeschlagen: ");
   rcApiOut(dsmHandle, rc);
   free(bkup_buff);
   return (RC_SESSION_FAILED);
}
```

Abbildung 5. Beispiel für die Verwendung von drei Dateibereichen, Teil 1

```
/* Dateibereich aktualisieren. */

dsmFSUpd     updFilespace;         /* Aktual. des Dateibereichs */

updFilespace.stVersion = dsmFSUpdVersion;
updFilespace.fsType = 0;             /* keine Änderung */
updFilespace.occupancy.hi = 0;
updFilespace.occupancy.lo = 50;
updFilespace.capacity.hi = 0;
updFilespace.capacity.lo = 200;
strcpy(updFilespace.fsAttr.unixFSAttr.fsInfo,
       "Aktualisierung für Dateibereich") ;
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);
```

Abbildung 6. Beispiel für die Verwendung von drei Dateibereichen, Teil 2

```
/* Dateibereich löschen. */

printf("\nDateibereich wird gelöscht
rc = dsmDeleteFS (dsmHandle,fsName,DSM_REPOS_ALL);
if (rc)
{
   printf(" FEHLGESCHLAGEN!!! ");
   rcApiOut(dsmHandle, rc);
}
else printf("  OK!\n");
```

Abbildung 7. Beispiel für die Verwendung von drei Dateibereichen, Teil 3

Objekte Verwaltungsklassen zuordnen

Ein wichtiges Merkmal von IBM Spectrum Protect ist die Verwendung von Maßnahmen (Verwaltungsklassen), die definieren, wie Objekte im IBM Spectrum Protect-Speicher gespeichert und verwaltet werden. Ein Objekt wird bei seiner Sicherung oder Archivierung einer Verwaltungsklasse zugeordnet.

Diese Verwaltungsklasse legt Folgendes fest:
- Wie viele Versionen des Objekts aufbewahrt werden, wenn es gesichert wird
- Wie lange Archivierungskopien aufbewahrt werden
- An welcher Position das Objekt in der Speicherhierarchie auf dem Server eingefügt werden soll

Verwaltungsklassen bestehen sowohl aus Sicherungskopiengruppen als auch aus Archivierungskopiengruppen. Eine Kopiengruppe besteht aus einer Reihe von Attributen, die die Verwaltungsmaßnahmen für ein Objekt definieren, das gesichert oder archiviert wird. Wenn eine Sicherungsoperation ausgeführt wird, werden die Attribute in der Sicherungskopiengruppe angewendet. Wenn eine Archivierungsoperation ausgeführt wird, werden die Attribute in der Archivierungskopiengruppe angewendet.

Die Sicherungs- oder Archivierungskopiengruppe in einer bestimmten Verwaltungsklasse kann leer oder NULL sein. Wird ein Objekt an die NULL-Sicherungskopiengruppe gebunden, kann dieses Objekt nicht gesichert werden. Wird ein Objekt an die NULL-Archivierungskopiengruppe gebunden, kann dieses Objekt nicht archiviert werden.

Da die Verwendung einer Maßnahme ein sehr wichtiger Bestandteil von IBM Spectrum Protect ist, ist es für die API erforderlich, dass allen Objekten, die an den Server gesendet werden, zunächst mithilfe des Aufrufs **dsmBindMC** eine Verwaltungsklasse zugeordnet wird. Mit der IBM Spectrum Protect-Software können Sie eine Einschluss-/Ausschlussliste für die Verwaltungsklassenbindung verwenden. Der Aufruf **dsmBindMC** verwendet die aktuelle Einschluss-/Ausschlussliste, um die Verwaltungsklassenbindung durchzuführen.

Einschlussanweisungen (include) können eine bestimmte Verwaltungsklasse einem Sicherungs- oder Archivierungsobjekt zuordnen. Ausschlussanweisungen (exclude) können die Sicherung von Objekten, aber nicht die Archivierung von Objekten verhindern.

Für die API ist erforderlich, dass **dsmBindMC** vor der Sicherung oder Archivierung eines Objekts aufgerufen wird. Der Aufruf **dsmBindMC** gibt eine mcBindKey-Struktur zurück, die Informationen zu Verwaltungsklassen und Kopiengruppen enthält, die dem Objekt zugeordnet sind. Überprüfen Sie das Ziel für die Kopiengruppe, bevor eine Sendeoperation ausgeführt wird. Wenn Sie mehrere Objekte in einer einzigen

Transaktion senden, müssen die Objekte dasselbe Ziel für Kopiengruppe haben. Der Funktionsaufruf **dsmBindMC** gibt folgende Informationen zurück:

Tabelle 9. Im Aufruf dsmBindMC zurückgegebene Informationen

Informationen	Beschreibung
Verwaltungsklasse	Der Name der Verwaltungsklasse, die dem Objekt zugeordnet wurde. Der Anwendungsclient kann den Aufruf **dsmBeginQuery** senden, um alle Attribute dieser Verwaltungsklasse zu bestimmen.
Sicherungskopiengruppe	Gibt Auskunft darüber, ob eine Sicherungskopiengruppe für diese Verwaltungsklasse vorhanden ist. Wenn eine Sicherungsoperation ausgeführt wird und keine Sicherungskopiengruppe vorhanden ist, kann dieses Objekt nicht an den Speicher gesendet werden. Sie empfangen einen Fehlercode, wenn Sie eine Sendeoperation mit dem Aufruf **dsmSendObj** auszuführen versucht haben.
Kopienzielort für Sicherungen	Dieses Feld gibt den Speicherpool an, an den die Daten gesendet werden. Wenn Sie eine Sicherungstransaktion mit mehreren Objekten ausführen, müssen alle Kopienzielorte innerhalb dieser Transaktion identisch sein. Wenn ein Objekt andere Kopienzielorte als vorherige Objekte in der Transaktion hat, beenden Sie die aktuelle Transaktion und beginnen Sie eine neue Transaktion, damit Sie das Objekt senden können. Sie empfangen einen Fehlercode, wenn Sie versuchen, Objekte innerhalb derselben Transaktion an verschiedene Kopienzielorte zu senden.
Archivierungskopiengruppe	Gibt Auskunft darüber, ob eine Archivierungskopiengruppe für diese Verwaltungsklasse vorhanden ist. Wenn eine Archivierungsoperation ausgeführt wird und keine Archivierungskopiengruppe vorhanden ist, kann dieses Objekt nicht an den Speicher gesendet werden. Sie empfangen einen Fehlercode, wenn Sie eine Sendeoperation mit dem Aufruf **dsmSendObj** auszuführen versucht haben.
Kopienzielort für Archivierungen	Dieses Feld gibt den Speicherpool an, an den die Daten gesendet werden. Wenn Sie eine Archivierungstransaktion mit mehreren Objekten ausführen, müssen alle Kopienzielorte innerhalb dieser Transaktion identisch sein. Wenn ein Objekt andere Kopienzielorte als vorherige Objekte in der Transaktion hat, beenden Sie die aktuelle Transaktion und beginnen Sie eine neue Transaktion, bevor Sie das Objekt senden. Sie empfangen einen Fehlercode, wenn Sie versuchen, Objekte innerhalb derselben Transaktion an verschiedene Kopienzielorte zu senden.

Sicherungskopien eines Objekts können an eine andere Verwaltungsklasse erneut gebunden werden, wenn eine nachfolgende Sicherung mit demselben Objektnamen ausgeführt wird, in der eine andere als die ursprüngliche Verwaltungsklasse verwendet wird. Wenn Sie beispielsweise ObjektA sichern und an Verwaltungsklasse1 binden und später ObjektA sichern und an Verwaltungsklasse2 binden, werden bei der aktuellsten Sicherung alle inaktiven Kopien an Verwaltungsklasse2 erneut gebunden. Die in Verwaltungsklasse2 definierten Parameter steuern dann alle Kopien. Wenn sich der Zielort ändert, werden die Daten jedoch nicht versetzt.

Sie können Sicherungskopien auch mit dem Aufruf **dsmUpdateObj** oder **dsmUpdateObjEx** mit der Aktion DSM_BACKUPD_MC an eine andere Verwaltungsklasse erneut binden.

Zugehörige Verweise:

Option Deduplication

Verwaltungsklassen abfragen

Anwendungen können Verwaltungsklassen abfragen, um zu bestimmen, welche Verwaltungsklasse für einen bestimmten Knoten gültig sind, und um zu bestimmen, welche Attribute sich in der Verwaltungsklasse befinden.

Sie können Objekte nur mithilfe des Aufrufs **dsmBindMC** an Verwaltungsklassen binden. Möglicherweise sollen Ihre Anwendungen die Verwaltungsklassenattribute abfragen und den Endbenutzern anzeigen. Weitere Informationen finden Sie in „IBM Spectrum Protect-System abfragen" auf Seite 33.

In dem Beispiel in Abb. 8 wird mithilfe einer Anweisung 'switch' zwischen Sicherungs- und Archivierungsoperationen beim Aufrufen von **dsmBindMC** unterschieden. Die von diesem Aufruf zurückgegebenen Informationen werden in der Struktur **MCBindKey** gespeichert.

```
dsUint16_t     send_type;
dsUint32_t     dsmHandle;
dsmObjName   objName;     /* Struktur, die den Objektnamen enthält */
mcBindKey    MCBindKey;   /* Verwaltungsklasseninformationen       */
char         *dest;       /* Zielwert speichern                    */

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 fehlgeschlagen: ");
   rcApiOut(dsmHandle, rc);
   rc = (RC_SESSION_FAILED);
   return;
}
```

Abbildung 8. Beispiel für die Zuordnung einer Verwaltungsklasse zu einem Objekt

Verfall/Löschen anhalten und freigeben

Sie können das Löschen und den Verfall bestimmter Archivierungsobjekte als Reaktion auf eine anstehende oder laufende Aktion, die das Anhalten bestimmter Daten erfordert, anhalten. Falls eine Aktion eingeleitet wird, die möglicherweise Zugriff auf Daten erfordert, müssen diese Daten verfügbar sein, bis die Aktion abgeschlossen ist und der Zugriff auf die Daten im Rahmen dieses Prozesses nicht mehr erforderlich ist. Nachdem festgestellt wurde, dass die Aussetzung nicht mehr erforderlich ist (Freigabe), wird die normale Zeitplanung für Löschen und Verfall über den ursprünglichen Aufbewahrungszeitraum wieder aufgenommen.

Überprüfen Sie, ob der Server lizenziert ist, indem Sie einen Testaufruf **dsmRetentionEvent** ausgeben:

1. Führen Sie eine Abfrage nach einem einzelnen Objekt aus, das angehalten werden soll, und rufen Sie die ID ab.
2. Setzen Sie den Aufruf **dsmBeginTxn**, den Aufruf **dsmRetentionEvent** mit der Angabe `Hold` und den Aufruf **dsmEndTxn** ab.

3. Ist der Server nicht lizenziert, empfangen Sie ein Votum 'abort' (Abbrechen) mit dem Ursachencode DSM_RC_ABORT_LICENSE_VIOLATION.

Einschränkungen:
1. Sie können in einer einzigen Transaktion nicht mehrere Aufrufe **dsmRetentionEvent** absetzen.
2. Es ist nicht möglich, das Anhalten eines Objekts anzufordern, das bereits angehalten ist.

1. Führen Sie die folgenden Schritte aus, um Objekte anzuhalten:
   a. Fragen Sie den Server nach allen Objekten ab, die angehalten werden sollen. Rufen Sie die Objekt-ID für jedes Objekt ab.
   b. Setzen Sie zunächst einen Aufruf **dsmBeginTxn** und dann einen Aufruf **dsmRetentionEvent** mit der Liste der Objekte ab, gefolgt von einem Aufruf **dsmEventType**: eventHoldObj. Überschreitet die Anzahl Objekte den Wert von maxObjPerTxn, müssen Sie mehrere Transaktionen verwenden.
   c. Verwenden Sie die Antwort qryRespArchiveData im Funktionsaufruf **dsmGetNextQObj**, um zu bestätigen, dass die Objekte angehalten wurden. Überprüfen Sie den Wert von objHeld in qryRespArchiveData.
2. Führen Sie die folgenden Schritte aus, um Objekte freizugeben:
   a. Fragen Sie den Server nach allen angehaltenen Objekten ab, die freigegeben werden sollen. Rufen Sie die Objekt-ID für jedes Objekt ab.
   b. Setzen Sie zunächst einen Aufruf **dsmBeginTxn** und dann einen Aufruf **dsmRetentionEvent** mit der Liste der Objekte ab, gefolgt von einem Aufruf **dsmEventType**: eventReleaseObj. Überschreitet die Anzahl Objekte den Wert von maxObjPerTxn, müssen Sie mehrere Transaktionen verwenden.
   c. Verwenden Sie die Antwort **qryRespArchiveData** im Funktionsaufruf dsmGetNextQObj, um zu überprüfen, ob die angehaltenen Objekte freigegeben wurden. Überprüfen Sie den Wert von objHeld in **qryRespArchiveData**.

Aufbewahrungsschutz für Archivierungsdaten

Von IBM Spectrum Protect gesteuerte Daten können nicht durch Agenten ohne entsprechende Berechtigung, wie beispielsweise eine Person oder ein Programm, geändert werden. Dieser Schutz erstreckt sich auf das Löschen von Daten (z. B. Archivierungsobjekte) durch einen Agenten vor Ablauf des Aufbewahrungszeitraums.

Das Schützen des Aufbewahrungszeitraums für Archivierung hilft sicherzustellen, dass keine Person bzw. kein Programm absichtlich oder versehentlich Daten, die unter der Steuerung von IBM Spectrum Protect stehen, löscht. Ein Archivierungsobjekt, das an einen Server mit Aufbewahrungsschutz für Archivierung gesendet wird, ist vor versehentlichem Löschen geschützt und hat eine erzwungene Aufbewahrungsdauer. Beim Aufbewahrungszeitraum für Archivierung gibt es die folgenden Einschränkungen:

- Nur Archivierungsoperationen sind auf dem Server für den Aufbewahrungsschutz zulässig.
- Jedes Objekt, das nicht explizit über einen Wert im Funktionsaufruf **dsmBindMc** oder über Einschluss/Ausschlussanweisungen an eine Verwaltungsklasse gebunden ist, wird an den expliziten Namen der Standardverwaltungsklasse gebunden. Wenn die Standardverwaltungsklasse in der Knotenmaßnahme beispielsweise MC1 lautet, ist das Objekt explizit an MC1 und nicht an die Standardklasse (DEFAULT) gebunden. In einer Abfrageantwort würde angezeigt, dass das Objekt an MC1 gebunden ist.

- Nachdem Sie den Aufbewahrungsschutz für Archivierungsdaten aktiviert haben, führt jeder Versuch, ein Objekt vor dem Ablauf der Aufbewahrungsdauer zu löschen, am Ende der Transaktion zum Code DSM_RC_ABORT_DELETE_NOT_ALLOWED.

Anweisungen zum Definieren des Aufbewahrungsschutzes für ein Archivierungsobjekt finden Sie in der Dokumentation für den IBM Spectrum Protect-Server.

Führen Sie die folgenden Schritte aus, um den Aufbewahrungsschutz für Archivierungsdaten zu konfigurieren:

1. Führen Sie bei einer neuen Serverinstallation ohne alte Daten den Befehl **SET ARCHIVERETENTIONPROTECTION ON** aus.
2. Geben Sie in der API-Optionszeichenfolge im Funktionsaufruf **dsmInit** oder **dsmInitEx** die folgende Anweisung ein:

   ```
   -ENABLEARCHIVERETENTIONPROTECTION=yes
   ```

   Sie können auch die Option enablearchiveretentionprotection in der Datei dsm.opt auf anderen Systemen als UNIX-Systemen oder in der Datei dsm.sys auf UNIX-Systemen setzen:

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

   Weitere Informationen zu dieser Option finden Sie in „Die Option enablearchiveretentionprotection".
3. Senden Sie eine Abfrage an den Server, um sicherzustellen, dass der IBM Spectrum Protect-Server für den Aufbewahrungsschutz für Archivierung aktiviert ist. Prüfen Sie den Wert des Felds archiveRetentionProtection in der Struktur dsmQuerySessInfo.

Die Option enablearchiveretentionprotection

Die Option enablearchiveretentionprotection gibt an, ob der Aufbewahrungsschutz für Daten für Archivierungsobjekte auf dem für diesen Zweck zugeordneten IBM Spectrum Protect-Server aktiviert werden soll. Ihr Serveradministrator muss den Aufbewahrungsschutz für Daten auf einem neuen Server aktivieren, auf dem noch keine Objekte (Sicherungs- oder Archivierungsobjekte oder speicherverwaltete Objekte) gespeichert wurden. Wenn die API-Anwendung versucht, eine Sicherungsversion oder ein speicherverwaltetes Objekt auf dem Server zu speichern, wird eine Fehlernachricht ausgegeben.

Der folgende Hinweis in Kapitel 3, „Hinweise zum Entwerfen einer Anwendung", auf Seite 9: „Speichern Sie keine objectID-Werte zur Verwendung in späteren Zurückschreibungen. Die Persistenz dieser Werte ist während der Lebensdauer des Objekts nicht garantiert." gilt nicht für Archivierungsmanageranwendungen, da der Archivierungsmanagerserver keinen Export oder Import unterstützt. Archivierungsmanageranwendungen können die Objekt-ID (objectID) speichern, um die Leistung beim Zurückschreiben von Objekten zu verbessern.

Wenn der Server den Befehl **SET ARCHIVERETENTIONPROTECTION ON** ausgibt, können Sie ein archiviertes Objekt erst dann mithilfe des Befehls **delete filespace** vom Server löschen, wenn die Maßnahmenparameter der Archivierungskopiengruppe erfüllt sind. Die entsprechende Serverdokumentation enthält Informationen zur Konfiguration einer Verwaltungsklasse.

Maßnahme für ereignisgesteuerte Aufbewahrungsdauer

Bei einer Maßnahme für ereignisgesteuerte Aufbewahrungsdauer beginnt der Aufbewahrungszeitraum eines Archivierungsobjekts mit einem Geschäftsereignis, beispielsweise dem Schließen eines Bankkontos. Bei der ereignisgesteuerten Aufbewahrungsdauer wird die IBM Spectrum Protect-Maßnahme für die Datenaufbewahrung eng mit den Geschäftsanforderungen für Daten abgestimmt. Wenn das Ereignis eintritt, sendet die Anwendung ein Ereignis **eventRetentionActivate** für das betreffende Objekt an den Server, um die Aufbewahrungsdauer einzuleiten.

Führen Sie die folgenden Schritte aus, um eine ereignisgesteuerte Maßnahme für Aufbewahrungsdauer zu verwenden:

1. Erstellen Sie auf dem Server eine Verwaltungsklasse mit einer **Archivierungskopiengruppe** des Typs `EVENT`. Weitere Informationen finden Sie in der Dokumentation für den IBM Spectrum Protect-Server.
2. Fragen Sie die Verwaltungsklasse ab, um sicherzustellen, dass die Klasse ereignisgesteuert ist. Wenn die Verwaltungsklasse ereignisgesteuert ist, enthält das Feld **retainInit** in der Struktur **archDetailCG** den Wert `ARCH_RETINIT_EVENT`.
3. Binden Sie die Objekte über 'include', **archmc** oder explizit über das Attribut **mcNameP** in der Struktur **ObjAttr** im Funktionsaufruf **dsmSendObj** an die ereignisgesteuerte Verwaltungsklasse.
4. Fragen Sie zu dem Zeitpunkt, an dem Sie den Aufbewahrungszeitraum für das Objekt beginnen möchten, den Server nach allen betroffenen Objekten ab. Prüfen Sie, ob sie sich im `Wartestatus` befinden und rufen Sie die Objekt-ID ab. Im Wartestatus enthält das Feld **retentionInitiated** in der Struktur **qryRespArchiveData** den Wert `DSM_ARCH_RETINIT_PENDING`.
5. Setzen Sie zunächst einen Aufruf **dsmBeginTxn** und dann einen Aufruf **dsmRetentionEvent** mit der Liste der Objekte ab, gefolgt von einem Aufruf **dsmEventType**: `eventRetentionActivate`. Überschreitet die Anzahl Objekte den Wert von `maxObjPerTxn`, müssen Sie mehrere Transaktionen verwenden.

   Einschränkung: Sie können pro Transaktion nur einen einzigen Aufruf **dsmRetentionEvent** ausgeben.
6. Fragen Sie die Objekte ab, um sicherzustellen, dass die Aufbewahrungsdauer aktiviert ist. Wenn die Aufbewahrungsdauer aktiviert ist, enthält das Feld **retentionInitiated** in der Struktur **qryRespArchiveData** den Wert `I`.

IBM Spectrum Protect-System abfragen

Die API enthält mehrere Abfragen, beispielsweise eine Verwaltungsklassenabfrage, die von Anwendungen verwendet werden können.

Alle Abfragen, die den Aufruf **dsmBeginQuery** verwenden, beinhalten die folgenden Schritte:

1. Aufruf **dsmBeginQuery** mit dem geeigneten Abfragetyp senden:
   - Sicherung
   - Archivierung
   - Aktive gesicherte Objekte
   - Dateibereich
   - Verwaltungsklasse

   Der Aufruf **dsmBeginQuery** informiert die API über das Datenformat, das vom Server zurückgegeben wird. Die entsprechenden Felder können in die Daten-

strukturen eingefügt werden, die über die **dsmGetNextQObj**-Aufrufe übergeben werden. Der Aufruf zum Beginnen der Abfrage (begin query) ermöglicht dem Anwendungsclient außerdem, den Geltungsbereich der Abfrage festzulegen, indem die Parameter ordnungsgemäß im Aufruf 'begin query' angegeben werden.

Einschränkung: Auf UNIX- und Linux-Systemen kann nur der Rootbenutzer aktive gesicherte Objekte abfragen. Dieser Abfragetyp wird als "Direktaufruf" bezeichnet.

2. Geben Sie den Aufruf **dsmGetNextQObj** ein, um jeden Datensatz aus der Abfrage abzurufen. Dieser Aufruf übergibt einen Puffer, der groß genug ist, um die aus der Abfrage zurückgegebenen Daten aufzunehmen. Jeder Abfragetyp hat eine entsprechende Datenstruktur für die zurückgegebenen Daten. Dem Abfragetyp 'Sicherung' beispielsweise ist die Struktur **qryRespBackupData** zugeordnet, die gefüllt wird, wenn der Aufruf **dsmGetNextQObj** gesendet wird.
3. Der Aufruf **dsmGetNextQObj** gibt in der Regel einen der folgenden Codes zurück:
   - `DSM_RC_MORE_DATA`: Senden Sie den Aufruf **dsmGetNextQObj** erneut.
   - `DSM_RC_FINISHED`: Es sind keine weitere Daten vorhanden. Senden Sie den Aufruf **dsmEndQuery**.
4. Senden Sie den Aufruf **dsmEndQuery**. Wenn alle Abfragedaten abgerufen wurden bzw. keine weiteren Abfragedaten benötigt werden, geben Sie den Aufruf **dsmEndQuery** ein, um den Abfrageprozess zu beenden. Die API löscht alle verbleibenden Daten aus dem Abfragedatenstrom und gibt alle für die Abfrage genutzten Ressourcen frei.

Abb. 9 zeigt das Zustandsdiagramm für Abfrageoperationen.

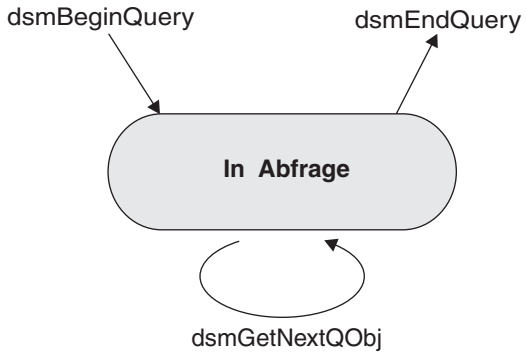


Abbildung 9. Zustandsdiagramm für allgemeine Abfragen

Abb. 10 auf Seite 35 zeigt das Ablaufdiagramm für Abfrageoperationen.

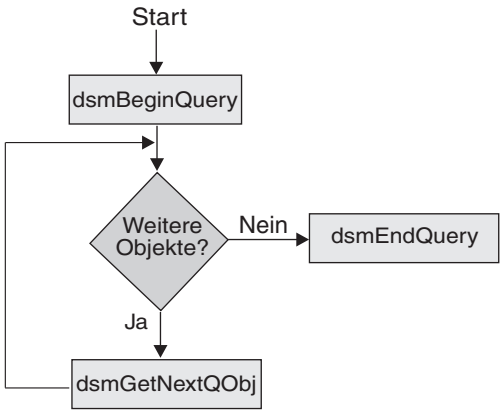


Abbildung 10. Ablaufdiagramm für allgemeine Abfragen

Beispiel für das Abfragen des Systems

In diesem Beispiel für eine Verwaltungsklassenabfrage werden die Werte aller Felder in den Sicherungs- und Archivierungskopiengruppe für eine bestimmte Verwaltungsklasse gedruckt.

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

/* Struktur qMCData mit gewünschten Abfragekriterien füllen      */
qMCData.stVersion = qryMCDataVersion;   /* Strukturversion       */
qMCData.mcName    = mc;                 /* Verwaltungsklassenname*/
qMCData.mcDetail  = bTrue;              /* vollständige Details? */

/* Parameter für Datenblock zum Abrufen/Senden von Daten defin.  */
qData.stVersion = DataBlkVersion;
qData.bufferLen = sizeof(qryRespMCDetailData);
qData.bufferPtr = (char *)&qRespMCData;

qRespMCData.stVersion = qryRespMCDetailDataVersion;
```

```
if ((rc = dsmBeginQuery(dsmHandle,qtMC,(dsmQueryBuff *)&qMCData)))
{
   printf("*** dsmBeginQuery fehlgeschlagen: ");
   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("Verwaltungsklasse
         printf("              Name:
         printf("Sicherungs-CG-Name:
            .
            .  /* andere Felder der Sich.-/Archiv.-Kopiengruppen */
            .
         printf("        Kopienziel:
      }
         else
      {
         done = bTrue;
         if (rc != DSM_RC_FINISHED)
         {
            printf("*** dsmGetNextQObj fehlgeschlagen: ");
            rcApiOut(dsmHandle, rc);
         }
      }
      if (rc == DSM_RC_FINISHED) done = bTrue;
   }
   rc = dsmendQuery (dsmHandle);
}
```

Abbildung 11. Beispiel für die Ausführung einer Systemabfrage

Servereffizienz

Beachten Sie diese Richtlinien beim Abrufen von Objekten vom IBM Spectrum Protect-Server oder beim Senden von Objekten an den Server.

- Wenn Sie Objekte vom IBM Spectrum Protect-Server abrufen, beachten Sie folgende Richtlinien:

- Rufen Sie Daten in der Zurückschreibungsreihenfolge ab, die vom IBM Spectrum Protect-Server bereitgestellt wird. Die Zurückschreibungsreihenfolge ist insbesondere bei Bandeinheiten wichtig, da das Abrufen von Daten, die sich nicht in der korrekten Reihenfolge befinden, das Zurückspulen und Laden von Bändern zur Folge haben kann.
- Selbst wenn die Daten auf einer Platteneinheit gespeichert sind, können Sie Zeit sparen, wenn beim Abrufen die Reihenfolge eingehalten wird.
- Führen Sie so viel Arbeit wie möglich in einer einzigen IBM Spectrum Protect-Serversitzung aus.
- Starten und stoppen Sie keine Mehrfachsitzungen.

• Wenn Sie Objekte an den IBM Spectrum Protect-Server senden, beachten Sie folgende Richtlinien:
  - Senden Sie mehrere Objekte in einer einzigen Transaktion.
  - Vermeiden Sie es, ein einziges Objekt pro Transaktion zu senden; dies gilt insbesondere dann, wenn die Daten direkt an eine Bandeinheit gesendet werden. Im Rahmen der Bandeinheitentransaktion wird sichergestellt, dass die Daten in den Arbeitsspeicherpuffern des Bands auf Datenträger geschrieben werden.

Zugehörige Konzepte:

„Objekte nach Zurückschreibungsreihenfolge auswählen und sortieren" auf Seite 72

Zugehörige Informationen:

„Sitzung starten oder beenden" auf Seite 15

Daten an einen Server senden

Die API ermöglicht Anwendungsclients das Senden von Daten oder benannten Objekten und ihrer zugehörigen Daten in IBM Spectrum Protect-Serverspeicher.

Tipp: Sie können Daten entweder sichern oder archivieren. Führen Sie alle Sendeoperationen innerhalb einer Transaktion aus.

Das Transaktionsmodell

Die während einer Sicherungs- oder Archivierungsoperation an den IBM Spectrum Protect-Speicher gesendeten Daten werden innerhalb einer Transaktion gesendet. Ein Transaktionsmodell stellt ein hohes Maß an Datenintegrität bereit, bedingt jedoch einige Einschränkungen, die ein Anwendungsclient berücksichtigen muss.

Starten Sie eine Transaktion durch einen Aufruf von **dsmBeginTxn** und beenden Sie eine Transaktion durch einen Aufruf von **dsmEndTxn**. Eine einzelne Transaktion ist eine atomare Aktion. Die innerhalb der Grenzen einer Transaktion gesendeten Daten werden entweder am Ende der Transaktion auf dem System festgeschrieben oder rückgängig gemacht, wenn die Transaktion vorzeitig endet.

Transaktionen können aus Sendeoperationen für Einzelobjekte oder aus Sendeoperationen für mehrere Objekte bestehen. Um die Systemleistung durch Verringerung des Systemaufwands zu verbessern, sollten Sie kleinere Objekte in einer Transaktion mit mehreren Objekten senden. Der Anwendungsclient bestimmt, ob einzelne oder mehrere Transaktionen angemessen sind.

Senden Sie alle Objekte in einer Transaktion mit mehreren Objekten an dasselbe Kopienziel. Wenn Sie ein Objekt an ein anderes Ziel als das vorherige Objekt senden müssen, beenden Sie die aktuelle Transaktion und starten Sie eine neue. Innerhalb der neuen Transaktion können Sie das Objekt an das neue Kopienziel senden.

Anmerkung: Objekte, die keine Bitdaten enthalten (*sizeEstimate=0*), werden nicht auf Kopienzielkonsistenz überprüft.

IBM Spectrum Protect begrenzt die Anzahl Objekte, die in einer Transaktion mit mehreren Objekten gesendet werden können. Diesen Grenzwert finden Sie, wenn Sie **dsmQuerySessInfo** aufrufen und das Feld **maxObjPerTxn** überprüfen. Dieses Feld enthält den Wert der auf Ihrem Server definierten Option *TXNGroupmax*.

Der Anwendungsclient muss die innerhalb einer Transaktion gesendeten Objekte überwachen, um im Fall einer vorzeitigen Beendigung der Transaktion eine Wiederholung oder Fehlerbehandlung ausführen zu können. Der Server oder der Client kann eine Transaktion jederzeit stoppen. Der Anwendungsclient muss darauf vorbereitet sein, nicht selbst gestartete plötzliche Transaktionsbeendigungen zu bearbeiten.

Dateizusammenfassung

IBM Spectrum Protect-Server verwenden eine Funktion, die als Dateizusammenfassung bezeichnet wird. Bei der Dateizusammenfassung werden alle Objekte, die in einer einzigen Transaktion gesendet werden, zusammen gespeichert; dadurch wird Speicherbereich eingespart und die Leistung verbessert. Sie können die Objekte weiterhin einzeln abfragen und zurückschreiben.

Um diese Funktion verwenden zu können, müssen alle Objekte in einer Transaktion denselben Dateibereichsnamen haben. Wenn sich der Dateibereichsname in einer Transaktion ändert, schließt der Server das vorhandene zusammengefasste Objekt und beginnt ein neues.

LAN-unabhängige Datenübertragung

Die API kann die Vorteile der LAN-unabhängigen Datenübertragung nutzen, wenn die Option **dsmSetUp** für Multithreading auf ON gesetzt ist. Die API gibt das Vorhandensein eines LAN-unabhängigen Ziels in der **Query Mgmt Class**-Antwortstruktur **archDetailCG** oder **backupDetailCG** im Feld **bLanFreeDest** zurück.

Sie können LAN-unabhängige Operationen auf Plattformen verwenden, die vom Speicheragenten unterstützt werden. Die Macintosh-Plattform ist ausgeschlossen.

LAN-unabhängige Informationen werden in den folgenden Ausgabestrukturen bereitgestellt. Die Ausgabestruktur (**dsmEndGetDataExOut_t**) für **dsmEndGetData** umfasst das Feld **totalLFBytesRecv**. Dies ist die Gesamtanzahl empfangener LAN-unabhängiger Byte. Die Ausgabestruktur (**dsmEndSendObjExOut_t**) für **dsmEndSendObjEx** umfasst das Feld **totalLFBytesSent**. Dies ist die Gesamtanzahl gesendeter LAN-unabhängiger Byte.

Zugehörige Informationen:

LAN-unabhängige Datenversetzung: Übersicht über den Speicheragenten

Gleichzeitiges Schreiben

IBM Spectrum Protect-Serverspeicherpools können so konfiguriert werden, dass sie während einer Sicherung oder Archivierung gleichzeitig in einen primären Speicherpool und in einen oder mehrere Kopienspeicherpools schreiben. Verwenden Sie diese Konfiguration zum Erstellen mehrerer Kopien des Objekts.

Wenn eine Operation für gleichzeitiges Schreiben fehlschlägt, kann der Rückkehrcode für die Funktion **dsmEndTxn** DSM_RC_ABORT_STGPOOL_COPY_CONT_NO lauten; dies gibt an, dass der Schreibvorgang in einen der Kopienspeicherpools fehlgeschlagen ist und die Option COPYCONTINUE für den IBM Spectrum Protect-Speicherpool auf NO gesetzt wurde. Die Anwendung wird beendet und der Fehler muss vom IBM Spectrum Protect-Serveradministrator behoben werden.

Weitere Informationen zum Konfigurieren von Operationen für gleichzeitiges Schreiben finden Sie in der Dokumentation für den IBM Spectrum Protect-Server.

API-Leistung verbessern

Sie können die Leistung der API mithilfe der Clientoptionen tcpbuffsize und tcpnodelay sowie dem API-Parameter **DataBlk** verbessern.

Tabelle 10 enthält eine Beschreibung der Aktionen, die Sie zum Verbessern der API-Leistung durchführen können.

Tabelle 10. Optionen zum Sichern/Archivieren und der API-Parameter zum Verbessern der Leistung

Optionen des Clients für Sichern/ Archivieren	**Beschreibung**
tcpbuffsize	Gibt die Größe des TCP-Puffers an. Der Standardwert ist 31 KB. Setzen Sie zur Verbesserung der Leistung den Wert auf 32 KB.
tcpnodelay	Sie gibt an, ob kleine Puffer statt angehalten zu werden an den Server gesendet werden sollten. Setzen Sie diese Option für alle Plattformen auf *yes*. Diese Option ist nur für Windows und AIX gültig.
API-Parameter	**Beschreibung**
DataBlk	Dieser Parameter wird im Funktionsaufruf **dsmSendData** verwendet, um die Größe des Anwendungspuffers zu bestimmen. Die besten Ergebnisse können Sie erzielen, wenn Sie für den Parameter ein Vielfaches des Wertes von *tcpbuffsize* verwenden und dann den Wert von tcpbuffsize minus 4 Byte angeben. Geben Sie beispielsweise für diesen Parameter den Wert 28 an, wenn tcpbuffsize einen Wert von 32 KB hat.

Jeder Aufruf **dsmSendData** erfolgt synchron und kehrt erst zurück, nachdem für die in **dataBlkPtr** an die API übertragenen Daten eine Flushoperation in das Netz ausgeführt wurde. Die API fügt jedem Transaktionspuffer, der in das Netz gestellt wird, 4 Byte Systemaufwand hinzu.

Hat beispielsweise der Transaktionspuffer eine Größe von 32 KB und der **DataBlk**-Anwendungspuffer eine Größe von 31 KB, passt jeder **DataBlk**-Anwendungspuffer in einen Kommunikationspuffer und es kann sofort eine Flushoperation ausgeführt werden. Wenn der **DataBlk**-Anwendungspuffer jedoch exakt 32 KB hat, sind, da die API pro Transaktionspuffer 4 Byte hinzufügt, zwei Flushoperationen erforderlich: eine mit 32 KB und eine mit 4 Byte. Wird die Option tcpnodelay auf no gesetzt, kann die Flushoperation für die 4 Byte bis zu 200 Millisekunden dauern.

API für das Senden von Leistungsdaten an die Clientleistungsüberwachung konfigurieren

Die Clientleistungsüberwachung ist eine Komponente des Tivoli Storage Manager Administration Center, die zum Anzeigen von Leistungsdaten verwendet wird, die von der API erfasst wurden. Die Clientleistungsüberwachung zeichnet Leistungsdaten für Clientsicherungs-, -archivierungs- und -zurückschreibungsoperationen auf und zeigt diese Daten an.

Bei aktivierter Leistungsüberwachung können Sie Leistungsdaten anzeigen, die von der API unter Verwendung der Leistungsüberwachung erfasst werden. Die Leistungsüberwachung ist im Tivoli Storage Manager Administration Center verfügbar. Ab Version 7.1 ist die Komponente Administration Center nicht mehr in Tivoli Storage Manager oder in den IBM Spectrum Protect-Verteilungen enthalten. Wenn Sie ein Administration Center haben, das mit einem vorherigen Serverrelease installiert wurde, können Sie es weiterhin zum Anzeigen von Leistungsdaten verwenden. Wenn Sie über kein installiertes Administration Center verfügen, können Sie die zuvor freigegebene Version von ftp://public.dhe.ibm.com/storage/tivoli-storage-management/maintenance/admincenter/v6r3/ herunterladen. Informationen zur Verwendung der Leistungsüberwachung finden Sie in der Dokumentation zu Tivoli Storage Manager Version 6.3.

Optionen für die Clientleistungsüberwachung konfigurieren

Sie aktivieren IBM Spectrum Protect-Clients für die Verwendung der Leistungsüberwachung, indem Sie Parameter in der Clientoptionsdatei angeben. Sie geben diese Optionen für jeden Client an, der überwacht werden soll.

Setzen Sie bei der Überwachung der Leistung auf UNIX- und Linux-Computern den Grenzwert des Deskriptors für offene Dateien auf mindestens 1024, indem Sie den folgenden Befehl verwenden:

```
ulimit -n 1024
```

Führen Sie die folgenden Schritte aus, um die Optionen für die Clientleistungsüberwachung zu konfigurieren:

1. Öffnen Sie die Clientoptionsdatei für jeden überwachten Client. Abhängig von Ihrer Konfiguration befinden sich die Clientoptionen in einer der folgenden Dateien:
   - `dsm.opt`
   - `dsm.sys`
2. Fügen Sie der Clientoptionsdatei die folgenden Optionen hinzu:

   ```
   PERFMONTCPSERVERADDRESS
   PERFMONTCPPORT
   PERFMONCOMMTIMEOUT
   ```

PERFMONTCPSERVERADDRESS

Die Option PERFMONTCPSERVERADDRESS gibt den Hostnamen oder die IP-Adresse des Systems an, auf dem die Clientleistungsüberwachung installiert ist.

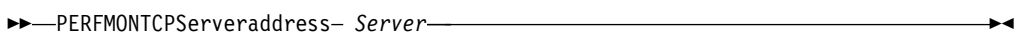
Unterstützte Clients

Diese Option ist plattformunabhängig und wird für alle Clients unterstützt.

Optionsdatei

Definieren Sie diese Option in der Clientoptionsdatei (`dsm.opt` oder `dsm.sys`).

Syntax



Parameter

Server
Der Server-Hostname oder die IP-Adresse des Systems, auf dem die Clientleistungsüberwachung installiert ist (dies ist der Server, auf dem auch das Administration Center ausgeführt wird).

Beispiele

Optionsdatei:

```
PERFMONTCPSERVERADDRESS 131.222.10.5
```

Befehlszeile:
Diese Option kann mit der Befehlszeile nicht definiert werden.

PERFMONTCPPORT

Die Anschlussnummer, an der die Clientleistungsüberwachung für Leistungsdaten von den Clients empfangsbereit ist.

Unterstützte Clients

Diese Option ist plattformunabhängig und wird für alle Clients unterstützt.

Optionsdatei

Definieren Sie diese Option in der Clientoptionsdatei (`dsm.opt` oder `dsm.sys`).

Syntax

Parameter

Anschluss
Der Anschluss, der für Clientleistungsdaten überwacht wird. Anschluss 5129 ist der Standardanschluss.

Beispiele

Optionsdatei:

```
PERFMONTCPPPORT 5000
```

Befehlszeile:

Diese Option kann mit der Befehlszeile nicht definiert werden.

PERFMONCOMMTIMEOUT

Gibt die maximale Zeit in Sekunden an, die der Aufruf dsmTerminate auf das Empfangen von Leistungsdaten wartet, bevor eine Sitzung beendet wird.

Unterstützte Clients

Diese Option ist plattformunabhängig und wird für alle Clients unterstützt.

Optionsdatei

Definieren Sie diese Option in der Clientoptionsdatei (`dsm.opt` oder `dsm.sys`).

Syntax

Parameter

Sekunden

Die Zeit, die auf das Empfangen von verbleibenden Leistungsdaten gewartet wird, bevor die Sitzung beendet wird.

Beispiele

Optionsdatei:

```
PERFMONCOMMTIMEOUT 60
```

Befehlszeile:

Diese Option kann mit der Befehlszeile nicht definiert werden.

Objekte an den Server senden

Anwendungsclients können Daten oder benannte Objekte und ihre zugehörigen Daten mithilfe der API-Sicherungs- und -Archivierungsfunktionen in IBM Spectrum Protect-Speicher senden. Die Sicherungs- und Archivierungskomponenten des Systems ermöglichen die Verwendung unterschiedlicher Verwaltungsprozeduren für Daten, die in den Speicher gesendet werden.

Das Attribut für die Größenschätzung ist eine Schätzung der Gesamtgröße des Datenobjekts, das an den Server gesendet werden soll. Wenn der Anwendung die exakte Objektgröße nicht bekannt ist, geben Sie für *sizeEstimate* einen höheren Schätzwert an. Wenn die Schätzung kleiner als die tatsächliche Größe ist, nutzt der IBM Spectrum Protect-Server zusätzliche Ressourcen für die Handhabung zusätzlicher Bereichszuordnungen.

Hinweise:

- Diese Größe sollte so genau wie möglich geschätzt werden. Der Server verwendet dieses Attribut für die effiziente Bereichszuordnung und Objektpositionierung in seinen Speicherressourcen.
- Wenn die Schätzung kleiner als die tatsächliche Größe ist, ordnet ein Server, der Caching verwendet, keinen zusätzlichen Speicherbereich zu und stoppt die Sendeoperation.

Möglicherweise treten Probleme auf, wenn der Wert für *sizeEstimate* viel zu groß ist. Unter Umständen ist auf dem Server nicht genügend Speicherbereich für die geschätzte Größe, aber für die tatsächliche Größe verfügbar. Es ist auch möglich, dass der Server langsamere Einheiten verwendet.

Sie können Objekte, deren Größe 2 Gigabyte überschreitet, sichern oder archivieren. Die Objekte können komprimiert oder unkomprimiert sein.

Um eine Sendeoperation zu starten, rufen Sie **dsmSendObj** auf. Sind mehr Daten verfügbar, als in einer einzigen Operation gesendet werden können, können Sie **dsmSendData** wiederholt aufrufen, um die verbleibenden Informationen zu übertragen. Rufen Sie **dsmEndSendObj** auf, um die Sendeoperation abzuschließen.

Erläuterungen zu Sicherungs- und Archivierungsobjekten

Die Sicherungskomponente des IBM Spectrum Protect-Systems unterstützt mehrere Versionen benannter Objekte, die auf dem Server gespeichert sind.

Jedes Objekt, das auf dem Server gesichert wird und dessen Name mit dem Namen eines Objekts übereinstimmt, das bereits von diesem Client auf dem Server gespeichert ist, unterliegt der Versionssteuerung. Bei Objekten wird von einem aktiven oder inaktiven Status auf dem Server ausgegangen. Die neueste Kopie eines Objekts auf dem Server, die nicht inaktiviert wurde, ist im aktiven Status. Jedes andere Objekt mit demselben Namen wird als inaktiv betrachtet, unabhängig davon, ob es sich um eine ältere Version oder um eine inaktivierte Kopie handelt. Verwaltungsklassenkonstrukte definieren verschiedene Verwaltungskriterien. Sie werden aktiven und inaktiven Objekten auf dem Server zugeordnet.

Tabelle 11 enthält die für den aktiven und den inaktiven Status gültigen Kopiengruppenfelder.

Tabelle 11. Felder für Sicherungskopiengruppen

Feld	Beschreibung
VEREXISTS	Die Anzahl inaktiver Versionen, wenn aktive Versionen vorhanden sind.
VERDELETED	Die Anzahl inaktiver Versionen, wenn keine aktiven Versionen vorhanden sind.
RETEXTRA	Die Aufbewahrungsdauer für inaktive Versionen in Tagen.
RETONLY	Die Aufbewahrungsdauer für die letzten inaktiven Versionen in Tagen, wenn keine aktiven Versionen vorhanden sind.

Wenn jede Sicherungsversion einen eindeutigen Namen hat, z. B. eine Zeitmarke im Namen, findet die Versionssteuerung nicht automatisch statt: jedes Objekt ist aktiv. Aktive Objekte verfallen nie, daher wäre es die Aufgabe einer Anwendung, diese mit dem Aufruf **dsmDeleteObj** zu inaktivieren. In dieser Situation wäre es für

die Anwendung erforderlich, dass die inaktivierten Objekte so bald wie möglich verfallen. Der Benutzer würde eine Sicherungskopiengruppe mit VERDELETED=0 und RETONLY=0 definieren.

Die Archivierungskomponente des IBM Spectrum Protect-Systems erlaubt die Speicherung von Objekten auf dem Server mit Steuerangaben für den Aufbewahrungs- oder Verfallszeitraum anstelle der Versionssteuerung. Jedes gespeicherte Objekt ist eindeutig, auch wenn sein Name mit einem bereits archivierten Objekt übereinstimmt. Bei Archivierungsobjekten ist den Metadaten ein Beschreibungsfeld zugeordnet, mit dem während einer Abfrage ein bestimmtes Objekt identifiziert werden kann.

Jedem Objekt auf dem IBM Spectrum Protect-Server wird eine eindeutige Objekt-ID zugewiesen. Die Persistenz des ursprünglichen Werts wird während der Lebensdauer eines Objekts nicht garantiert (insbesondere nach einem Export oder Import). Daher sollte eine Anwendung die ursprüngliche Objekt-ID zur Verwendung in zukünftigen Zurückschreibungen nicht abfragen und speichern. Stattdessen sollte eine Anwendung den Objektnamen und das Einfügedatum speichern. Sie können mit diesen Informationen während einer Zurückschreibung Objekte abfragen und das Einfügedatum verifizieren. Dann kann das Objekt mit der aktuellen Objekt-ID zurückgeschrieben werden.

Komprimierung

Konfigurationsoptionen auf einem bestimmten Knoten legen in Verbindung mit der **dsmSendObj**-Option `objCompressed` fest, ob IBM Spectrum Protect das Objekt während eines Sendevorgangs komprimiert. Außerdem werden Objekte mit einer Größenschätzung (`sizeEstimate`), die `DSM_MIN_COMPRESS_SIZE` unterschreitet, nie komprimiert.

Wenn das Objekt bereits komprimiert ist (`objCompressed=bTrue`), wird es nicht noch einmal komprimiert. Wenn es nicht komprimiert ist, trifft IBM Spectrum Protect die Entscheidung, ob das Objekt komprimiert werden soll, anhand der Werte der Komprimierungsoption, die vom Administrator und in den API-Konfigurationsquellen definiert ist.

Der Administrator kann die Komprimierungsschwellenwerte auf dem Server mit dem Befehl **register node** ändern (`compression=yes`, `no` oder `Vom Client bestimmt`). Bei Angabe von `Vom Client bestimmt` wird das Komprimierungsverhalten durch den Wert der Komprimierungsoption in den Konfigurationsquellen bestimmt.

Bei einigen Datentypen, z. B. bereits komprimierte Daten, kann die Datengröße sogar zunehmen, wenn sie mit dem Komprimierungsalgorithmus verarbeitet werden. In diesem Fall wird der Rückkehrcode DSM_RC_COMPRESS_GREW generiert. Wenn Sie feststellen, dass dieser Fall eintreten könnte, Sie die Sendeoperation aber dennoch fortsetzen wollen, teilen Sie den Endbenutzern mit, folgende Option in ihrer Optionsdatei anzugeben:

```
COMPRESSAlways Yes
```

Wenn Sie während der Ausführung der Funktion **dsmSendData** bei aktivierter Komprimierung den Rückkehrcode DSM_RC_COMPRESS_GREW empfangen, möchten Sie eventuell von vorn beginnen und das Objekt ohne Komprimierung erneut senden. Um dies durchzusetzen, setzen Sie `ObjAttr.objCompressed` in **dsmSendObj** auf `bTrue`.

Informationen zum tatsächlichen Komprimierungsverhalten während einer **dsmSendObj**-Operation werden durch den Aufruf **dsmEndSendObjEx** zurückgegeben. `objCompressed` gibt an, ob die Komprimierung durchgeführt wurde. `totalBytesSent` ist die Anzahl der von der Anwendung gesendeten Byte. `totalCompressedSize` ist die Anzahl Byte nach der Komprimierung. Der Aufruf **dsmEndSendObjEx** verfügt außerdem über ein Feld `totalLFBytesSent`, das die Gesamtanzahl der LAN-unabhängig gesendeten Byte enthält.

Achtung: Soll Ihre Anwendung eine Zurückschreibung oder einen Abruf von Teilobjekten ausführen, können die Daten während des Sendevorgangs nicht komprimiert werden. Um dies durchzusetzen, setzen Sie `ObjAttr.objCompressed` in **dsmSendObj** auf `bTrue`.

Komprimierungstyp

Der vom Client verwendete Komprimierungstyp wird durch die während der Sicherungs- oder Archivierungsverarbeitung verwendete Kombination von Komprimierung und clientseitiger Datendeduplizierung bestimmt.

Der vom Client verwendete Komprimierungsalgorithmus wird von der API in einem neuen Feld in den Strukturen **qryRespArchiveData** und **qryRespBackupData** angezeigt:

```
dsChar_t        compressAlg[20];  /* Name des Komprimierungsalgorithmus */
```

Die folgenden Komprimierungstypen werden angezeigt:

LZ4 Eine schnellere und effizientere Komprimierungsmethode, die der Client verwendet, wenn ein vom Client dedupliziertes Objekt an einen LZ4-kompatiblen Containerspeicherpool auf dem IBM Spectrum Protect-Server gesendet wird. Der Server muss Version 7.1.5 oder eine höhere Version aufweisen und Containerspeicherpools verwenden. Die clientseitige LZ4-Komprimierung wird nur verwendet, wenn die clientseitige Datendeduplizierung aktiviert ist.

LZW Ein traditioneller Komprimierungstyp, den der Client in den folgenden Situationen verwendet:

- Vom Client dedupizierte Objekte werden an traditionelle Speicherpools (keine Containerspeicherpools) auf dem Server gesendet.
- Das Clientobjekt wird nicht der clientseitigen Datendeduplizierung unterzogen.
- Das Clientobjekt wird nur der traditionellen serverseitigen Datendeduplizierung unterzogen.

Leeres Feld
Das Objekt wird vom Client nicht komprimiert. Das Objekt wird nicht komprimiert, da die Option `compression` auf *no* gesetzt oder die Option während der Sicherungs- oder Archivierungsverarbeitung nicht angegeben wurde. Auch wenn das Objekt vom Client nicht komprimiert wird, könnte es vom Server komprimiert werden.

Der Komprimierungstyp ist nicht konfigurierbar. Er wird vom Client für Sichern/Archivieren während der Sicherungs- oder Archivierungsverarbeitung bestimmt.

Beispiel

Das folgende Beispiel zeigt das Feld `Compression Type` (Komprimierungstyp) in der Ausgabe der Sicherungs- und Archivierungsabfragen von der 64-Bit-Beispielanwendung **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
```

Pufferkopieneliminierung

Mit der Pufferkopieneliminierungsfunktion wird die Kopie von Datenpuffern zwischen einer Anwendung und dem IBM Spectrum Protect-Server entfernt, was eine bessere Prozessorauslastung zur Folge hat. Den größten Effekt hat dieses Konzept in einer LAN-unabhängigen Umgebung.

Die Puffer für die Datenversetzung werden von IBM Spectrum Protect zugeordnet und ein Verweis wird an die Anwendung zurückgegeben. Die Anwendung stellt die Daten in den bereitgestellten Puffer und dieser Puffer wird über die Übertragungsschichten unter Verwendung gemeinsam genutzten Speichers an den Speicheragenten übergeben. Die Daten werden dann auf die Bandeinheit versetzt, wodurch Datenkopien eliminiert werden. Diese Funktion kann mit Sicherungs- oder Archivierungsoperationen verwendet werden.

Achtung: Wenn Sie diese Methode verwenden, müssen Sie besonderes Augenmerk auf die korrekte Pufferhandhabung und die Größe der Puffer legen. Die Puffer werden gemeinsam von den Komponenten genutzt und jede Speicherüberschreibung aufgrund eines Programmierfehlers hat schwerwiegende Fehler zur Folge.

Die gesamte Aufruffolge für die Sicherung/Archivierung ist wie folgt:

```
dsmInitEx (UseTsmBuffers = True, numTsmBuffers
= [wie viele von IBM Spectrum Protect zugeordnete Puffer die Anwendung zuordnen muss])
dsmBeginTxn
für jedes Objekt in der Transaktion
  dsmBindMC
    dsmSendObject
```

```
      dsmRequestBuffer
      dsmSendBufferData (sendet und gibt den verwendeten Puffer frei)
    dsmEndSendObjEx
dsmEndTxn
für jeden noch angehaltenen Puffer
      dsmReleaseBuffer
dsmTerminate
```

Die Funktion **dsmRequestBuffer** kann mehrfach aufgerufen werden bis zu der Höhe des Wertes, der durch die Option numTsmBuffers angegeben wird. Die Anwendung kann über zwei Threads verfügen: einen Produzententhread, der Puffer mit Daten füllt, und einen Konsumententhread, der diese Puffer mit dem Aufruf **dsmSendBufferData** an IBM Spectrum Protect sendet. Wenn ein Aufruf **dsmRequestBuffer** ausgegeben und dadurch der in **numTsmBuffers** angegebene Wert erreicht wird, wird der Aufruf **dsmRequestBuffer** so lange blockiert, bis ein Puffer freigegeben wird. Die Freigabe des Puffers kann erfolgen durch Aufrufen von **dsmSendBufferData**, wodurch ein Puffer gesendet und freigegeben wird, oder durch Aufrufen von **dsmReleaseBuffer**. Weitere Informationen finden Sie unter callbuff.c im API-Musterverzeichnis.

Tritt an irgendeinem Punkt beim Senden ein Fehler auf, muss die Anwendung alle angehaltenen Puffer freigeben und die Sitzung beenden. Beispiel:

```
Bei Fehler
  für jeden von der Anwendung angehaltenen Datenpuffer
    dsmReleaseBuffer aufrufen
dsmTerminate
```

Wenn **dsmTerminate** von einer Anwendung aufgerufen wird und noch ein Puffer angehalten ist, wird die API nicht verlassen. Der folgende Code wird zurückgegeben: DSM_RC_CANNOT_EXIT_MUST_RELEASE_BUFFER. Wenn die Anwendung den Puffer nicht freigeben kann, muss sie den Prozess verlassen, um eine Bereinigung zu erzwingen.

Pufferkopieneliminierung und Zurückschreibung und Abruf

Der IBM Spectrum Protect-Server steuert das in den Puffer zu übertragende Datenvolumen, basierend auf der Bandzugriffsoptimierung beim Zurückschreiben und Abrufen. Diese Methode ist weniger vorteilhaft für die Anwendung als die normale Methode zum Abrufen von Daten. Prüfen Sie bei der Prototyperstellung die Leistung der Pufferkopieneliminierungsmethode und verwenden Sie diese Methode nur, wenn Sie eine nennenswerte Verbesserung erkennen.

Das maximale Datenvolumen in einem einzelnen vom IBM Spectrum Protect-Server zurückgegebenen Puffer beträgt (256K Byte – Headeraufwand). Folglich profitieren von diesem Datenabrufmechanismus nur Anwendungen, die kleine Pufferschreiboperationen bearbeiten. Die Anwendung muss die Anzahl Byte im Puffer genau beachten, abhängig von der Objektgröße, vom Netz und von anderen Bedingungen. In einigen Situationen kann die Verwendung der Pufferkopieneliminierung eine schlechtere Leistung als bei der normalen Zurückschreibung bewirken. Die API speichert die Daten normalerweise zwischen und gibt eine feste Länge an die Anwendung zurück. Die Anwendung kann dann die Anzahl der Datenschreiboperationen zurück auf die Platte steuern.

Wenn Sie die Pufferkopieneliminierung verwenden, erstellen Sie einen Daten-Caching-Mechanismus für Puffer, die kleiner sind als die bevorzugte Schreibpuffergröße. Wenn eine Anwendung beispielsweise 64-KB-Datenblöcke auf Platte schreibt, muss die Anwendung die folgenden Schritte ausführen:

1. **dsmGetBufferData** aufrufen.

2. Blöcke von 64 KB auslagern.
3. Beim letzten Block den Rest in einen temporären Puffer (**tempBuff**) kopieren, einen weiteren Aufruf **dsmGetBufferData** ausgeben und den temporären Puffer (**tempBuff**) mit dem Rest der Daten füllen.
4. Das Schreiben von 64-KB-Blöcken fortsetzen:

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

In diesem Beispiel gibt es sechs direkte und eine zwischengespeicherte Plattenschreiboperation.

Die gesamte Aufruffolge für das Zurückschreiben und Abrufen ist wie folgt:

dsmInitEx (`UseTsmBuffers = True numTsmBuffers` = Anzahl der Puffer, die die Anwendung zuordnen will).

```
dsmBeginGetData
While obj id
  dsmGetObj (keine Daten in diesem Aufruf wiederhergestellt- Puffer auf NULL gesetzt)
 While data to read
   dsmGetBufferData (gibt die Daten im Datenpuffer zurück)
  ...process data...
  dsmReleaseBuffer
dsmEndGetObj
dsmEndGetData
```

Implementieren Sie für jeden Aufruf **dsmGetBufferData** einen Aufruf **dsmReleaseBuffer**. Der Aufruf **dsmGetBufferData** und der entsprechende Aufruf **dsmReleaseBuffer** müssen nicht aufeinanderfolgen. Eine Anwendung könnte mehrere Aufrufe **dsmGetBufferData** ausgeben, zunächst, um mehrere Puffer abzurufen, und dann später die entsprechenden Aufrufe **dsmReleaseBuffer** ausgeben. Mustercode für diese Funktion finden Sie unter `callbuff.c` im API-Musterverzeichnis.

Einschränkung: Da die API den Puffer bereitstellt und das Ziel eine Minimierung der Prozessorauslastung ist, ist eine weitere Verarbeitung der Daten im Puffer nicht zulässig. Die Anwendung kann Verschlüsselung und Komprimierung in Verbindung mit der Pufferkopieneliminierung nicht verwenden, da diese beiden Operationen Datenverarbeitung und Kopien erfordern.

Implementieren Sie sowohl den regulären Datenversetzungspfad als auch die Pufferkopieneliminierung, damit der Benutzer in der Lage ist, je nach Bedarf zwischen beiden Pfaden zu wechseln. Wenn der Benutzer Daten komprimieren oder verschlüsseln muss, verwenden Sie den vorhandenen Mechanismus. Besteht eine Einschränkung bezüglich des Prozessors, verwenden Sie den neuen Mechanismus. Diese beiden Mechanismen sind komplementär und ersetzen sich gegenseitig nicht vollständig.

API-Verschlüsselung

Es stehen zwei Verfahren zum Verschlüsseln von Daten zur Verfügung: die anwendungsverwaltete Verschlüsselung und die IBM Spectrum Protect-Clientverschlüsselung.

Wählen und verwenden Sie nur eines dieser Verfahren zum Verschlüsseln von Daten. Die Verfahren schließen sich gegenseitig aus. Wenn Sie Daten mit beiden Verfahren verschlüsseln, können Sie einige Daten nicht zurückschreiben oder abrufen. Beispiel: Angenommen, eine Anwendung verwendet die anwendungsverwaltete Verschlüsselung zum Verschlüsseln von Objekt A und verwendet anschließend die IBM Spectrum Protect-Clientverschlüsselung zum Verschlüsseln von Objekt B. Wenn die Anwendung für eine Zurückschreibungsoperation festlegt, dass die IBM Spectrum Protect-Clientverschlüsselung verwendet werden soll, und versucht, beide Objekte zurückzuschreiben, kann nur Objekt B zurückgeschrieben werden. Objekt A kann nicht zurückgeschrieben werden, weil es von der Anwendung und nicht vom Client verschlüsselt wurde.

Unabhängig vom verwendeten Verschlüsselungsverfahren muss IBM Spectrum Protect die Kennwortauthentifizierung aktivieren. Standardmäßig verwendet der Server `SET AUTHENTICATION ON`.

Die API verwendet entweder die 128-Bit-AES-Verschlüsselung oder die 256-Bit-AES-Verschlüsselung. Die 256-Bit-AES-Datenverschlüsselung stellt eine höhere Datenverschlüsselungsebene als die 128-Bit-AES-Datenverschlüsselung bereit. Mit 256-Bit-AES-Verschlüsselung gesicherte Dateien können nicht mit einem Client, der eine ältere Version aufweist, zurückgeschrieben werden. Die Verschlüsselung kann mit oder ohne Komprimierung aktiviert werden. Wenn Sie mit Verschlüsselung arbeiten, können Sie nicht die Zurückschreibung von Teilobjekten und keine Abruf- und Pufferkopieneliminierungsfunktionen verwenden.

Anwendungsverwaltete Verschlüsselung

Bei der anwendungsverwalteten Verschlüsselung stellt die Anwendung der API das Schlüsselkennwort (unter Verwendung des Schlüssels DSM_ENCRYPT_USER) zur Verfügung und die Anwendung ist für die Verwaltung des Schlüsselkennworts zuständig.

Achtung: Wenn der Verschlüsselungsschlüssel nicht gespeichert wird und Sie den Schlüssel vergessen, können Ihre Daten nicht wiederhergestellt werden.

Die Anwendung stellt das Schlüsselkennwort im Aufruf **dsmInitEx** bereit und muss das korrekte Schlüsselkennwort zum Zeitpunkt der Zurückschreibung bereitstellen.

Achtung: Wenn das Schlüsselkennwort verloren geht, gibt es keine Möglichkeit, die Daten zurückzuschreiben.

Für Sicherungs- und Zurückschreibungsoperationen (oder Archivierungs- und Abrufoperationen) für dasselbe Objekt muss dasselbe Schlüsselkennwort verwendet werden. Dieses Verfahren ist nicht von der Version des IBM Spectrum Protect-Servers abhängig. Zum Konfigurieren dieses Verfahrens müssen in der Anwendung folgende Schritte ausgeführt werden:

1. Die Variable `bEncryptKeyEnabled` muss im Aufruf **dsmInitEx** auf `bTrue` gesetzt werden und die Variable `encryptionPasswordP` muss auf eine Zeichenfolge mit dem Schlüsselkennwort für die Verschlüsselung verweisen.
2. Die Option `include.encrypt` muss für die Objekte auf `encrypt` gesetzt werden. Um beispielsweise alle Daten zu verschlüsseln, definieren Sie Folgendes:

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

und

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

Geben Sie Folgendes an, um das Objekt /FS1/DB2/FULL zu verschlüsseln:

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

3. Unter Windows muss in der Optionszeichenfolge, die im Aufruf **dsmInitEx** an die API übergeben wird, ENCRYPTKEY=PROMPT|SAVE festgelegt werden. Diese Option kann auch in dsm.opt (Windows) oder dsm.sys (UNIX oder Linux) angegeben werden.

Standardmäßig wird die Option encryptkey auf prompt gesetzt. Diese Einstellung stellt sicher, dass der Schlüssel nicht automatisch gespeichert wird. Wird für encryptkey 'save' angegeben, wird der Schlüssel von IBM Spectrum Protect auf der lokalen Maschine gespeichert; in diesem Fall ist jedoch nur ein einziger Schlüssel für alle IBM Spectrum Protect-Operationen mit demselben Knotennamen gültig.

Nachdem ein Objekt gesendet wurde, gibt **dsmEndSendObjEx** an, ob ein Objekt verschlüsselt wurde und welches Verfahren verwendet wurde. Gültige Werte im Feld *encryptionType*:

- DSM_ENCRYPT_NO
- DSM_ENCRYPT_USER
- DSM_ENCRYPT_CLIENTENCRKEY

In der folgenden Tabelle sind die API-Verschlüsselungstypen, Voraussetzungen und verfügbaren Funktionen aufgelistet.

Tabelle 12. API-Verschlüsselungstypen, Voraussetzungen und verfügbare Funktionen.

Typ	Voraussetzungen	Verfügbare Funktion
ENCRYPTIONTYPE	Keine	Festlegen von ENCRYPTIONTYPE in der Optionszeichenfolge, die unter Windows im Aufruf **dsmInitEx** an die API übergeben wird. Der Standardwert ist ENCRYPTIONTYPE=AES128.
EncryptKey=save	Keine	API und Sicherung/Archivierung
EncryptKey=prompt	Keine	API und Sicherung/Archivierung
EncryptKey=generate	Keine	API und Sicherung/Archivierung
EnableClientEncryptKey	Keine	Nur API

Anmerkung: Es wird empfohlen, auf dem Server die Authentifizierung auf ON zu setzen. Wird die Authentifizierung auf OFF gesetzt, wird der Schlüssel nicht verschlüsselt, während die Daten verschlüsselt werden. Diese Vorgehensweise wird jedoch nicht empfohlen.

Tabelle 13 auf Seite 51 zeigt, wie sowohl berechtigte Benutzer als auch nicht berechtigte Benutzer Daten während einer Sicherungs- oder Zurückschreibungsoperation abhängig von dem für die Option passwordaccess angegebenen Wert verschlüsseln oder entschlüsseln können. Die Datei TSM.PWD muss vorhanden sein, damit die folgenden Operationen von berechtigten und nicht berechtigten Benutzern ausgeführt werden können. Der berechtigte Benutzer erstellt die Datei TSM.PWD und setzt die Option encryptkey auf 'save' und die Option passwordaccess auf 'generate'.

Tabelle 13. Verschlüsselung oder Entschlüsselung von Daten mit einem anwendungsverwalteten Schlüssel unter UNIX oder Linux.

Operation	Option passwordaccess	Option encryptkey	Ergebnis
Sicherung durch berechtigten Benutzer	generate	save	Daten verschlüsselt.
	generate	prompt	Daten verschlüsselt, wenn encryptionPasswordP ein Verschlüsselungskennwort enthält.
	prompt	save	Daten verschlüsselt, wenn encryptionPasswordP ein Verschlüsselungskennwort enthält.
	prompt	prompt	Daten verschlüsselt, wenn encryptionPasswordP ein Verschlüsselungskennwort enthält.
Zurückschreibung durch berechtigten Benutzer	generate	save	Daten verschlüsselt.
	generate	prompt	Daten verschlüsselt, wenn encryptionPasswordP ein Verschlüsselungskennwort enthält.
	prompt	save	Daten verschlüsselt, wenn encryptionPasswordP ein Verschlüsselungskennwort enthält.
	prompt	prompt	Daten verschlüsselt, wenn encryptionPasswordP ein Verschlüsselungskennwort enthält.
Sicherung durch nicht berechtigten Benutzer	generate	save	Daten verschlüsselt.
	generate	prompt	Daten verschlüsselt, wenn encryptionPasswordP ein Verschlüsselungskennwort enthält.
	prompt	save	Daten verschlüsselt, wenn encryptionPasswordP ein Verschlüsselungskennwort enthält.
	prompt	prompt	Daten verschlüsselt, wenn encryptionPasswordP ein Verschlüsselungskennwort enthält.
Zurückschreibung durch nicht berechtigten Benutzer	generate	save	Daten verschlüsselt.
	generate	prompt	Daten verschlüsselt, wenn encryptionPasswordP ein Verschlüsselungskennwort enthält.
	prompt	save	Daten verschlüsselt, wenn encryptionPasswordP ein Verschlüsselungskennwort enthält.
	prompt	prompt	Daten verschlüsselt, wenn encryptionPasswordP ein Verschlüsselungskennwort enthält.

IBM Spectrum Protect-Clientverschlüsselung

Die IBM Spectrum Protect-Clientverschlüsselung verwendet den vom Wert DSM_ENCRYPT_CLIENTENCRKEY verwalteten Schlüssel, um Ihre Daten zu schützen. Die Clientverschlüsselung ist transparent für die Anwendung, von der die API verwendet wird. Allerdings sind Zurückschreibungs- und Abrufoperationen von Teilobjekten für verschlüsselte oder komprimierte Objekte nicht möglich.

Sowohl bei der IBM Spectrum Protect-Clientverschlüsselung als auch bei der anwendungsverwalteten Verschlüsselung bezieht sich das Verschlüsselungskennwort auf einen Zeichenfolgewert, mit dem der tatsächliche Verschlüsselungsschlüssel generiert wird. Der Wert für die Option für das Verschlüsselungskennwort kann 1 bis

63 Zeichen lang sein, aber der daraus generierte Schlüssel hat immer eine Länge von 8 Byte bei 56-Bit-DES, 16 Byte bei 128-Bit-AES und 32 Byte bei 256-Bit-AES.

Achtung: Wenn der Verschlüsselungsschlüssel nicht verfügbar ist, können Daten weder zurückgeschrieben noch abgerufen werden. Wenn Sie `ENABLECLIENTENCRYPTKEY` für die Verschlüsselung verwenden, wird der Verschlüsselungsschlüssel in der Serverdatenbank gespeichert. Für Objekte, die dieses Verfahren verwenden, muss die Serverdatenbank vorhanden sein und die korrekten Werte für die Objekte enthalten, damit eine ordnungsgemäße Zurückschreibung möglich ist. Stellen Sie sicher, dass Sie Ihre Serverdatenbank häufig sichern, um einen Datenverlust zu verhindern.

Dies ist im Hinblick auf die Implementierung das einfachere Verfahren; dabei wird jeweils ein Verschlüsselungszufallsschlüssel pro Sitzung generiert und auf dem IBM Spectrum Protect-Server mit dem Objekt in der Serverdatenbank gespeichert. Während der Zurückschreibung wird der gespeicherte Schlüssel für die Entschlüsselung verwendet. Bei der Verwendung dieses Verfahrens ist IBM Spectrum Protect für die Verwaltung des Schlüssels zuständig; die Anwendung muss sich überhaupt nicht damit befassen. Da der Schlüssel in der Serverdatenbank gespeichert ist, muss eine gültige IBM Spectrum Protect-Datenbank für eine Zurückschreibungsoperation eines verschlüsselten Objekts vorhanden sein. Wenn der Schlüssel zwischen der API und dem Server übertragen wird, wird er ebenfalls verschlüsselt. Die Übertragung des Schlüssels ist sicher; wenn der Schlüssel in der Datenbank des IBM Spectrum Protect-Servers gespeichert wird, wird er verschlüsselt. Die einzige Zeit, während der der Schlüssel im Exportdatenstrom nicht verschlüsselt ist, ist beim Export der Daten eines Knotens zwischen Servern.

Gehen Sie wie folgt vor, um die IBM Spectrum Protect-Clientverschlüsselung zu aktivieren:

1. Geben Sie `-ENABLECLIENTENCRYPTKEY=YES` in der Optionszeichenfolge an, die im Aufruf dsmInitEx an die API übergeben wird, oder geben Sie die Option in der Systemoptionsdatei `dsm.opt` (Windows) bzw. `dsm.sys` (UNIX oder Linux) an.
2. Geben Sie `include.encrypt` für die Objekte an, die verschlüsselt werden sollen. Um beispielsweise alle Daten zu verschlüsseln, definieren Sie Folgendes:

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

   und

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

   Geben Sie Folgendes an, um das Objekt `/FS1/DB2/FULL` zu verschlüsseln:

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

Datendeduplizierung

Die Datendeduplizierung ist eine Methode zum Verringern des Speicherbedarfs, indem redundante Daten eliminiert werden.

Übersicht

Bei IBM Spectrum Protect sind zwei Typen von Datendeduplizierung verfügbar: *clientseitige Datendeduplizierung* und *serverseitige Datendeduplizierung*.

Als *clientseitige Datendeduplizierung* wird ein Datendeduplizierungsverfahren bezeichnet, das der Client für Sichern/Archivieren verwendet, um bei der Sicherungs- und Archivierungsverarbeitung redundante Daten zu entfernen, bevor die

Daten an den IBM Spectrum Protect-Server übertragen werden. Mithilfe der clientseitigen Datendeduplizierung kann das Datenvolumen, das über ein lokales Netz gesendet wird, reduziert werden.

Die *serverseitige Datendeduplizierung* ist ein Datendeduplizierungsverfahren, das vom Server ausgeführt wird. Der IBM Spectrum Protect-Administrator kann die zu verwendende Position für die Datendeduplizierung (Client oder Server) mit dem Parameter **DEDUP** im Serverbefehl **REGISTER NODE** oder **UPDATE NODE** angeben.

Funktionale Erweiterungen

Mit der clientseitigen Datendeduplizierung haben Sie folgende Möglichkeiten:

- Bestimmte Dateien auf einem Client von der Datendeduplizierung ausschließen.
- Einen Datendeduplizierungscache aktivieren, der den Datenaustausch im Netz zwischen dem Client und dem Server reduziert. Der Cache enthält Bereiche, die bei vorherigen Teilsicherungsoperationen an den Server gesendet wurden. Der Client fragt nicht den Server, sondern seinen Cache ab, ob ein Bereich vorhanden ist.

  Geben Sie eine Größe und eine Position für einen Client-Cache an. Wenn eine Inkonsistenz zwischen dem Server und dem lokalen Cache festgestellt wird, wird der lokale Cache entfernt und neu gefüllt.

  Anmerkung: Für Anwendungen, die die IBM Spectrum Protect-API verwenden, darf der Datendeduplizierungscache wegen möglicher Sicherungsfehler nicht verwendet werden, die verursacht werden, wenn der Cache nicht mit dem IBM Spectrum Protect-Server synchron ist. Wenn mehrere, gleichzeitig ablaufende Sitzungen des Clients für Sichern/Archivieren konfiguriert sind, muss für jede Sitzung ein separater Cache konfiguriert werden.
- Aktivieren Sie sowohl die clientseitige Datendeduplizierung als auch die Komprimierung, um das vom Server gespeicherte Datenvolumen zu reduzieren. Jeder Bereich wird komprimiert, bevor er an den Server gesendet wird. Der Kompromiss besteht zwischen den Speichereinsparungen und der für die Komprimierung der Clientdaten erforderlichen Verarbeitungskapazität. Wenn Sie Daten auf dem Clientsystem komprimieren und deduplizieren, verwenden Sie im Allgemeinen etwa doppelt soviel Verarbeitungskapazität wie bei der Datendeduplizierung allein.

  Der Server kann mit deduplizierten komprimierten Daten arbeiten. Außerdem können Clients für Sichern/Archivieren vor Version 6.2 deduplizierte komprimierte Daten zurückschreiben.

Die clientseitige Datendeduplizierung verwendet den folgenden Prozess:

- Der Client erstellt Bereiche. *Bereiche* sind Teile von Dateien, die mit anderen Dateibereichen verglichen werden, um Duplikate zu identifizieren.
- Der Client und der Server arbeiten zusammen, um doppelte Bereiche zu identifizieren. Der Client sendet Bereiche, die nicht doppelt sind, an den Server.
- Nachfolgende clientseitige Datendeduplizierungsoperationen erstellen neue Bereiche. Einige oder alle dieser Bereiche können mit den Bereichen übereinstimmen, die in vorherigen Datendeduplizierungsoperationen erstellt und an den Server gesendet wurden. Übereinstimmende Bereiche werden nicht erneut an den Server gesendet.

Vorteile

Die clientseitige Deduplizierung von Daten bietet mehrere Vorteile:

- Sie kann das über das lokale Netz (LAN) gesendete Datenvolumen reduzieren.
- Die zum Identifizieren doppelter Daten erforderliche Verarbeitungsleistung wird vom Server auf Clientknoten verlagert. Die serverseitige Deduplizierung von Daten ist für Speicherpools, die für die Deduplizierung aktiviert sind, immer aktiviert. Dateien, die sich in den deduplizierungsfähigen Speicherpools befinden und die vom Client dedupliziert wurden, erfordern jedoch keine zusätzliche Verarbeitung.
- Die zum Entfernen doppelter Daten auf dem Server erforderliche Verarbeitungsleistung wird eliminiert. Dadurch werden Speichereinsparungen auf dem Server sofort wirksam.

Die clientseitige Deduplizierung von Daten hat einen möglichen Nachteil. Der Server verfügt erst dann über vollständige Kopien von Clientdateien, wenn Sie die primären Speicherpools, die die clientseitigen Bereiche enthalten, in einem nicht dedupliziertен Kopienspeicherpool sichern. (*Bereiche* sind Teile einer Datei, die während des Prozesses für die Deduplizierung von Daten erstellt werden.) Während der Speicherpoolsicherung in einem nicht dedupliziertен Speicherpool werden clientseitige Bereiche in zusammenhängenden Dateien neu erstellt.

Standardmäßig müssen primäre Speicherpools mit sequenziellem Zugriff, die für die Deduplizierung von Daten definiert sind, in nicht dedupliziertен Kopienspeicherpools gesichert werden, bevor sie zurückgefordert und doppelte Daten entfernt werden können. Mit dem Standardwert wird sichergestellt, dass der Server immer über Kopien vollständiger Dateien in einem primären Speicherpool oder einem Kopienspeicherpool verfügt.

Wichtig: Eine weitere Datenreduktion wird erzielt, wenn Sie die clientseitige Datendeduplizierung zusammen mit der Komprimierung aktivieren. Jeder Bereich wird komprimiert, bevor er an den Server gesendet wird. Bei der Komprimierung wird Speicherbereich eingespart, die Verarbeitungszeit auf der Client-Workstation verlängert sich jedoch.

Die folgenden Optionen gelten für die Datendeduplizierung:

- Deduplication
- Dedupcachepath
- Dedupcachesize
- Enablededupcache
- Exclude.dedup
- Include.dedup

Clientseitige Datendeduplizierung der API

Die *clientseitige Datendeduplizierung* wird von der API auf dem Client für Sichern/Archivieren verwendet, um redundante Daten während der Sicherungs- und Archivierungsverarbeitung zu entfernen, bevor die Daten an den IBM Spectrum Protect übertragen werden.

Die clientseitige Datendeduplizierung wird von der API verwendet, um redundante Daten während der Sicherungs- und Archivierungsverarbeitung zu entfernen, bevor die Daten an den IBM Spectrum Protect-Server übertragen werden. Mithilfe der clientseitigen Datendeduplizierung kann das Datenvolumen, das über ein lokales Netz gesendet wird, reduziert werden. Mithilfe der clientseitigen Datendeduplizierung kann auch der Speicherbereich des IBM Spectrum Protect-Servers reduziert werden.

Wenn der Client für die clientseitige Datendeduplizierung aktiviert ist und Sie eine Sicherungs- oder Archivierungsoperation ausführen, werden die Daten als Bereiche an den Server gesendet. Wenn die nächste Sicherungs- oder Archivierungsoperation ausgeführt wird, identifizieren der Client und der Server die Datenbereiche, die bereits gesichert oder archiviert wurden, und es werden nur die eindeutigen Datenbereiche an den Server gesendet.

Bei der clientseitigen Datendeduplizierung müssen der Server und die API Version 6.2 oder höher haben.

Bevor Sie die clientseitige Datendeduplizierung zum Sichern oder Archivieren Ihrer Dateien verwenden, muss das System die folgenden Voraussetzungen erfüllen:

- Für den Client muss die Option `deduplication` aktiviert sein.
- Der Server muss den Client mit dem Parameter **`DEDUP=CLIENTORSERVER`** im Befehl **`REGISTER NODE`** oder **`UPDATE NODE`** für die clientseitige Datendeduplizierung aktivieren.
- Das Speicherpoolziel für die Daten muss ein für die Datendeduplizierung aktivierter Speicherpool sein. Der für die Datendeduplizierung aktivierte Speicherpool darf nur den Einheitentyp FILE haben.
- Stellen Sie sicher, dass die Dateien an die korrekte Verwaltungsklasse gebunden sind.
- Eine Datei kann von der clientseitigen Datendeduplizierungsverarbeitung ausgeschlossen werden. Standardmäßig sind alle Dateien eingeschlossen.
- Dateien müssen größer als 2 KB sein.
- Der Server kann die maximale Transaktionsgröße für die Datendeduplizierung begrenzen, indem die Option `CLIENTDEDUPTXNLIMIT` auf dem Server gesetzt wird. Informationen zu dieser Option finden Sie in der Dokumentation für den Server.

Ist eine dieser Voraussetzungen nicht erfüllt, werden die Daten normal, d. h. ohne clientseitige Datendeduplizierung verarbeitet.

Nachfolgend einige der Einschränkungen, die für die Datendeduplizierung gelten:

- LAN-unabhängige Datenversetzung und clientseitige Datendeduplizierung schließen sich gegenseitig aus. Wenn Sie sowohl LAN-unabhängige Datenversetzung als auch clientseitige Datendeduplizierung aktivieren, werden LAN-unabhängige Datenversetzungsoperationen ausgeführt und die clientseitige Datendeduplizierung wird ignoriert.
- Verschlüsselung und clientseitige Datendeduplizierung schließen sich gegenseitig aus. Wenn Sie sowohl Verschlüsselung als auch clientseitige Datendeduplizie-

rung aktivieren, werden Verschlüsselungsoperationen ausgeführt und die clientseitige Datendeduplizierung wird ignoriert. Verschlüsselte Dateien und Dateien, die für die clientseitige Datendeduplizierung auswählbar sind, können in derselben Operation verarbeitet werden, die Verarbeitung erfolgt jedoch in unterschiedlichen Transaktionen.

Voraussetzungen:

1. In jeder Transaktion müssen entweder alle Dateien für die Datendeduplizierung eingeschlossen oder von der Datendeduplizierung ausgeschlossen werden. Sind in einer Transaktion diese Dateien gemischt, schlägt die Transaktion fehl und von der API wird der Rückkehrcode `DSM_RC_NEEDTO_ENDTXN` zurückgegeben.
2. Verwenden Sie Speichereinheitenverschlüsselung zusammen mit clientseitiger Datendeduplizierung. Da SSL in Kombination mit clientseitiger Deduplizierung verwendet wird, ist keine Clientverschlüsselung erforderlich.

- Die folgenden Funktionen sind für die clientseitige Datendeduplizierung nicht verfügbar:
  - HSM-Client (HSM = IBM Spectrum Protect for Space Management)
  - API-Puffer für gemeinsamen Zugriff
  - NAS
  - Subdateisicherung
- Pufferkopieneliminierung kann nicht mit Datenkonvertierungen wie Komprimierung, Verschlüsselung und Datendeduplizierung verwendet werden.
- Wenn Sie die clientseitige Deduplizierung verwenden, erkennt und unterlässt die API (mit RC=254) Sicherungen von Dateibereichen, die auf dem Server als verfallen markiert sind, während Daten an den Server gesendet werden. Soll die Operation wiederholt werden, müssen Sie diese Programmierung in die aufrufende Anwendung einschließen.
- Operationen für gleichzeitiges Schreiben auf dem Server haben Vorrang vor der clientseitigen Datendeduplizierung. Sind Operationen für gleichzeitiges Schreiben aktiviert, erfolgt keine clientseitige Datendeduplizierung.

  Einschränkung: Ist die clientseitige Datendeduplizierung aktiviert, ist für die API keine Wiederherstellung von einem Zustand möglich, bei dem kein Speicherbereich mehr im Zielpool des Servers verfügbar ist, selbst wenn ein nächster Pool definiert ist. Der Ursachencode DSM_RS_ABORT_DESTINATION_POOL_CHANGED zum Stoppen wird zurückgegeben und die Operation schlägt fehl. Es gibt zwei Möglichkeiten zur Wiederherstellung in dieser Situation:

  1. Bitten Sie den Administrator, dem ursprünglichen Dateipool weitere Arbeitsdatenträger hinzuzufügen.
  2. Wiederholen Sie die Operation mit inaktivierter Datendeduplizierung.

Für noch geringere Anforderungen an die Bandbreite können Sie einen lokalen Cache für die Datendeduplizierung aktivieren. Der lokale Cache verhindert, dass Abfragen an den IBM Spectrum Protect-Server gesendet werden. Der Standardwert für ENABLEDEDUPCACHE ist NO, d. h., der Cache ist mit dem Server synchron. Wenn der Cache mit dem Server nicht synchron ist, sendet die Anwendung alle Daten erneut. Wenn Ihre Anwendung eine fehlgeschlagene Transaktion wiederholen kann und der lokale Cache verwendet werden soll, setzen Sie die Option ENABLEDEDUPCACHE in der Datei dsm.opt (Windows) oder in der Datei dsm.sys (UNIX) auf YES.

Am Ende einer Zurückschreibung wird, wenn *alle* Daten über die API zurückgeschrieben wurden und das Objekt vom Client dedupliziert wurde, ein End-to-End-Digest berechnet und mit dem zum Zeitpunkt der Sicherung berechneten Wert verglichen. Stimmen diese Werte nicht überein, wird der Fehler DSM_RC_DIGEST_VALIDATION_ERROR zurückgegeben. Wenn eine Anwendung diesen Fehler empfängt, sind die Daten beschädigt. Dieser Fehler kann auch die Folge eines temporären Fehlers im Netz sein; wiederholen Sie in diesem Fall die Zurückschreibung oder den Abruf.

Das folgende Beispiel zeigt den Befehl zum Abfragen der Sitzung mit Datendeduplizierungsinformationen:

```
Werte für dsmQuerySessInfo:
Serverinformationen:
Servername: SERVER1
Server-Host: AVI
Server-Port: 1500
Serverdatum: 2009/10/6 20:48:51
Servertyp: Windows
Serverversion: 6.2.0.0
Aufbewahrungsschutz für Archivierung (Server): NEIN
Clientinformationen:
Clientknotentyp: API Test1
Begrenzer für Dateibereich (Client): :
Begrenzer für HL & LL (Client): \
Clientkomprimierung: Vom Client bestimmt (3u)
Archivierung löschen (Client): Client kann archivierte Objekte löschen
Sicherung löschen (Client): Client kann Sicherungsobjekte NICHT löschen
Maximale Anzahl Objekte in Transaktion mit mehreren Objekten: 4096
LAN-unabhängig aktiviert: NEIN
Deduplizierung: Client oder Server
Allgemeine Sitzungsinformationen:
Knoten: AVI
Eigner:
API-Konfigurationsdatei:
```

Das folgende Beispiel zeigt den Befehl zum Abfragen der Verwaltungsklasse mit Datendeduplizierungsinformationen:

```
Maßnahmeninformationen:
Domänenname: DEDUP
Name der Maßnahmengruppe: DEDUP
Maßnahmenaktivierungsdatum: 0/0/0 0:0:0
Standardverwaltungsklasse: DEDUP
Aufbewahrungszeitraum für Sicherung: 30 Tage
Aufbewahrungszeitraum für Archivierung: 365 Tage
Verwaltungsklasse 1:
Name: DEDUP
Beschreibung: Deduplizierung - Standardwerte
Name der Sicherungskopiengruppe: STANDARD
Häufigkeit: 0
Versionsdaten vorhanden: 2
Versionsdaten gelöscht: 1
Extraversionen aufbewahren: 30
Nur Versionen aufbewahren: 60
Kopienziel: AVIFILEPOOL
LAN-unabhängiges Ziel: NEIN
Daten deduplizieren: JA

Name der Archivierungskopiengruppe: STANDARD
Häufigkeit: 10000
Versionen aufbewahren: 365
Kopienziel: AVIFILEPOOL
```

```
LAN-unabhängiges Ziel: NEIN
Anfangsversion aufbewahren: CREATE
Mindestversionen aufbewahren: 65534
Daten deduplizieren: JA
```

Zugehörige Verweise:

Option Deduplication

Dateien von der Datendeduplizierung ausschließen

Falls gewünscht, können Sie Sicherungs- oder Archivierungsdateien von der Datendeduplizierung ausschließen.

Um Dateien von der Datendeduplizierungsverarbeitung auszuschließen, führen Sie diese Schritte aus:

1. Definieren Sie die Option `exclude.dedup` für die Objekte, die ausgeschlossen werden sollen.

   Um beispielsweise alle Deduplizierungsdaten für UNIX-Systeme auszuschließen, definieren Sie Folgendes:

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

2. Um alle Deduplizierungsdaten für Windows-Systeme auszuschließen, definieren Sie Folgendes:

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

Wichtig: Wenn ein Objekt an einen Datendeduplizierungspool gesendet wird, erfolgt selbst dann eine Datendeduplizierung auf dem Server, wenn das Objekt von der clientseitigen Datendeduplizierung ausgeschlossen ist.

Dateien für die Datendeduplizierung einschließen

Falls gewünscht, können Sie Sicherungs- oder Archivierungsdateien für die Datendeduplizierung einschließen.

Um die Liste der einzuschließenden Dateien einzugrenzen, kann die Option `include.dedup` zusammen mit der Option `exclude.dedup` verwendet werden.

Standardmäßig werden alle auswählbaren Objekte für die Datendeduplizierung eingeschlossen.

Nachfolgend einige Beispiele für UNIX und Linux:

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

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

Nachfolgend einige Beispiele für Windows:

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

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

Serverseitige Datendeduplizierung

Die serverseitige Datendeduplizierung ist eine Datendeduplizierung, die vom Server ausgeführt wird.

Der IBM Spectrum Protect-Administrator kann die zu verwendende Position für die Datendeduplizierung (Client oder Server) mit dem Parameter **DEDUP** im Serverbefehl **REGISTER NODE** oder **UPDATE NODE** angeben.

In einem Speicherpool (Dateipool), der für die Datendeduplizierung aktiviert ist, wird nur eine einzige Instanz eines Datenbereichs aufbewahrt. Andere Instanzen desselben Datenbereichs werden durch einen Verweis auf die aufbewahrte Instanz ersetzt.

Weitere Informationen zur serverseitigen Datendeduplizierung finden Sie in der Dokumentation für den IBM Spectrum Protect-Server.

Anwendungsübernahme

Wenn der IBM Spectrum Protect-Server wegen einer Betriebsunterbrechung nicht mehr verfügbar ist, können Anwendungen, die die API verwenden, für die Datenwiederherstellung automatisch eine Übernahme auf einem Sekundärserver durchführen.

Der IBM Spectrum Protect-Server, zu dem der Client und die API während des normalen Produktionsprozesses eine Verbindung herstellen, wird als *Primärserver* bezeichnet. Wenn der Primärserver für die Knotenreplikation konfiguriert ist, wird er auch als *Quellenreplikationsserver* bezeichnet. Die Clientknotendaten auf dem Quellenreplikationsserver können auf den *Zielreplikationsserver* repliziert werden. Dieser Server wird auch als *Sekundärserver* bezeichnet und er ist der Server, auf dem der Client automatisch eine Übernahme durchführt, wenn der Primärserver ausfällt.

Der Client und die API müssen für die automatisierte Clientübernahme konfiguriert und mit einem Server der Version 7.1 (oder einer höheren Version) verbunden sein, der Clientknotendaten repliziert. Die Konfiguration für die API ist dieselbe wie die Konfiguration für den Client für Sichern/Archivieren.

Im normalen Betrieb werden Verbindungsdaten für den Sekundärserver automatisch während des Anmeldeprozesses vom Primärserver an den Client gesendet. Die Informationen zum Sekundärserver werden automatisch in der Clientoptionsdatei gespeichert.

Die Clientanwendung versucht bei jeder Anmeldung beim IBM Spectrum Protect-Server, den Primärserver anzusprechen. Wenn der Primärserver nicht verfügbar ist, führt die Anwendung, unter Verwendung der Informationen zum Sekundärserver in der Clientoptionsdatei, eine Übernahme auf dem Sekundärserver durch. Im Übernahmemodus kann die Anwendung den Sekundärserver abfragen und replizierte Daten zurückschreiben oder abrufen.

Sie müssen die Anwendung mindestens einmal auf dem Primärserver sichern. Die API kann nur dann eine Übernahme auf dem Sekundärserver zum Wiederherstellen von Daten durchführen, wenn die Daten vom Clientknoten vom Primärserver auf den Sekundärserver repliziert wurden.

Zugehörige Konzepte:

Konfiguration und Verwendung der automatisierten Clientübernahme

Übernahmestatusinformationen

Die API stellt Statusinformationen bereit, die von Anwendungen verwendet werden können, um den Übernahmestatus und den Status replizierter Clientdaten auf dem Sekundärserver festzustellen.

Der Replikationsstatus gibt an, ob die neueste Sicherung auf dem Sekundärserver repliziert wurde. Wenn die Zeitmarke der neuesten Sicherungsoperation in der API mit der Zeitmarke der Sicherung auf dem Sekundärserver übereinstimmt, ist der Replikationsstatus aktuell. Wenn die beiden Zeitmarken nicht übereinstimmen, ist der Replikationsstatus nicht aktuell und sind die replizierten Daten möglicherweise nicht auf dem neuesten Stand.

Die folgenden Replikationsstatusinformationen werden in der Antwort auf die Dateibereichsabfrage (**query filespace**) im Funktionsaufruf **dsmGetNextQObj** in der Struktur **qryRespFSData** zurückgegeben:

Tabelle 14. Von der API zurückgemeldete Replikationsstatusinformationen

Statusinformationen	Typ	Definition
Start der letzten Replikation	**lastReplStartDate**	Der Zeitpunkt, zu dem die letzte Replikation gestartet wurde.
Ende der letzten Replikation	**lastReplCmpltDate**	Der Zeitpunkt, zu dem die letzte Replikation beendet wurde, auch wenn es einen Fehler gab.
Letztes Sicherungsspeicherdatum (Server)	**lastBackOpDateFromServer**	Die letzte Speicherungszeitmarke, die auf dem Server gespeichert wurde.
Letztes Sicherungsspeicherdatum (lokal)	**lastBackOpDateFromLocal**	Die letzte Speicherungszeitmarke, die auf dem Client gespeichert wurde.

Der Übernahmestatus wird durch das Feld **bIsFailOverMode** in der Struktur **dsmInitExOut_t** zurückgemeldet.

Die Struktur- und Typdefinitionen der API finden Sie in Anhang B, „Quellendateien für API-Typdefinitionen", auf Seite 171.

Der Rückgabecode DSM_RC_SIGNON_FAILOVER_MODE gibt an, dass Client und API eine Übernahme auf dem Sekundärserver ausgeführt haben und im Übernahmemodus ausgeführt werden.

Beispiel für das Anmelden während einer Übernahme

Die folgende Ausgabe ist ein Beispiel für das Anmelden des Servers während einer Übernahme:

```
Anmeldung
Anmeldung für Knoten khoyt (owner) mit Kennwort khoytpass läuft
ANS2106I Verbindung zum primären IBM Spectrum Protect-Server 123.45.6.78 ist
fehlgeschlagen

ANS2107I Es wird versucht, eine Verbindung zum sekundären Server TARGET unter
123.45.6.79 : 1501 herzustellen

ANS2108I Verbindung zum sekundären Server TARGET hergestellt.
Kennung bei Rückgabe = 1

************************************************************
Nach dsmInitEx:
Server TARGET Ver/Rel/St  7/1/0/0
Benutzerberechtigungen:       Owner
Name des Replikationsservers: TARGET
Name des Home-Servers:        MINE
Verbindung zum Replikationsserver hergestellt
************************************************************
```

Beispiel für den Befehl zum Abfragen der Sitzung

Es folgt eine Beispielausgabe für den Befehl zum Abfragen der Sitzung (**query session**). Es werden die Informationen zum Sekundärserver (Replikationsserver) aufgelistet:

```
query session
Werte für dsmQuerySessInfo:
  Serverinformationen:
    Servername: TARGET
    Server-Host: 123.45.6.79
    Serveranschluss: 1500
        Serverdatum: 2013/5/21  14:13:32
    Servertyp: Windows
        Serverversion: 7.1.0.0
    Aufbewahrungsschutz für Archivierung (Server): NEIN
  Replikationsserverinformationen
    Name des Home-Servers: MINE
    Name des Replikationsservers: TARGET
       Host: 123.45.6.79
       Anschluss: 1501
    Übernahmestatus: Verbindung zum Replikationsserver hergestellt
  Clientinformationen:
        Clientknotentyp: Unix
    Begrenzer für Dateibereich (Client): /
    Begrenzer für HL & LL (Client): /
    Clientkomprimierung: Vom Client bestimmt (3u)
        Archivierung löschen (Client): Client kann archivierte Objekte löschen
        Sicherung löschen (Client): Client kann Sicherungsobjekte NICHT löschen
    Maximale Anzahl Objekte in Transaktion mit mehreren Objekten: 4096
        LAN-unabhängig aktiviert: NEIN
        Deduplizierung: Nur Server
  Allgemeine Sitzungsinformationen:
        Knoten: KHOYT
    Zugriffsknoten:
    Eigner:
    API-Konfigurationsdatei:   Maßnahmeninformationen:
        Domänenname: STANDARD
    Name der Maßnahmengruppe: STANDARD
    Maßnahmenaktivierungsdatum: 0/0/0 0:0:0
        Standardverwaltungsklasse: STANDARD
    Aufbewahrungszeitraum für Sicherung: 30 Tage
        Aufbewahrungszeitraum für Archivierung: 365 Tage
```

Beispiel für den Befehl für eine Dateibereichsabfrage

Es folgt eine Beispielausgabe für den Befehl für eine Dateibereichsabfrage (**query filespace**). Es wird der Replikationsstatus eines Dateibereichs auf dem Sekundärserver aufgelistet:

```
filespace query
Abzufragendes Dateibereichsmuster:*
Sind die vorherigen Antworten korrekt (y/n/q)?

Dateibereichsname  Typ          Belegung   Kapazität  Start         Ende
-------------- -------------- ---------- ---------- ------------- -------------
/fs             API:Beispiel           100        300   0/0/0 0:0:0   0/0/0 0:0:0

     Start der letzten Replikation:  2013/5/21 21:3:2
     Ende der letzten Replikation:   2013/5/21 21:3:3
                                          Server                    Lokal
     Letztes Sicherungsspeicherdatum:     2013/5/21 21:18:25        2013/5/21 21:18:25
     Letztes Archivierungsspeicherdatum:  0/0/0 0:0:0               0/0/0 0:0:0
     Letztes HSM-Speicherdatum:           0/0/0 0:0:0               0/0/0 0:0:0
     FSINFO: Beispiel für API-Dateibereichsinformationen
```

Zugehörige Verweise:

„dsmGetNextQObj" auf Seite 118

Beispielablaufdiagramme für Sicherung und Archivierung

Der Entwurf der API basiert auf direkten Logikabläufen und klaren Übergängen zwischen den verschiedenen Zuständen des Anwendungsclients. Aufgrund dieser deutlichen Zustandsübergänge werden Logik- und Programmfehler frühzeitig im Entwicklungszyklus abgefangen und somit Qualität und Zuverlässigkeit des Systems erheblich verbessert.

Sie können beispielsweise einen Aufruf **dsmSendObj** erst absetzen, wenn eine Transaktion gestartet wurde und zuvor ein Aufruf **dsmBindMC** für das Objekt, das gesichert wird, erfolgt ist.

Abb. 12 auf Seite 63 zeigt das Zustandsdiagramm für die Ausführung von Sicherungs- oder Archivierungsoperationen innerhalb einer Transaktion. Der Pfeil, der von „In 'Objekt senden'" auf **dsmEndTxn** zeigt, gibt an, dass ein Aufruf **dsmEndTxn** nach einem Aufruf **dsmSendObj** oder **dsmSendData** gestartet werden kann. Dies kann ausgeführt werden, wenn während des Sendevorgangs eines Objekts eine Fehlerbedingung aufgetreten ist und die gesamte Operation gestoppt werden soll. In diesem Fall müssen Sie für das Votum DSM_VOTE_ABORT angeben. Rufen Sie jedoch unter normalen Umständen **dsmEndSendObj** auf, bevor Sie die Transaktion beenden.

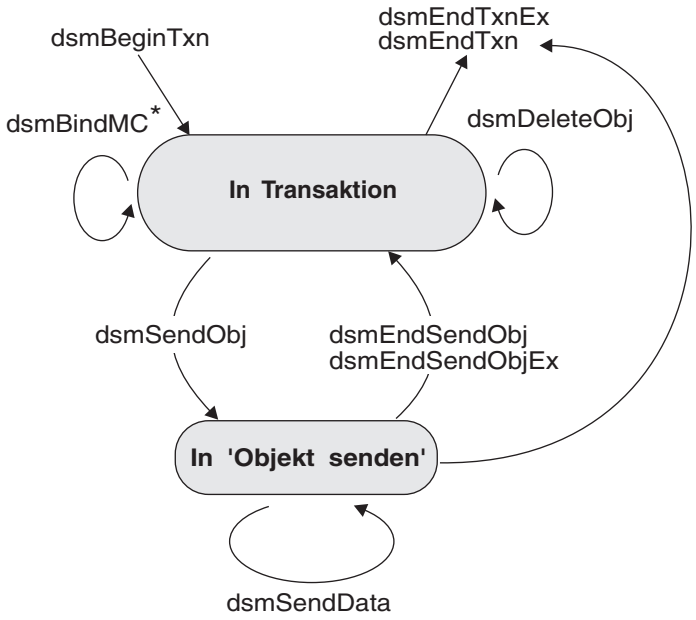


Abbildung 12. Zustandsdiagramm für Sicherungs- und Archivierungsoperationen

Abb. 13 auf Seite 64 zeigt das Ablaufdiagramm für die Ausführung von Sicherungs- oder Archivierungsoperationen innerhalb einer Transaktion.

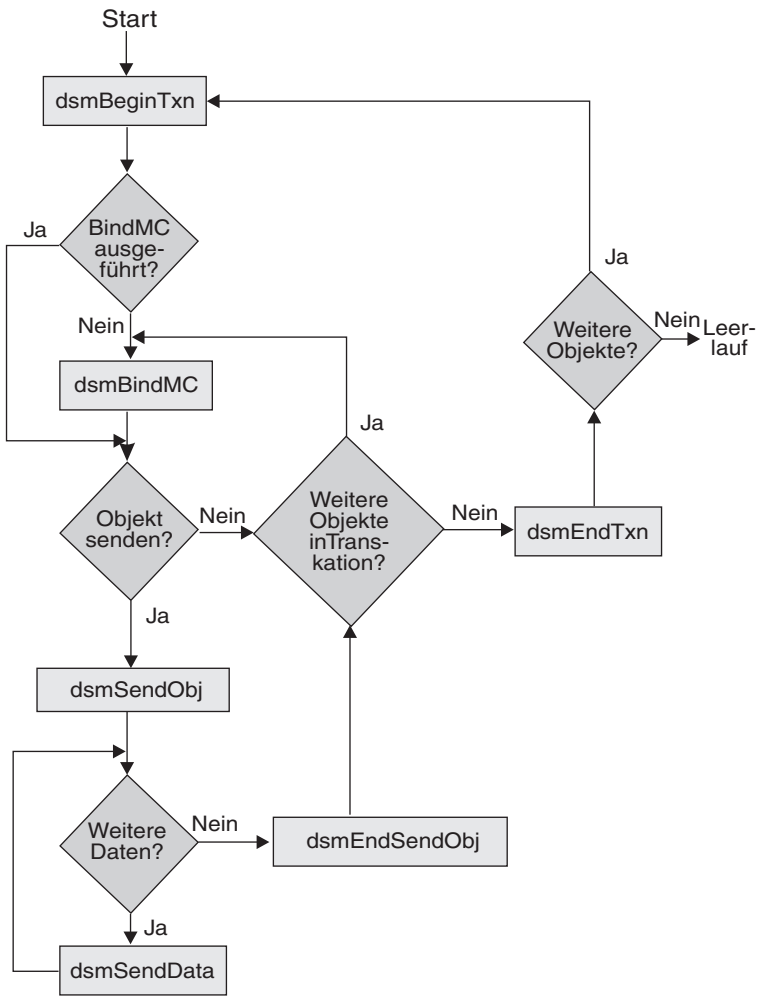


Abbildung 13. Ablaufdiagramm für Sicherungs- und Archivierungsoperationen

Das wichtigste Merkmal in diesen beiden Diagrammen ist die Schleife zwischen den folgenden API-Aufrufen innerhalb einer Transaktion:

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

Der Aufruf **dsmBindMC** ist insofern einzigartig, als Sie ihn innerhalb oder außerhalb einer Transaktionsgrenze starten können. Sie können ihn auch, falls erforderlich, von einer anderen Transaktion aus starten. Einzige Voraussetzung für den Aufruf **dsmBindMC** ist, dass er vor der Sicherung oder Archivierung eines Objekts erfolgt. Wenn das Objekt, das gesichert oder archiviert wird, keiner Verwaltungsklasse zugeordnet ist, wird von **dsmSendObj** ein Fehlercode zurückgegeben. In dieser Situation wird die Transaktion beendet, indem **dsmEndTxn** aufgerufen wird. (Diese Fehlerbedingung wird in dem Ablaufdiagramm nicht gezeigt.)

Das Ablaufdiagramm veranschaulicht, wie eine Anwendung Transaktionen mit mehreren Objekten verwenden würde. Es zeigt, wo Entscheidungspunkte gesetzt werden können, um festzulegen, ob das Objekt, das gesendet wird, in die Transak-

tion passt oder ob eine neue Transaktion gestartet werden muss.

Codebeispiel für API-Funktionen, mit denen Daten zum IBM Spectrum Protect-Speicher gesendet werden

Dieses Beispiel veranschaulicht die Verwendung von API-Funktionen, mit denen Daten an IBM Spectrum Protect-Speicher gesendet werden. Der Aufruf **dsmSendObj** erscheint in einer Anweisung 'switch', sodass verschiedene Parameter abhängig davon, ob eine Sicherungs- oder Archivierungsoperation ausgeführt wird, aufgerufen werden können.

Der Aufruf **dsmSendData** erfolgt innerhalb einer Schleife, die wiederholt Daten sendet, bis ein Flag gesetzt wird, das der Programmausführung das Verlassen der Schleife ermöglicht. Die gesamte Sendeoperation wird innerhalb der Transaktion ausgeführt.

Der dritte Parameter im Aufruf **dsmSendObj** gibt einen Puffer an, der die Archivierungsbeschreibung enthält. Da Sicherungsobjekte keine Beschreibung haben, ist der Wert für diesen Parameter bei der Sicherung eines Objekts NULL.

Abb. 8 auf Seite 30 zeigt ein Beispiel für die Verwendung des Funktionsaufrufs **dsmBindMC**.

```
if ((rc = dsmBeginTxn(dsmHandle)) )       /* API-Sitzungskennung */
{
   printf("*** dsmBeginTxn fehlgeschlagen: ");
   rcApiOut(dsmHandle, rc);
   return;
}

/*  dsmBindMC aufrufen, falls nicht bereits erfolgt*/
objAttr.sizeEstimate.hi = 0;       /* Schätzung für      */
objAttr.sizeEstimate.lo = 32000;   /* Objektgröße        */
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 fehlgeschlagen: ");
   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 fehlgeschlagen: ");
      rcApiOut(dsmHandle, rc);
      done = bTrue;
   }
   /* DataBlk-Puffer für nächste Sendeoperation anpassen   */
}
rc = dsmEndSendObj(dsmHandle);
if (rc)
{
   printf("*** dsmEndSendObj fehlgeschlagen: ");
   rcApiOut(dsmHandle, rc);
}
txn_reason = 0;
c = dsmEndTxn(dsmHandle,            /* API-Sitzungskennung      */
             DSM_VOTE_COMMIT,      /* Transakt. festschreiben  */
             &txn_reason);         /*Ursache b. Transakt.-Abbr.*/
if (rc || txn_reason)
{
   printf("*** dsmEndTxn fehlgeschlagen: rc = ");
   rcApiOut(dsmHandle, rc);
   printf("   Ursache =
}
```

Abbildung 14. Beispiel für das Senden von Daten an einen Server

Dateigruppierung

Die IBM Spectrum Protect-API verfügt über ein Protokoll zum Gruppieren logischer Dateien, mit dem mehrere einzelne Objekte in Gruppen zusammengefasst werden. Sie können diese Gruppen auf dem Server als eine logische Gruppe referenzieren und verwalten. Bei einer logischen Gruppe müssen alle Gruppenmitglieder und der Gruppenleiter zu demselben Knoten und Dateibereich auf dem Server gehören.

Jede logische Gruppe hat einen Gruppenleiter. Wenn der Gruppenleiter gelöscht wird, wird auch die Gruppe gelöscht. Sie können kein Mitglied löschen, das zu einer Gruppe gehört. Der Verfall aller Mitglieder in einer Gruppe ist vom Gruppenleiter abhängig. Ist beispielsweise ein Mitglied für den Verfall markiert, verfällt es erst, wenn auch der Gruppenleiter verfällt. Ist ein Mitglied jedoch nicht für den Verfall markiert und der Gruppenleiter verfällt, verfallen auch alle Mitglieder.

Dateigruppen enthalten nur Sicherungsdaten und können keine Archivierungsdaten enthalten. Für Archivierungsobjekte kann über das Feld **Archivierungsbeschreibung** eine Art von Gruppierung erfolgen, sofern eine Anwendung dies erfordert.

Mit dem Aufruf **dsmGroupHandler** werden die Operationen gruppiert. Die Funktion **dsmGroupHandler** muss innerhalb einer Transaktion aufgerufen werden. Die meisten Fehlerbedingungen für Gruppen werden entweder in Aufrufen **dsmEndTxn** oder **dsmEndTxnEx** abgefangen.

Die Ausgabestruktur in **dsmEndTxnEx** enthält ein neues Feld, **groupLeaderObjId**. Dieses Feld enthält die Objekt-ID des Gruppenleiters, wenn eine Gruppe in dieser Transaktion geöffnet wurde. Eine Gruppe kann transaktionsübergreifend erstellt werden. Eine Gruppe wird erst festgeschrieben oder gespeichert, wenn auf dem Server eine Operation zum Schließen ausgeführt wird. Bei der Funktion **dsmGroupHandler** handelt es sich um eine Schnittstelle, die fünf verschiedene Operationen akzeptieren kann. Diese umfassen:

- DSM_GROUP_ACTION_OPEN
- DSM_GROUP_ACTION_CLOSE
- DSM_GROUP_ACTION_ADD
- DSM_GROUP_ACTION_ASSIGNTO
- DSM_GROUP_ACTION_REMOVE

In Tabelle 15 sind die Aktionen für den Funktionsaufruf **dsmGroupHandler** aufgelistet:

Tabelle 15. Funktion dsmGroupHandler

Aktion	Beschreibung
OPEN	Mit der Aktion OPEN wird eine Gruppe erstellt. Das nächste Objekt, das gesendet wird, wird der Gruppenleiter. Der Gruppenleiter darf keinen Inhalt haben. Alle Objekte nach dem ersten Objekt werden Mitglieder, die der Gruppe hinzugefügt werden. Um eine Gruppe zu erstellen, öffnen Sie eine Gruppe und übergeben Sie eine eindeutige Zeichenfolge, um die Gruppe zu identifizieren. Diese eindeutige Kennung ermöglicht es, mehrere Gruppen mit demselben Namen zu öffnen. Das nächste Objekt, das gesendet wird, nachdem die Gruppe geöffnet wurde, ist der Gruppenleiter. Alle anderen Objekte, die gesendet werden, sind Gruppenmitglieder.

Tabelle 15. Funktion dsmGroupHandler (Forts.)

Aktion	Beschreibung
CLOSE	Mit der Aktion CLOSE wird eine offene Gruppe festgeschrieben und gespeichert. Um die Gruppe zu schließen, übergeben Sie den Objektnamen und die eindeutige Zeichenfolge, die in der Operation zum Öffnen verwendet wird. Die Anwendung muss auf offene Gruppen prüfen und diese, falls erforderlich, schließen oder löschen. Eine Gruppe wird erst festgeschrieben oder gespeichert, wenn sie geschlossen wird. Eine Aktion CLOSE schlägt unter den folgenden Bedingungen fehl: • Die Gruppe, die Sie zu schließen versuchen, hat denselben Namen wie eine vorhandene offene Gruppe. • Zwischen der aktuellen geschlossenen Gruppe und der neuen Gruppe mit demselben Namen, die geschlossen werden soll, besteht eine Inkompatibilität in Bezug auf die Verwaltungsklasse. Führen Sie in diesem Fall die folgenden Schritte aus: 1. Fragen Sie die vorherige geschlossene Gruppe ab. 2. Wenn die Verwaltungsklasse der vorhandenen geschlossenen Gruppe nicht mit der Verwaltungsklasse übereinstimmt, die der aktuellen offenen Gruppe zugeordnet ist, geben Sie einen Funktionsaufruf **dsmUpdateObject** des Typs DSM_BACKUPD_MC aus. Mit diesem Befehl wird die vorhandene Gruppe mit der neuen Verwaltungsklasse aktualisiert. 3. Rufen Sie die Aktion CLOSE auf.
ADD	Mit der Aktion ADD wird ein Objekt einer Gruppe hinzugefügt. Alle Objekte, die nach der Aktion ADD gesendet werden, werden der Gruppe zugeordnet.
ASSIGNTO	Mit der Aktion ASSIGNTO wird dem Client die Zuordnung von Objekten, die auf dem Server vorhanden sind, zu der deklarierten Peergruppe ermöglicht. Mit dieser Transaktion wird die Peergruppenbeziehung definiert. Die Aktion ASSIGNTO entspricht mit Ausnahme der folgenden Punkte der Aktion ADD: • Die Aktion ADD findet Anwendung auf Objekte in unvollständigen Transaktionen. • Die Aktion ASSIGNTO findet Anwendung auf ein Objekt, das sich auf dem Server befindet.
REMOVE	Mit der Aktion REMOVE wird ein Mitglied oder eine Liste mit Mitgliedern aus einer Gruppe entfernt. Ein Gruppenleiter kann nicht aus einer Gruppe entfernt werden. Ein Gruppenmitglied muss entfernt werden, bevor es gelöscht werden kann.

Verwenden Sie die folgenden Abfragetypen für die Gruppenunterstützung:
- **qtBackupGroups**
- **qtOpenGroups**

Mit **qtBackupGroups** werden Gruppen abgefragt, die geschlossen sind, während mit **qtOpenGroups** Gruppen abgefragt werden, die offen sind. Für den Abfragepuffer für die neuen Typen gibt es die Felder **groupLeaderObjId** und **objType**. Die Abfrage wird abhängig von den Werten für diese beiden Felder unterschiedlich ausgeführt. Die folgende Tabelle zeigt einige Abfragemöglichkeiten:

Tabelle 16. Beispiele für Abfragen

groupLeaderObjId.hi	groupLeaderObjId.lo	objType	Ergebnis
0	0	NULL	Gibt eine Liste aller Gruppenleiter zurück.
grpLdrObjId.hi	grpLdrObjId.lo	0	Gibt eine Liste für alle Gruppenmitglieder zurück, die dem angegebenen Gruppenleiter (**grpLdrObjId**) zugeordnet sind.
grpLdrObjId.hi	grpLdrObjId.lo	objType	Gibt unter Verwendung von **BackQryRespEnhanced3** eine Liste für jedes Gruppenmitglied zurück, das dem angegebenen Gruppenleiter (**grpLdrObjId**) zugeordnet ist und einen übereinstimmenden Objekttyp (**objType**) hat.

Die Antwortstruktur (**qryRespBackupData**) von **dsmGetNextQObj** umfasst zwei Felder für die Gruppenunterstützung:

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

Bei diesen Feldern handelt es sich um boolesche Flags. Das folgende Beispiel zeigt das Erstellen der Gruppe, das Hinzufügen von Mitgliedern zu der Gruppe und das Schließen der Gruppe, um diese auf dem IBM Spectrum Protect-Server festzuschreiben.

```
dsmBeginTxn
  dsmGroupHandler (PEER, OPEN, leader, uniqueId)
 dsmBeginSendObj
  dsmEndSendObj
dsmEndTxnEx (mit objId des Leiters)
Schleife für mehrere Transaktionen
{
 dsmBeginTxn
    dsmGroupHandler (PEER, ADD, member, groupLeaderObjID)
    Schleife für mehrere Objekte
  {
   dsmBeginSendObj
      Schleife für Daten
   {
    dsmSendData
   }
   dsmEndSendObj
  }
 dsmEndTxn
}
dmBeginTxn
  dsmGroupHandler(CLOSE)
dsmEndTxn
```

Abbildung 15. Beispiel für Pseudocode zum Erstellen einer Gruppe

Beispielcode können Sie dem Musterprogramm für eine Gruppe, `dsmgrp.c`, entnehmen, das sich im API-Verzeichnis `sampsrc` befindet.

Daten von einem Server empfangen

Anwendungsclients können Daten oder benannte Objekte und ihre zugehörigen Daten aus IBM Spectrum Protect-Speicher über die Zurückschreibungs- und Abruffunktionen empfangen. Die Zurückschreibungsfunktion greift auf Objekte zu, die zuvor gesichert wurden; die Abruffunktion greift auf Objekte zu, die zuvor archiviert wurden.

Einschränkung: Die API kann nur Objekte zurückschreiben und abrufen, die mithilfe von API-Aufrufen gesichert oder archiviert wurden.

Zurückschreibungs- und Abruffunktionen starten beide mit einer Abfrageoperation. Abhängig davon, ob die Daten ursprünglich gesichert oder archiviert wurden, gibt die Abfrage unterschiedliche Informationen zurück. Beispielsweise gibt eine Abfrage für Sicherungsobjekte Informationen zurück, die angeben, ob ein Objekt aktiv oder inaktiv ist, während eine Abfrage für Archivierungsobjekte Informationen wie Objektbeschreibungen zurückgibt. Beide Abfragen geben Objekt-IDs zurück, mit deren Hilfe das Objekt auf dem Server eindeutig identifiziert wird.

Zurückschreibung oder Abruf von Teilobjekten

Der Anwendungsclient kann nur einen Teil des Objekts empfangen. Das wird als Zurückschreibung von Teilobjekten bzw. Abruf von Teilobjekten bezeichnet.

Achtung: Eine Teilzurückschreibung oder ein Teilabruf komprimierter oder verschlüsselter Objekte führt zu unvorhersehbaren Ergebnissen.

Anmerkung: Wenn Sie Ihre Anwendung zur Verwendung einer Zurückschreibung oder eines Abruf von Teilobjekten codieren, können Sie Daten während des Sendevorgangs nicht komprimieren. Um dies durchzusetzen, setzen Sie *ObjAttr.objCompressed* auf *bTrue*.

Um eine Zurückschreibung oder einen Abruf von Teilobjekten durchzuführen, ordnen Sie die beiden folgenden Datenfelder jedem **GetList**-Eintrag des Objekts zu:

offset Die relative Byteadresse des Objekts, ab der Daten zurückgegeben werden sollen.

length Die Anzahl der zurückzugebenden Objektbyte.

Verwenden Sie DSM_MAX_PARTIAL_GET_OBJ, um die maximale Anzahl der Objekte zu bestimmen, die eine Zurückschreibung oder einen Abruf von Teilobjekten für eine bestimmte **dsmBeginGetData**-Liste durchführen können.

Die folgenden, im Aufruf **dsmBeginGetData** verwendeten Datenfelder legen fest, welcher Abschnitt des Objekts zurückgeschrieben oder abgerufen wird:

- Sind die relative Adresse und die Länge null, wird das gesamte Objekt aus dem IBM Spectrum Protect-Speicher zurückgeschrieben oder abgerufen.
- Ist die relative Adresse größer als null, die Länge aber null, wird das Objekt ab der relativen Adresse bis zum Ende zurückgeschrieben oder abgerufen.
- Ist die Länge größer als null, wird nur der Abschnitt des Objekts ab der relativen Adresse mit der angegebenen Länge zurückgeschrieben oder abgerufen.

Daten zurückschreiben oder abrufen

Nachdem eine Abfrage ausgeführt und eine Sitzung mit dem IBM Spectrum Protect-Server hergestellt wurde, können Sie eine Prozedur zum Zurückschreiben oder Abrufen von Daten ausführen.

Führen Sie die folgenden Schritte aus, um Daten zurückzuschreiben oder abzurufen:

1. Fragen Sie den IBM Spectrum Protect-Server nach Sicherungs- oder Archivierungsdaten ab.
2. Legen Sie die Objekte fest, die zurückgeschrieben oder vom Server abgerufen werden sollen.
3. Sortieren Sie die Objekte anhand des Felds **Zurückschreibungsreihenfolge**.
4. Senden Sie den Aufruf **dsmBeginGetData** mit der Liste der Objekte, auf die zugegriffen werden soll.
5. Senden Sie den Aufruf **dsmGetObj**, um jedes Objekt vom System abzurufen. Unter Umständen sind mehrere Aufrufe **dsmGetData** für jedes Objekt erforderlich, um alle zugehörigen Objektdaten abzurufen. Senden Sie den Aufruf **dsmEndGetObj**, nachdem alle Daten für ein Objekt abgerufen wurden.
6. Senden Sie den Aufruf **dsmEndGetData**, nachdem alle Daten für alle Objekte empfangen wurden, oder um die Empfangsoperation zu beenden.

Server abfragen

Fragen Sie vor dem Beginn einer Zurückschreibungs- oder Abrufoperationen zunächst den IBM Spectrum Protect-Server ab, um die Objekte zu bestimmen, die aus dem Speicher abgerufen werden können.

Um die Abfrage zu senden, muss die Anwendung die Parameterlisten und Strukturen für den Aufruf **dsmBeginQuery** verwenden. Die Struktur muss den Dateibereich, den die Abfrage prüft, und Musterübereinstimmungseinträge für die Felder für den übergeordneten Namen und den untergeordneten Namen umfassen. Wenn die Sitzung mit dem Wert NULL für den Eignernamen initialisiert wurde, müssen Sie das Feld für den Eigner nicht angeben. Wenn die Sitzung jedoch mit einem expliziten Eignernamen initialisiert wurde, werden nur Objekte zurückgegeben, denen dieser Eignername zugeordnet ist.

Die zeitpunktgesteuerte Abfrage **BackupQuery** stellt eine Momentaufnahme des Systems zu einem bestimmten Zeitpunkt bereit. Indem Sie ein gültiges Datum angeben, können Sie alle Dateien abfragen, die bis zu diesem Zeitpunkt gesichert wurden. Selbst wenn ein Objekt über eine aktive Sicherung eines späteren Datums verfügt, überschreibt der Zeitpunktwert einen Objektstatus (objState), sodass die vorherige inaktive Kopie zurückgegeben wird. Weitere Informationen enthält das folgende Beispiel: pitDate.

Eine Abfrage gibt alle Informationen zurück, die mit dem Objekt gespeichert sind, sowie die Informationen in der folgenden Tabelle.

Tabelle 17. Zurückgegebene Informationen der Abfrage des Servers

Feld	Beschreibung
`copyId`	Die Werte `copyIdHi` und `copyIdLo` geben eine aus 8 Byte bestehende Zahl an, die dieses Objekt für diesen Knoten im IBM Spectrum Protect-Speicher eindeutig kennzeichnet. Fordern Sie mithilfe dieser ID ein bestimmtes Objekt im Speicher für die Zurückschreibungs- oder Abrufverarbeitung an.

Tabelle 17. Zurückgegebene Informationen der Abfrage des Servers (Forts.)

Feld	Beschreibung
`restoreOrderExt`	Der Wert für `restoreOrderExt` gibt an, wie Objekte aus IBM Spectrum Protect-Speicher auf möglichst effiziente Art und Weise empfangen werden können. Sortieren Sie die Objekte für die Zurückschreibung anhand dieses Werts, um sicherzustellen, dass Bänder nur einmal geladen und von vorne nach hinten gelesen werden.

Sie müssen einen Teil oder alle Abfrageinformationen für die spätere Verarbeitung aufbewahren. Sie müssen die Felder `copyId` und `restoreOrderExt` aufbewahren, da sie für die eigentliche Zurückschreibungsoperation benötigt werden. Sie müssen auch alle anderen Informationen aufbewahren, die erforderlich sind, um eine Datendatei zu öffnen oder ein Ziel zu identifizieren.

Rufen Sie **dsmEndQuery** auf, um die Abfrageoperation zu beenden.

Objekte nach Zurückschreibungsreihenfolge auswählen und sortieren

Nachdem die Sicherungs- oder Archivabfrage ausgeführt wurde, muss der Anwendungsclient bestimmen, welche Objekte (sofern vorhanden) zurückgeschrieben oder abgerufen werden sollen.

Anschließend sortieren Sie die Objekte in aufsteigender Reihenfolge. Diese Sortierung ist sehr wichtig für die Leistung der Zurückschreibungsoperation. Durch das Sortieren der Objekte anhand der Felder **restoreOrderExt** wird sichergestellt, dass die Daten vom Server in der effizientesten Reihenfolge gelesen werden.

Alle Daten auf Platte werden zuerst zurückgeschrieben, gefolgt von den Daten in Datenträgerklassen, die das Laden von Datenträgern (wie z. B. Bändern) erfordern. Über das Feld **restoreOrderExt** wird außerdem sichergestellt, dass beim Lesen der Daten auf Band die korrekte Verarbeitungsreihenfolge (vom Anfang des Bands bis zum Ende des Bands) eingehalten wird.

Eine korrekte Sortierung anhand des Felds **restoreOrderExt** bedeutet, dass keine doppelten Bandladevorgänge und kein unnötiges Zurückspulen von Bändern erfolgen.

Ein Wert ungleich null im Feld **restoreOrderExt**.top korreliert mit einer eindeutigen Einheit mit seriellem Zugriff auf dem IBM Spectrum Protect-Server. Da eine Einheit mit seriellem Zugriff nur jeweils von einer einzigen Sitzung bzw. einem einzigen Mountpunkt verwendet werden kann, muss die Anwendung - wenn sie mehrere Sitzungen verwendet - sicherstellen, dass Zurückspeicherungen mit demselben Wert für **restoreOrderExt**.top nicht gleichzeitig erfolgen. Andernfalls kann die erste Sitzung zwar auf die Objekte zugreifen, die anderen Sitzungen müssen jedoch warten, bis die erste Sitzung beendet ist und die Einheit verfügbar wird.

Das nachfolgende Beispiel zeigt die Sortierung von Objekten anhand von Feldern für die **Zurückschreibungsreihenfolge**.

Abbildung 16. Objekte anhand der Felder für die Zurückschreibungsreihenfolge sortieren

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

```
}  SortOrder;               /* für Sortierung verwendete Struktur */

=====================================================================
/* Der Code für die Sortierung beginnt hier */
dsmQueryType      queryType;
qryBackupData     queryBuffer;
DataBlk           qDataBlkArea;
qryRespBackupData qbDataArea;
dsInt16_t   rc;
dsBool_t  done = bFalse;
int i = 0;
int qry_item;
SortOrder  sortorder[100]; /* Sortierung kann zur Zeit nur für bis
                              100 Einträge erfolgen. Definieren Sie
                              die Array-Größe wie erforderlich.   */
/*--------------------------------------------------------------------+
| ANMERKUNG: Stellen Sie sicher, dass die korrekten
|  Initialisierungen für queryType, queryBuffer, qDataBlkAre
|  und qbDataArea ausgeführt wurden.
|
--------------------------------------------------------------------*/

   qDataBlkArea.bufferPtf = (char*) &qbDataArea;

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

   /*---------------------------------------+
   | Rückkehrcode von dsmBeginQuery überprüfen
   +---------------------------------------*/
   while (!done)
   {
      rc = dsmGetNextQObj(dsmHandle, &qDataBlkArea);
 if ((rc == DSM_RC_MORE_DATA) ||
          (rc == DSM_RC_FINISHED))
           &&( qDataBlkArea.numBytes))
      {
         /*****************************************/
         /* Übertrag. von restoreOrderExt und objId*/
         /*****************************************/
         sortorder[i].restoreOrderExt = qbDataArea.restoreOrderExt;
         sortorder[i].objId = qbDataArea.objId;

      } /*  if ((rc == DSM_RC_MORE_DATA) || (rc == DSM_RC_FINISHED)) */
         else
      {
          done = bTrue;
          /***************************/
          /* Entspr. Aktion ausführen */
          /***************************/
      }

      i++;
      qry_item++;

   } /* while (!done) */
   rc = dsmEndQuery(dsmHandle);
  /* Rückkehrcode überprüfen */
   /*****************************************************/
   /* Sortierung des Arrays mit qsort. Nach dem Aufruf  */
   /* wird sortorder anhand des Felds restoreOrderExt   */
   /* sortiert.                                         */
   /*****************************************************/

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

 /*--------------------------------------------------------------------+
```

```
| ANMERKUNG: Stellen Sie sicher, dass sortierte Objekt-IDs
|       extrahiert werden und in jeder gewünschten Datenstruktur
|       gespeichert werden.
-------------------------------------------------------------------*/

/*-------------------------------------------------------------------+
| int SortRestoreOrder(SortOrder *a, SortOrder *b)
|
| Mit dieser Funktion werden Felder restoreOrder aus zwei
| Strukturen miteinander verglichen.
| 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);
}
```

Aufruf dsmBeginGetData starten

Nachdem Sie die zu empfangenden Objekte ausgewählt und sortiert haben, übergeben Sie sie für eine Zurückschreibungs- oder Abrufoperation an IBM Spectrum Protect. Mit dem Aufruf **dsmBeginGetData** wird eine Zurückschreibungs- oder Abrufoperationen gestartet. Die Objekte werden in der von Ihnen angeforderten Reihenfolge an den Anwendungsclient zurückgegeben.

Geben Sie die Informationen für diese beiden Parameter in diesen Aufrufen an:

mountWait
: Über diesen Parameter wird dem Server mitgeteilt, ob der Anwendungsclient auf das Laden von Offlinedatenträgern wartet, um Daten für ein Objekt abzurufen, oder ob dieses Objekt während der Verarbeitung der Zurückschreibungs- oder Abrufoperationen übersprungen werden soll.

dsmGetObjListP
: Dieser Parameter ist eine Datenstruktur, die das Feld **objId** enthält, das eine Liste aller Objekt-IDs ist, die zurückgeschrieben oder abgerufen werden. Jedes Feld **objId** ist einer Struktur **partialObjData** zugeordnet, die beschreibt, ob das gesamte über **objId** angegebene Objekt oder nur ein bestimmter Teil des Objekts abgerufen wird.

Jedes Feld **objId** hat eine Länge von acht Byte, sodass eine einzige Zurückschreibungs- oder Abrufanforderung tausende von Objekten enthalten kann. Die Anzahl Objekte, die in einem einzigen Aufruf angefordert werden können, ist auf DSM_MAX_GET_OBJ oder DSM_MAX_PARTIAL_GET_OBJ beschränkt.

Jedes zurückzuschreibende oder abzurufende Objekt empfangen

Nachdem der Aufruf **dsmBeginGetData** gesendet wurde, können Sie eine Prozedur ausführen, um jedes Objekt, das vom Server gesendet wird, zu empfangen.

Der Rückkehrcode DSM_RC_MORE_DATA gibt an, dass ein Puffer zurückgegeben wurde und Sie **dsmGetData** erneut aufrufen müssen. Überprüfen Sie **DataBlk.numBytes** auf die tatsächliche Anzahl zurückgegebener Byte.

Wenn Sie alle Daten für ein Objekt abrufen, müssen Sie einen Aufruf **dsmEndGetObj** senden. Wenn weitere Objekte empfangen werden sollen, senden Sie **dsmGetObj** erneut.

Wenn Sie den Prozess stoppen möchten, um beispielsweise alle verbleibenden Daten im Zurückschreibungsdatenstrom für alle noch nicht empfangenen Objekte zu löschen, senden Sie den Aufruf **dsmEndGetData**. Dieser Aufruf führt für die Daten, die vom Server an den Client gesendet werden, eine Flushoperation aus. Die Ausführung mithilfe dieser Methode kann jedoch einige Zeit in Anspruch nehmen. Wenn Sie eine Zurückschreibungsoperation beenden möchten, schließen Sie die Sitzung mithilfe von **dsmTerminate**.

1. Senden Sie den Aufruf **dsmGetObj**, um das Objekt, das Sie aus dem Datenstrom angefordert haben, zu identifizieren und den ersten Datenblock, der dem Objekt zugeordnet ist, abzurufen.
2. Senden Sie, falls erforderlich, weitere Aufrufe **dsmGetData**, um die verbleibenden Objektdaten abzurufen.

Beispielablaufdiagramme für Zurückschreibung und Abruf

Mithilfe eines Zustandsdiagramms und eines Ablaufdiagramms kann die Ausführung von Zurückschreibungs- oder Abrufoperationen veranschaulicht werden.

Der Pfeil, der von „In 'Objekt abrufen'" auf **dsmEndGetData** zeigt, gibt an, dass ein Aufruf **dsmEndGetData** nach einem Aufruf **dsmGetObj** oder **dsmGetData** gesendet werden kann. Dies ist möglicherweise erforderlich, wenn während des Abrufs eines Objekts aus IBM Spectrum Protect-Speicher eine Fehlerbedingung aufgetreten ist und die Operation gestoppt werden soll. Rufen Sie jedoch unter normalen Umständen zunächst **dsmEndGetObj** auf.

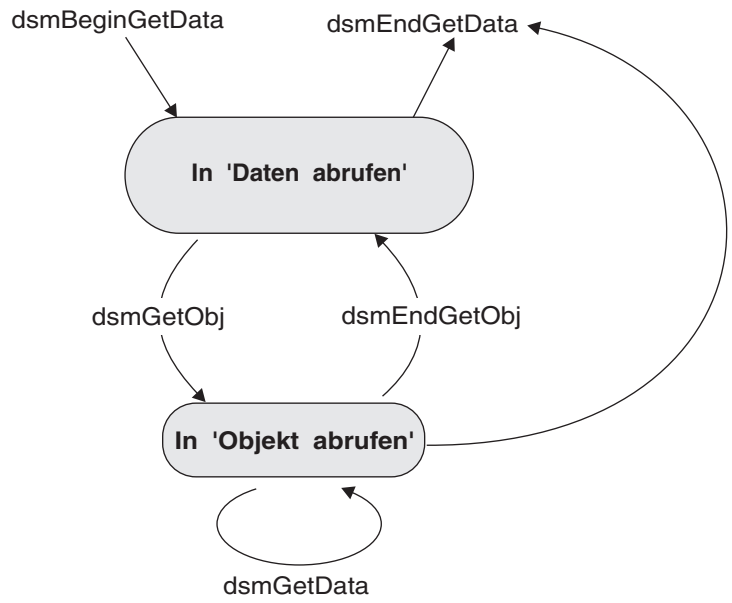


Abbildung 17. Zustandsdiagramm für Zurückschreibungs- und Abrufoperationen

Abb. 18 zeigt das Ablaufdiagramm für die Ausführung von Zurückschreibungs- oder Abrufoperationen.

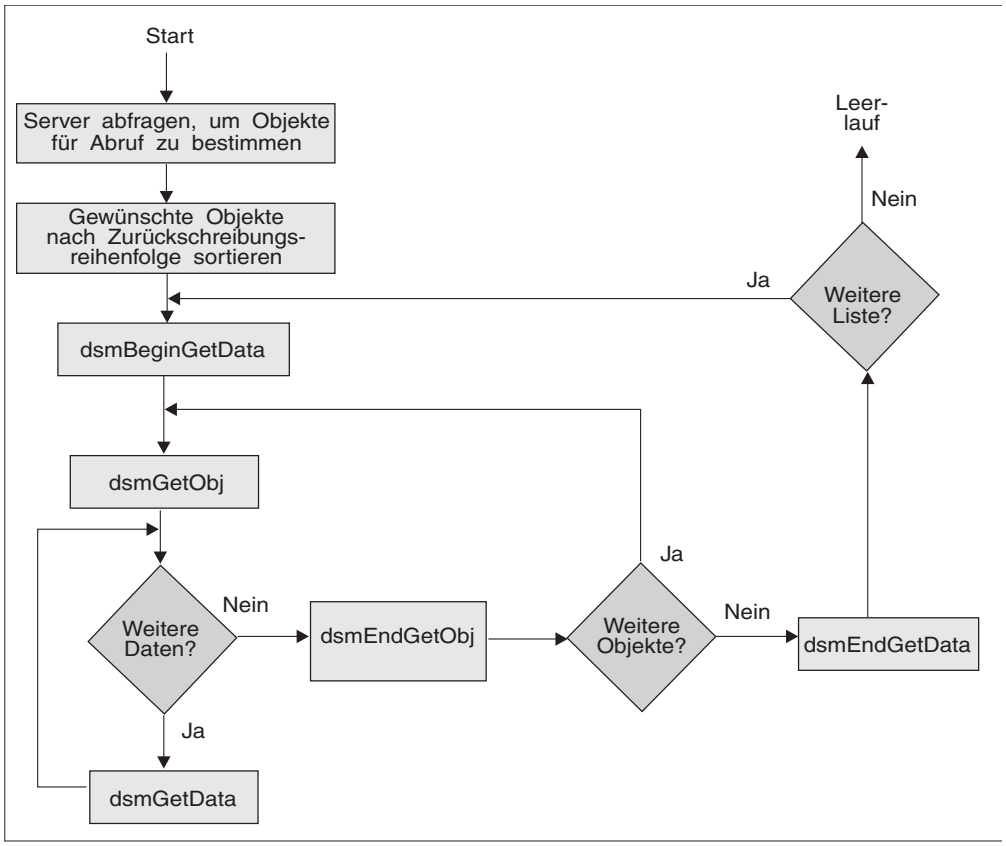


Abbildung 18. Ablaufdiagramm für Zurückschreibungs- und Abrufoperationen

Codebeispiel für das Empfangen von Daten von einem Server

Dieses Beispiel veranschaulicht die API-Funktion zum Abrufen von Daten aus IBM Spectrum Protect-Speicher

Der Funktionsaufruf **dsmBeginGetData** erscheint in einer Anweisung 'switch', sodass verschiedene Parameter abhängig davon, ob eine Zurückschreibungs- oder Abrufoperationen ausgeführt wird, aufgerufen werden können. Der Funktionsaufruf **dsmGetData** erfolgt innerhalb einer Schleife, die wiederholt Daten vom Server abruft, bis ein Flag gesetzt wird, das der Programmausführung das Verlassen der Schleife ermöglicht.

Abbildung 19. Beispiel für das Empfangen von Daten von einem Server

```
/* Aufruf dsmBeginQuery und Erstellen einer verketteten Liste mit     */
/* zurückzuschreibenden Objekten.                                     */
/* Verarbeitung dieser Liste zum Erstellen der korrekten Liste für    */
/* die Aufrufe GetData.                                               */
/* Definieren der Struktur getList zum Verweisen auf diese Liste.     */
/* Dieses Beispiel dient zur Ausführung eines Abrufs von Teilobjekten.*/
/* Um nur vollständige Objekte abzurufen, muss Folgendes definiert    */
/* werden:                                                            */
/*       getList.stVersion = dsmGetListVersion;                       */
/*       getList.partialObjData = NULL;                               */
dsmGetList getList;
getList.stVersion = dsmGetListPORVersion;   /* Strukturversion        */
getList.numObjId  = items;               /* Anzahl Einträge in Liste  */
getList.objId     = (ObjID *)rest_ibuff;
                                    /* Liste der Objekt-IDs für Zu-
                                       rückschreibung                 */
getList.partialObjData = (PartialObjData *) part_ibuff;
                                      /* Liste der Teilobjektdaten    */
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 fehlgeschlagen: ");
   rcApiOut(dsmHandle, rc);
   return rc;
}
/* Jedes Objekt aus der Liste abrufen und überprüfen, ob es auf dem */
/* Server vorhanden ist. Ist dies der Fall, Strukturen mit Objekt-  */
/* attributen für Datenprüfungen initialisieren. Anschließend       */
/* dsmGetObj aufrufen.                                                */
 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 oder dsmGetData fehlgeschlagen: ");
      rcApiOut(dsmHandle, rc);
      done = bTrue;
   }
} /* while */
rc = dsmEndGetObj(dsmHandle);
/* check rc from dsmEndGetObj */
/* check rc from dsmEndGetData */
rc = dsmEndGetData(dsmHandle);
return 0;
```

Objekte auf dem Server aktualisieren und löschen

Ihre API-Anwendungen können Objekte, die archiviert oder gesichert wurden, mithilfe des Funktionsaufrufs **dsmUpdateObj** oder **dsmUpdateObjEx** aktualisieren. Verwenden Sie beide Aufrufe nur im Sitzungsstatus und aktualisieren Sie jeweils nur ein einziges Objekt. Verwenden Sie **dsmUpdateObjEx**, um mehrere Archivierungsobjekte zu aktualisieren, die denselben Namen enthalten.

Um ein Archivierungsobjekt auszuwählen, setzen Sie den Funktionsaufruf **dsmSendType** auf **stArchive**.

- Mit **dsmUpdateObj** wird nur das letzte Archivierungsobjekt mit dem zugeordneten Namen aktualisiert.
- Mit **dsmUpdateObjEx** kann jedes archivierte Objekt durch Angabe der korrekten Objekt-ID aktualisiert werden.

Bei einem archivierten Objekt kann die Anwendung die folgenden Felder aktualisieren:

- Beschreibung
- Objektinformationen
- Eigner

Um ein Sicherungsobjekt auszuwählen, setzen Sie **dsmSendType** auf **stBackup**. Bei gesicherten Objekten wird nur die aktive Kopie aktualisiert.

Bei einem gesicherten Objekt kann die Anwendung die folgenden Felder aktualisieren:

- Verwaltungsklasse
- Objektinformationen
- Eigner

Objekte vom Server löschen

API-Anwendungen können Aufrufe ausführen, um entweder Objekte, die archiviert wurden, zu löschen oder um Objekte, die gesichert wurden, zu inaktivieren. Das Löschen archivierter Objekte ist von der Knotenberechtigung abhängig, die bei der Registrierung des Knotens durch den Administrator erteilt wurde. Administratoren können festlegen, dass Knoten Archivierungsobjekte löschen können.

Mithilfe des Funktionsaufrufs **dsmDeleteObj** können Sie archivierte Objekte löschen und Sicherungsobjekte inaktivieren. Mithilfe von **delType** wird das Sicherungsobjekt vom Server entfernt. Ein Objekt wird auf der Basis von **objID** aus der Serverdatenbank gelöscht. Ein Objekt kann nur von seinem Eigner gelöscht werden. Jede Version (aktiv oder inaktiv) eines Objekts kann gelöscht werden. Der Server gleicht die Versionen ab. Wird eine aktive Version eines Objekts gelöscht, wird die erste inaktive Version aktiv. Wird eine inaktive Version eines Objekts gelöscht, werden alle älteren Versionen vorverlegt. Der Knoten muss mit der Berechtigung **backDel** registriert sein.

Ein archiviertes Objekt wird im Speicher zum Löschen markiert, wenn das System den nächsten Objektverfallszyklus ausführt. Sobald ein archiviertes Objekt vom Server gelöscht wurde, kann es nicht mehr abgerufen werden.

Wenn Sie ein Sicherungsobjekt auf dem Server inaktivieren, wird das Objekt vom aktiven Zustand in den inaktiven Zustand versetzt. Diesen Zuständen sind verschiedene Maßnahmen für die Aufbewahrungsdauer zugeordnet, die auf der zugeordneten Verwaltungsklasse basieren.

Ähnlich wie der Aufruf **dsmSendObj** wird ein Aufruf **dsmDeleteObj** innerhalb der Grenzen einer Transaktion gesendet. Das Zustandsdiagramm in Abb. 12 auf Seite 63 zeigt einen Aufruf **dsmBeginTxn**, dem ein Aufruf **dsmDeleteObj** vorangeht und auf den ein Aufruf **dsmEndTxn** folgt.

Ereignisse protokollieren

Eine API-Anwendung kann Ereignisnachrichten an zentralen Positionen protokollieren. Die Anwendung kann die Protokollierung an den IBM Spectrum Protect-Server und/oder die lokale Maschine übertragen. Der Funktionsaufruf **dsmLogEventEx** wird in einer Sitzung ausgeführt. Um Nachrichten anzuzeigen, die auf dem Server protokolliert sind, verwenden Sie den Befehl **query actlog** über den Verwaltungsclient.

Verwenden Sie die IBM Spectrum Protect-Clientoption `errorlogretention`, um die Clientfehlerprotokolldatei zu bereinigen, wenn die Anwendung zahlreiche Clientnachrichten in das Clientprotokoll schreibt (`dsmLogType` ist entweder `logLocal` oder `logBoth`).

Weitere Informationen zu IBM Spectrum Protect-Protokollen finden Sie in der Dokumentation für den IBM Spectrum Protect-Server.

Zustandsdiagrammzusammenfassung für die IBM Spectrum Protect-API

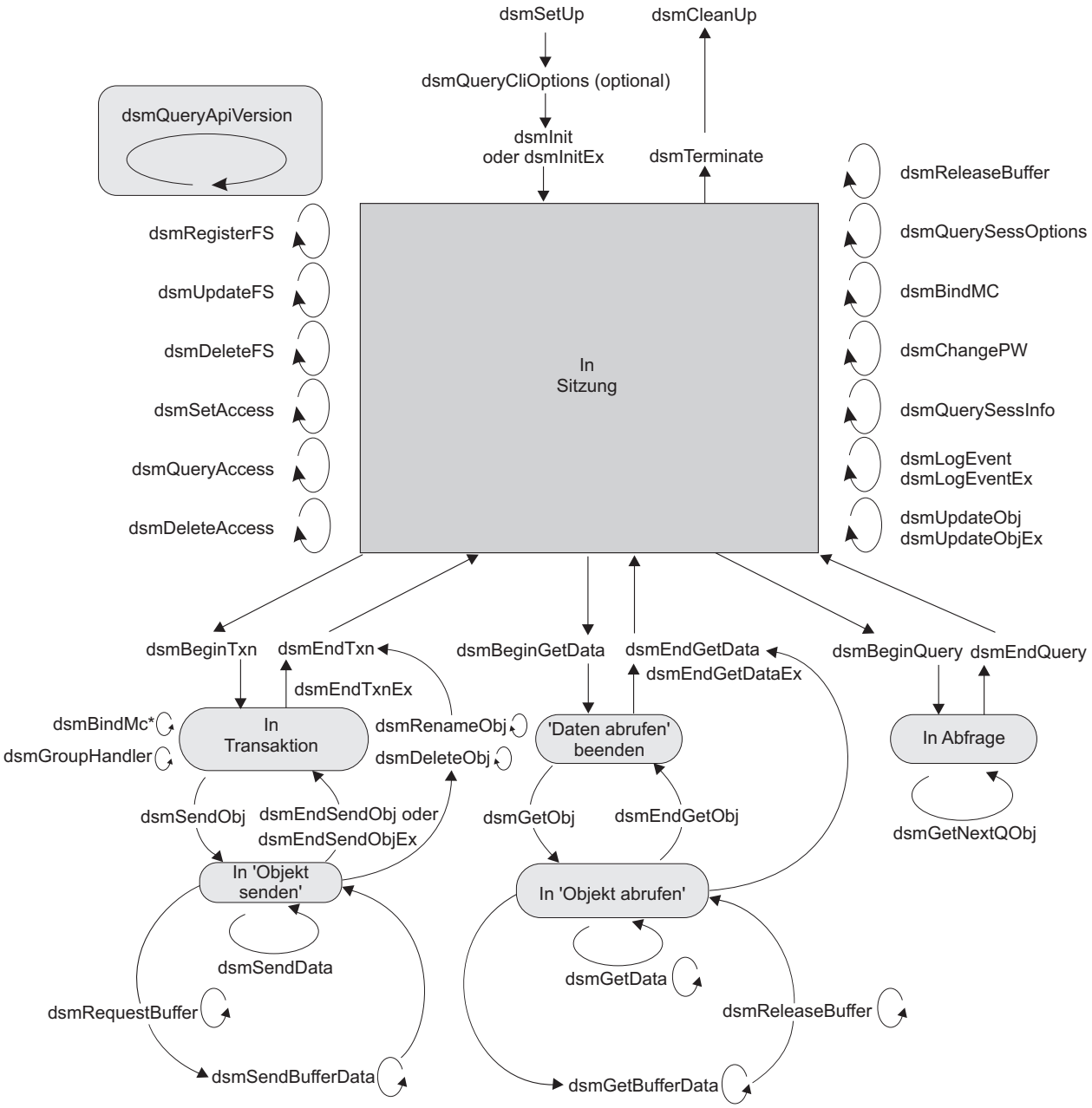
Überprüfen Sie alle Überlegungen beim Erstellen Ihrer eigenen Anwendung mit der IBM Spectrum Protect-API mithilfe dieses Zustandsdiagramms mit der Zusammenfassung einer vollständigen Anwendung.

Abb. 20 auf Seite 81 zeigt das Zustandsdiagramm für die API. Sie enthält alle zuvor angezeigten Zustandsdiagramme sowie verschiedene andere Aufrufe, die zuvor nicht angezeigt wurden.

Wichtige Punkte in diesem Diagramm umfassen:

- **dsmQueryApiVersionEx** kann jederzeit aufgerufen werden. Dieser Funktion ist kein Zustand zugeordnet. Ein Beispiel finden Sie in Abb. 1 auf Seite 14.
- Rufen Sie **dsmQueryCliOptions** nur vor einem Aufruf **dsmInitEx** auf.
- Verwalten Sie Dateibereiche mithilfe von **dsmRegisterFS**, **dsmUpdateFS** und **dsmDeleteFS**. Diese Aufrufe erfolgen innerhalb eines inaktiven Sitzungsstatus. Fragen Sie Dateibereiche mithilfe des Aufrufs **dsmBeginQuery** ab. Weitere Informationen zu Dateibereichsaufrufen finden Sie in „Dateibereiche verwalten" auf Seite 25.
- Senden Sie den Aufruf **dsmBindMC** innerhalb eines inaktiven Sitzungsstatus oder innerhalb eines Transaktionsstatus 'Objekt senden'. Siehe das Beispiel in Abb. 8 auf Seite 30.
- Senden Sie den Aufruf **dsmChangePW** innerhalb eines inaktiven Sitzungsstatus.

  Anmerkung: Wenn der Aufruf **dsmInitEx** einen Rückkehrcode zurückgibt, der angibt, dass ein Kennwort abgelaufen ist, muss der Aufruf **dsmChangePW** vor dem Start einer gültigen Sitzung erfolgen. Abb. 4 auf Seite 20 zeigt ein Beispiel für die Verwendung von **dsmChangePW**.
- Gibt ein Aufruf einen Fehler zurück, ändert sich der Zustand nicht. Wenn beispielsweise **dsmGetObj** einen Fehler zurückgibt, lautet der Zustand weiterhin "In 'Daten abrufen'" und ein Aufruf **dsmEndGetObj** stellt einen Fehler in der Aufruffolge dar.



* Kann innerhalb oder außerhalb einer Transaktion erfolgen

Abbildung 20. Zusammenfassungszustandsdiagramm für die API

Kapitel 4. Erläuterungen zur Interoperabilität

Bei der API unterscheidet man zwei Typen von Interoperabilität: Interoperabilität zwischen dem Client für Sichern/Archivieren und den API-Anwendungen sowie Interoperabilität zwischen verschiedenen Betriebssystemen.

Interoperabilität des Clients für Sichern/Archivieren

Die Befehlszeile für Sichern/Archivieren kann auf API-Objekte zugreifen, um begrenzte Interoperabilität zur Verfügung zu stellen. Das Anzeigen von API-Objekten und der Zugriff auf API-Objekte ist nur über den Befehlszeilenclient für Sichern/Archivieren und nicht über eine der grafischen Benutzerschnittstellen möglich. Der Befehlszeilenclient für Sichern/Archivieren kann nur den Dateiinhalt uns sonst nichts zurückschreiben. Daher sollten Sie ihn nur für eine Wiederherstellungsoperation verwenden.

Die folgenden Befehlszeilenaktionen sind verfügbar:
- Delete archive
- Delete filespace
- Query
- Restore
- Retrieve
- Set access

Die Pfadinformationen sind tatsächliche Verzeichnisse für Objekte des Clients für Sichern/Archivieren. Die Pfadinformationen der API-Objekte dagegen haben möglicherweise keine Beziehung zu vorhandenen Verzeichnissen, der Pfad kann völlig frei erfunden sein. Die Interoperabilität ändert diesen Aspekt dieser Objekttypen nicht. Um dieses Feature erfolgreich einsetzen zu können, halten Sie die Bedingungen und Konventionen ein.

Anmerkungen:

1. Es gibt keine Interoperabilität zwischen dem Client für Sichern/Archivieren und API-Objekten, die auf einem Aufbewahrungsschutzserver gespeichert sind.
2. Sie können nicht mit den grafischen Benutzerschnittstellen des Clients für Sichern/Archivieren auf Dateien zugreifen, die mit dem API-Client gespeichert wurden. Sie können auf diese Dateien nur über die Befehlszeile zugreifen.

Benennung von API-Objekten

Erstellen Sie eine konsistente Namenskonvention für API-Objektnamen. Die Namenskonvention muss den Dateibereichsnamen, das übergeordnete Qualifikationsmerkmal und das untergeordnete Qualifikationsmerkmal berücksichtigen. Der Dateibereichsname und die übergeordneten Qualifikationsmerkmale können sich auf tatsächliche Verzeichnisnamen beziehen. Jeder Objektname kann aus mehreren Verzeichnisnamen bestehen, die für das untergeordnete Qualifikationsmerkmal gelten.

Am einfachsten ist es, als untergeordnetes Qualifikationsmerkmal den Namen des Objekts ohne Verzeichnisinformationen als Präfix zu verwenden. Weitere Informationen finden Sie in „Objektnamen und -IDs" auf Seite 22.

Dateibereichsnamen müssen vollständig qualifiziert sein, wenn von der API oder der Befehlszeile für Sichern/Archivieren auf sie Bezug genommen wird. Registrieren Sie beispielsweise unter einem Betriebssystem UNIX oder Linux die folgenden Dateibereiche:

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

Wenn Sie auf `/a` Bezug nehmen, werden Objekte angezeigt, die sich nur auf Dateibereich `/a` beziehen. Wenn Objekte angezeigt werden sollen, die sich auf `/a/b` beziehen, müssen Sie `/a/b` als Dateibereichsnamen angeben.

Wenn Sie, nachdem Sie beide Dateibereiche registriert haben, Objekt `b` in Dateibereich `/a` sichern, werden bei einer Abfrage für `/a/b` weiterhin Objekte angezeigt, die sich nur auf Dateibereich `/a/b` beziehen.

Eine Ausnahme von dieser Einschränkung sind Dateibereichsreferenzen, wenn Sie versuchen, Dateibereiche mit der API abzufragen oder zu löschen. In beiden Fällen müssen die Dateibereichsnamen nicht vollständig qualifiziert sein, wenn ein Platzhalterzeichen verwendet wird. Beispielsweise bezieht sich `/a*` sowohl auf `/a` als auch auf `/a/b`.

Tipp: Wenn Interoperabilität für Sie wichtig ist, geben Sie keine Dateibereichsnamen an, die sich überschneiden.

Schließen Sie bei Windows-Systemen die Dateibereichsnamen für API-Objekte in geschweifte Klammern { } ein, wenn der Zugriff auf die Objekte über die Befehlszeilenschnittstelle für Sichern/Archivieren erfolgt. Windows-Betriebssysteme setzen Dateibereichsnamen automatisch in Großbuchstaben um, wenn die Namen registriert werden oder auf sie Bezug genommen wird. Diese automatische Funktion gibt es jedoch nicht für die restliche Objektnamensspezifikation. Wird vollständige Interoperabilität gewünscht, geben Sie beim Sichern von API-Objekten das übergeordnete Qualifikationsmerkmal und das untergeordnete Qualifikationsmerkmal in der Anwendung in Großbuchstaben an. Wenn Ihre Anwendung übergeordnete Qualifikationsmerkmale (Verzeichnisnamen) und untergeordnete Qualifikationsmerkmale (Dateinamen) vor dem Senden von Objekten an den Server nicht in Großbuchstaben umsetzt, können Sie über den Client für Sichern/Archivieren nicht direkt über den Namen auf die Objekte zugreifen.

Wenn beispielsweise ein Objekt auf dem Server als `{"Dateibereichsname"}\TEST\MYDIRNAME\file.txt` gespeichert ist, können Sie das Objekt `file.txt` nicht direkt zurückschreiben oder abfragen, weil Ihre Anwendung den Dateinamen nicht in Großbuchstaben umgesetzt hat, bevor die Datei auf den Server kopiert wurde. Die einzige Möglichkeit, mit diesen Objekten zu arbeiten, besteht in der Verwendung von Platzhalterzeichen. Beispielsweise muss ein Benutzer des Clients für Sichern/Archivieren bei einer Abfrage von `\TEST\MYDIRNAME\file.txt` Platzhalterzeichen für alle Teile des Objektnamens verwenden, die nicht in Großbuchstaben umgesetzt wurden, bevor sie an den Server gesendet wurden. Der folgende Befehl muss verwendet werden, um diese Datei `file.txt` abzufragen:

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

Wenn auch noch andere Qualifikationsmerkmale in Kleinschreibung gespeichert wurden, müssen auch sie unter Verwendung von Platzhalterzeichen abgefragt werden. Verwenden Sie beispielsweise zum Abfragen eines Objekts, das als `{"Dateibereichsname"}\TEST\mydirname\file.txt` gespeichert wurde, den folgenden Befehl:

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

Die nachfolgenden Beispiele veranschaulichen diese Konzepte. Weder in Windows- noch in UNIX- oder Linux-Umgebungen müssen Sie das vollständige übergeordnete oder untergeordnete Qualifikationsmerkmal angeben. Wenn Sie nicht das vollständige Qualifikationsmerkmal angeben, müssen Sie allerdings das Platzhalterzeichen verwenden.

Plattform	Beispiel
Windows	Um alle gesicherten Dateien im Dateibereich `MYFS` abzufragen, geben Sie die folgende Zeichenfolge ein: `dsmc q ba "{MYFS}\*\*"` Sie müssen für jedes übergeordnete und untergeordnete Qualifikationsmerkmal mindestens 1 Stern (*) verwenden.
UNIX oder Linux	Um alle gesicherten Dateien im Dateibereich `/A` abzufragen, geben Sie die folgende Zeichenfolge ein: `dsmc q ba "/A/*/*"` Sie müssen für jedes übergeordnete und untergeordnete Qualifikationsmerkmal mindestens 1 Stern (*) verwenden.

Für die API verwendbare Befehle des Clients für Sichern/Archivieren

Sie können eine Untergruppe der Befehle des Clients für Sichern/Archivieren in einer Anwendung verwenden. Sie können beispielsweise Objekte, deren Eigner andere Benutzer sind, auf demselben Knoten oder auf einem anderen Knoten anzeigen und verwalten.

Gehen Sie wie folgt vor, um Objekte, deren Eigner andere Benutzer sind, auf demselben Knoten oder auf einem anderen Knoten anzuzeigen und zu verwalten:

1. Erteilen Sie Zugriff mit dem Befehl **set access**.
2. Geben Sie den Eigner und den Knoten an. Verwenden Sie die Optionen *fromowner* und *fromnode* über die Befehlszeile für Sichern/Archivieren, um den Eigner und den Knoten anzugeben. Beispiel:

```
dsmc q ba "/A/*/*" -fromowner=anderer_Eigner -fromnode=anderer_Knoten
```

Tabelle 18 enthält die Befehle, die Sie für API-Objekte verwenden können.

Tabelle 18. Für API-Objekte verwendbare Befehle des Clients für Sichern/Archivieren

Befehl	Beschreibung
Delete Archive	Archivierte Dateien, deren Eigner der aktuelle Benutzer ist, können gelöscht werden. Die Einstellungen des Befehls set access haben keinen Einfluss auf diesen Befehl.
Delete Filespace	Der Befehl **delete filespace** wirkt sich auf API-Objekte aus.

Tabelle 18. Für API-Objekte verwendbare Befehle des Clients für Sichern/Archivieren (Forts.)

Befehl	Beschreibung
Query	Über die Befehlszeile für Sichern/Archivieren können Sie gesicherte und archivierte API-Objekte und Objekte, deren Eigner andere Benutzer sind oder die sich auf anderen Knoten befinden, abfragen. Informationen zum Abfragen von API-Objekten finden Sie in „Benennung von API-Objekten" auf Seite 83. Verwenden Sie die vorhandene Option *–fromowner*, um Objekte abzufragen, deren Eigner ein anderer Benutzer ist und für die die set access-Berechtigung erteilt wurde. Verwenden Sie die vorhandene Option *–fromnode*, um Objekte abzufragen, die sich auf einem anderen Knoten befinden und für die die set access-Berechtigung erteilt wurde. Weitere Informationen finden Sie in „**dsmInitEx**" auf Seite 127.
Restore **Retrieve**	**Anmerkung:** Verwenden Sie diese Befehle nur in Ausnahmesituationen. Mit dem anwendungsverwalteten Schlüssel verschlüsselte API-Objekte können zurückgeschrieben oder abgerufen werden, wenn der Verschlüsselungsschlüssel bekannt oder in der Kennwortdatei bzw. im Registry gespeichert ist. Mit der transparenten Verschlüsselung verschlüsselte API-Objekte können nicht mit dem Client für Sichern/Archivieren zurückgeschrieben oder abgerufen werden. Diese Befehle geben Daten als Bitdateien zurück, die mit Standarddateiattributen erstellt werden. Sie können API-Objekte, deren Eigner andere Benutzer sind oder die von einem anderen Knoten stammen, zurückschreiben oder abrufen. Der Befehl set access legt fest, welche Objekte in Frage kommen.
Set Access	Mit dem Befehl **set access** können Benutzer API-Objekte verwalten, deren Eigner ein anderer Benutzer ist oder die von einem anderen Knoten stammen.

Interoperabilität zwischen Betriebssystemen

Die IBM Spectrum Protect-API unterstützt die plattformübergreifende Interoperabilität. Anwendungen auf einem UNIX- oder Linux-System können Operationen für Dateibereiche und Objekte ausführen, die von einem Windows-System gesichert werden. Analog kann ein Windows-System Operationen für Dateibereiche und Objekte ausführen, die auf einem UNIX- oder Linux-System gesichert werden.

Standardmäßig sind die Namen von Objekten aus einem UNIX-System mit den Namen von Objekten aus anderen UNIX-Systemen kompatibel. Standardmäßig sind die Namen von Objekten aus Windows-Systemen nicht mit den Namen von Objekten aus UNIX-Systemen kompatibel. Die Benennung von Objekten in IBM Spectrum Protect-Dateibereichen wird durch mehrere Parameter gesteuert. Wenn Sie eine Anwendung ordnungsgemäß konfigurieren, können die Namen von Objekten verwendet werden, die sowohl auf Windows-Systemen als auch auf UNIX-Systemen ausgeführt werden. Verwenden Sie für die Sicherung und Zurückschreibung von Objekten dieselben Parameter.

Einschränkung: Eine Windows-Anwendung, die Unicode verwendet, erstellt einen Dateibereich, der nicht mit Anwendungen kompatibel ist, die auf UNIX-Systemen ausgeführt werden.

Führen Sie die folgenden Konfigurationstasks aus, um Interoperabilität zu erzielen:

1. Erstellen Sie eine konsistente Namenskonvention. Wählen Sie ein Zeichen für den Verzeichnisbegrenzer aus, zum Beispiel einen normalen Schrägstrich (/)

oder einen umgekehrten Schrägstrich (\). Das als Verzeichnisbegrenzer verwendete Zeichen muss vor dem Dateibereichsnamen, vor dem übergeordneten Qualifikationsmerkmal bzw. vor dem untergeordneten Qualifikationsmerkmal stehen.

2. Wenn Sie **dsmInitEx** aufrufen, setzen Sie den Wert für das Feld **dirDelimiter** auf das als Verzeichnisbegrenzer zu verwendende Zeichen und setzen Sie **bCrossPlatform** auf **bTrue**.
3. Setzen Sie das Flag **useUnicode** auf **bFalse**, wenn Sie die IBM Spectrum Protect-Schnittstelle verwenden. Unicode-Dateinamen sind nicht mit Nicht-Unicode-Dateinamen kompatibel.

Mehrere Knoten mit Clientknoten-Proxy-Unterstützung sichern

Sicherungen mehrerer Knoten, die Speicher gemeinsam nutzen, können auf dem IBM Spectrum Protect-Server auf einen gemeinsamen Zielknotennamen konsolidiert werden. Diese Methode ist hilfreich, wenn sich das System, das die Sicherung ausführt, im Laufe der Zeit ändern kann, beispielsweise im Fall eines Clusters. Sie können auch die Option asnodename zum Zurückschreiben von Daten von einem anderen System verwenden als demjenigen, das die Sicherung ausgeführt hat.

Verwenden Sie die Option asnodename in der **dsmInitEx**-Optionszeichenfolge, um Daten unter dem Zielknotennamen auf dem IBM Spectrum Protect-Server zu sichern, zu archivieren, zurückzuschreiben, abzurufen, abzufragen oder zu löschen. Sie können die Option asnodename auch in der Datei dsm.opt oder dsm.sys angeben.

Einschränkung: Verwenden Sie Zielknoten nicht als traditionelle Knoten, besonders dann nicht, wenn Sie Ihre Dateien verschlüsseln, bevor Sie auf dem Server sichern.

Führen Sie die folgenden Schritte aus, um diese Option zu aktivieren:

1. Installieren Sie den API-Client auf allen Knoten in einer gemeinsam genutzten Datenumgebung.
2. Falls noch nicht geschehen, registrieren Sie jeden Knoten beim IBM Spectrum Protect-Server. Registrieren Sie den gemeinsamen Zielknotennamen, der von allen in Ihrer gemeinsam genutzten Datenumgebung verwendeten Agentenknoten gemeinsam genutzt werden soll.
3. Registrieren Sie jeden Agentenknoten in der gemeinsam genutzten Datenumgebung beim Server. Der Agentenknotenname wird für die Authentifizierung verwendet. Es werden keine Daten unter Verwendung des Agentenknotennamens gespeichert, wenn die Option asnodename verwendet wird.
4. Bitten Sie Ihren Administrator, allen Knoten in der gemeinsam genutzten Umgebung mit dem Befehl **grant proxynode** Proxy-Berechtigung zu erteilen, damit sie auf den Zielknotennamen auf dem IBM Spectrum Protect-Server zugreifen können.
5. Verwenden Sie den Verwaltungsclientbefehl **query proxynode**, um die Clientknoten anzuzeigen, die die Berechtigung zum Ausführen von Clientoperationen für einen anderen Knoten haben. Diese Berechtigung wird mit dem Befehl **grant proxynode** erteilt. Oder verwenden Sie den Befehl **dsmQuery** mit dem Abfragetyp **qtProxyNodeAuth**, um die Knoten anzuzeigen, an die der vorliegende Knoten weiterleiten kann.
6. Wenn die Anwendung mit Benutzerverschlüsselung für Daten arbeitet (nicht TSMENCRKEY), müssen Sie sicherstellen, dass alle Knoten denselben Verschlüsse-

lungsschlüssel verwenden. Sie müssen denselben Verschlüsselungsschlüssel für alle Dateien verwenden, die in der Umgebung mit gemeinsam genutzten Knoten gesichert werden.

Zugehörige Tasks:

Daten mit Proxy-Unterstützung für Clientknoten sichern (UNIX- und Linux-Systeme)

Daten mit Proxy-Unterstützung für Clientknoten sichern (Windows-Systeme)

Kapitel 5. Verwendung der API mit Unicode

Die IBM Spectrum Protect-API unterstützt Unicode UCS-2, eine Doppelbyte-Codepage mit fester Länge, die Codepunkte für alle bekannten Codepages, wie z. B. Japanisch, Chinesisch oder Deutsch, enthält. Sie unterstützt bis zu 65.535 eindeutige Codepunkte.

Einschränkung: Diese Funktion ist nur unter Windows verfügbar.

Bei Verwendung von Unicode kann Ihre Anwendung Dateinamen in jedem Zeichensatz auf derselben Maschine sichern und zurückschreiben. Beispielsweise können Sie auf einer Maschine, auf der Englisch installiert ist, Dateinamen in jeder anderen Sprachencodepage sichern und zurückschreiben.

Situationen für die Verwendung von Unicode

Sie können Ihre Anwendung, die mehrere Sprachen unterstützt, vereinfachen, indem Sie eine Unicode-Anwendung schreiben und die Vorteile der Unicode-Schnittstelle von IBM Spectrum Protect nutzen.

Verwenden Sie die Unicode-Schnittstelle von IBM Spectrum Protect, wenn eine der folgenden Bedingungen erfüllt ist:

- Ihre Anwendung wurde bereits für Unicode kompiliert und wurde in einen Mehrbytezeichensatz (MBCS) konvertiert, bevor die IBM Spectrum Protect-API aufgerufen wurde.
- Sie schreiben eine neue Anwendung und Ihre Anwendung soll Unicode unterstützen.
- Ihre Anwendung verwendet eine Zeichenfolge, die von einem Betriebssystem oder einer anderen Anwendung, das bzw. die Unicode verwendet, an die Anwendung übergeben wurde.

Wenn Unicode nicht benötigt wird, ist es nicht erforderlich, die Anwendung erneut zu kompilieren.

Die API verwendet weiterhin die dsm-Schnittstelle. Das API-SDK enthält die Musterprogramme callmtu1.c und callmtu2.c, die die Verwendung der Unicode-API veranschaulichen. Kompilieren Sie diese Programme mithilfe von **makemtu**.

Unicode konfigurieren

Um Unicode konfigurieren und verwenden zu können, müssen Sie eine bestimmte Prozedur ausführen, damit die API einen Unicode-Dateibereich auf dem Server registriert und alle Dateinamen in diesem Dateibereich Unicode-Zeichenfolgen werden.

Einschränkung: Sie können Unicode-Dateinamen und andere Dateinamen nicht in demselben Dateibereich speichern.

1. Kompilieren Sie den Code mit dem Flag `-DUNICODE`.
2. Alle Zeichenfolgen in Ihrer Anwendung müssen wchar-Zeichenfolgen sein.
3. Folgen Sie den Strukturen in der Datei tsmapitd.h und den Funktionsdefinitionen in der Datei tsmapifp.h für Aufrufe an die API.

4. Setzen Sie das Flag *useUnicode* im Funktionsaufruf **tsmInitEx** auf *bTrue*. Jeder neue Dateibereich wird als Unicode-Dateibereich registriert.

Wenn Sie Daten an bereits registrierte Nicht-Unicode-Dateibereiche senden, sendet die API Dateinamen weiterhin als Nicht-Unicode. Benennen Sie die alten Dateibereiche auf dem Server in fsname_old um und starten Sie einen neuen Unicode-Dateibereich für neue Daten. Die API schreibt Nicht-Unicode-Daten aus den alten Dateibereichen zurück. Verwenden Sie das Feld **bIsUnicode** in der Struktur **tsmQryRespFSData**, die in einem Abfragedateibereich zurückgegeben wird, um festzustellen, ob es sich um einen Unicode-Dateibereich handelt.

Jeder Funktionsaufruf **dsmXXX** verfügt über einen entsprechenden Funktionsaufruf **tsmXXX**. Der Unterschied zwischen beiden sind die verwendeten Strukturen. Alle Strukturen des Funktionsaufrufs **tsmXXX** haben den Typ dsChar_t für Zeichenfolgewerte, wenn sie mit dem UNICODE-Flag kompiliert werden. dsChar_r wird wchar zugeordnet. Es gibt keinen anderen Unterschied zwischen diesen Schnittstellen.

Einschränkung: Verwenden Sie entweder die eine Schnittstelle oder die andere. Kombinieren Sie nicht die Schnittstellen des Funktionsaufrufs **dsmXXX** und des Funktionsaufrufs **tsmXXX**. Stellen Sie sicher, dass Sie die IBM Spectrum Protect-Strukturen und IBM Spectrum Protect-Versionsdefinitionen verwenden.

Einige Konstanten werden weiterhin in der Datei dsmapitd.h definiert, daher benötigen Sie beim Kompilieren sowohl die Datei dsmapitd.h als auch die Datei tsmapitd.h.

Sie können die IBM Spectrum Protect-Schnittstelle auf anderen Betriebssystemen verwenden, z. B. UNIX oder Linux. Auf diesen Betriebssystemen wird der Typ dsChar_t jedoch char zugeordnet, weil Unicode nur unter Windows unterstützt wird. Sie können nur eine einzige Variante der Anwendung schreiben und auf mehreren Betriebssystemen mithilfe der IBM Spectrum Protect-Schnittstelle kompilieren. Verwenden Sie die IBM Spectrum Protect-Schnittstelle, wenn Sie eine neue Anwendung schreiben.

Wenn Sie ein Upgrade einer vorhandenen Anwendung durchführen:
1. Konvertieren Sie die Strukturen und Aufrufe des Funktionsaufrufs **dsmXXX** in die IBM Spectrum Protect-Schnittstelle
2. Migrieren Sie vorhandene Dateibereiche
3. Sichern Sie neue Dateibereiche mit der Einstellung *true* für das Flag *useUnicode*

Anmerkung: Sobald Sie mithilfe eines Clients mit Unicode-Unterstützung auf einen Knoten zugegriffen haben, können Sie mit einer älteren Version der API oder mit einer API aus einem anderen Betriebssystem keine Verbindung zu demselben Knoten herstellen. Wenn Ihre Anwendung plattformübergreifende Funktionalität verwendet, dürfen Sie das Flag `Unicode` nicht verwenden. Es gibt keine plattformübergreifende Unterstützung zwischen Unicode- und Nicht-Unicode-Betriebssystemen.

Wenn Sie das Flag *useUnicode* aktivieren, werden alle Zeichenfolgestrukturen als Unicode-Zeichenfolgen behandelt. Auf dem Server sind nur die folgenden Felder wahre Unicode-Felder:
- Dateibereichsname
- Obere Ebene
- Untere Ebene
- Archivierungsbeschreibung

Alle übrigen Felder werden in MBCS in der lokalen Codepage konvertiert, bevor sie an den Server gesendet werden. Felder wie Knotenname sind wchar-Zeichenfolgen. Sie müssen in der aktuellen Ländereinstellung gültig sein. Sie können beispielsweise auf einem japanischen System Dateien mit chinesischen Namen sichern, der Knotenname muss jedoch eine gültige japanische Zeichenfolge sein. Die Optionsdatei bleibt in der aktuellen Codepage. Wenn Sie eine Einschluss-/Ausschlussliste in Unicode erstellen müssen, verwenden Sie die Option *inclexcl* mit einem Dateinamen und erstellen Sie eine Unicode-Datei, die Unicode-Muster enthält.

Zugehörige Verweise:

Option inclexcl

Kapitel 6. API-Funktionsaufrufe

In Tabelle 19 wird eine alphabetische Liste der API-Funktionsaufrufe mit einer Kurzbeschreibung und der Angabe, wo detaillierte Informationen zu dem Funktionsaufruf gefunden werden können, bereitgestellt. Die Liste umfasst Folgendes:

Element	Beschreibung
Zweck	Beschreibt den Funktionsaufruf.
Syntax	Enthält den tatsächlichen C-Code für den Funktionsaufruf. Dieser Code wurde aus der UNIX- oder Linux-Version der Headerdatei dsmapifp.h kopiert. . Siehe Anhang C, „Quellendatei mit den API-Funktionsdefinitionen", auf Seite 215. Diese Datei unterscheidet sich geringfügig von der Datei auf anderen Betriebssystemen. Anwendungsprogrammierer für andere Betriebssysteme müssen in Bezug auf die exakte Syntax der API-Definitionen Ihre Version der Headerdatei dsmapifp.h überprüfen.
Parameter	Beschreibt jeden Parameter in dem Funktionsaufruf und kennzeichnet ihn abhängig von seiner Verwendung als Eingabe (E) oder Ausgabe (A). Einige Parameter sind sowohl als Eingabe als auch als Ausgabe (E/A) gekennzeichnet. Die Datentypen, auf die in diesem Abschnitt Bezug genommen wird, sind in der Headerdatei dsmapitd.h definiert. Siehe Anhang B, „Quellendateien für API-Typdefinitionen", auf Seite 171.
Rückkehrcodes	Enthält eine Liste der Rückkehrcodes, die für den Funktionsaufruf spezifisch sind. Allgemeine Systemfehler wie Kommunikationsfehler, Serverfehler oder Benutzerfehler, die in jedem Aufruf auftreten können, sind nicht aufgelistet. Die Rückkehrcodes sind in der Headerdatei dsmrc.h definiert. Siehe Anhang A, „Quellendatei mit API-Rückkehrcodes: `dsmrc.h`", auf Seite 159.

Tabelle 19. API-Funktionsaufrufe

Funktionsaufruf und Position	Beschreibung
„dsmBeginGetData" auf Seite 96	Startet eine Zurückschreibungs- oder Abrufoperation für eine Liste mit Objekten im Speicher.
„dsmBeginQuery" auf Seite 97	Startet für IBM Spectrum Protect eine Anforderung zum Abfragen von Informationen.
„dsmBeginTxn" auf Seite 102	Startet eine oder mehrere Transaktionen, die eine vollständige Aktion starten. Entweder werden alle Aktionen erfolgreich ausgeführt oder keine wird erfolgreich ausgeführt.
„dsmBindMC" auf Seite 103	Ordnet eine Verwaltungsklasse dem Objekt zu, das übergeben wird, bzw. bindet eine Verwaltungsklasse an dieses Objekt.
„dsmChangePW" auf Seite 105	Ändert ein IBM Spectrum Protect-Kennwort.
„dsmCleanUp" auf Seite 106	Dieser Aufruf wird verwendet, wenn **`dsmSetUp`** aufgerufen wurde.
„dsmDeleteAccess" auf Seite 106	Löscht aktuelle Berechtigungsregeln für Sicherungsversionen oder archivierte Kopien Ihrer Objekte.
„dsmDeleteFS" auf Seite 107	Löscht einen Dateibereich aus dem Speicher.
„dsmDeleteObj" auf Seite 108	Inaktiviert Sicherungsobjekte oder löscht Archivierungsobjekte im Speicher.

Tabelle 19. API-Funktionsaufrufe (Forts.)

Funktionsaufruf und Position	Beschreibung
„dsmEndGetData" auf Seite 109	Beendet eine **dsmBeginGetData**-Sitzung, mit der Objekte aus dem Speicher abgerufen werden.
„dsmEndGetDataEx" auf Seite 110	Gibt die Gesamtanzahl gesendeter LAN-unabhängiger Byte an.
„dsmEndGetObj" auf Seite 110	Beendet eine **dsmGetObj**-Sitzung, mit der Daten für ein bestimmtes Objekt abgerufen werden.
„dsmEndQuery" auf Seite 111	Gibt das Ende einer Aktion **dsmBeginQuery** an.
„dsmEndSendObj" auf Seite 111	Gibt das Ende der Daten an, die in den Speicher gesendet werden.
„dsmEndSendObjEx" auf Seite 112	Gibt Komprimierungsangaben und die Anzahl gesendeter Byte an.
„dsmEndTxn" auf Seite 113	Beendet eine IBM Spectrum Protect-Transaktion.
„dsmEndTxnEx" auf Seite 114	Stellt Informationen zur Objekt-ID des Gruppenleiters für die Verwendung im Aufruf **dsmGroupHandlerfunction** bereit.
„dsmGetData" auf Seite 116	Ruft einen Bytestrom mit Daten aus IBM Spectrum Protect ab und stellt ihn in den Puffer des Aufrufenden.
„dsmGetBufferData" auf Seite 117	Ruft einen von IBM Spectrum Protect zugeordneten Datenpuffer aus dem IBM Spectrum Protect-Server ab.
„dsmGetNextQObj" auf Seite 118	Ruft die nächste Abfrageantwort aus einem vorherigen Aufruf **dsmBeginQuery** ab und stellt ihn in den Puffer des Aufrufenden.
„dsmGetObj" auf Seite 121	Ruft die angeforderten Objektdaten aus dem Datenstrom ab und stellt sie in den Puffer des Aufrufenden.
„dsmGroupHandler" auf Seite 122	Führt abhängig von der bereitgestellten Eingabe eine Aktion für eine logische Dateigruppe aus.
„dsmInit" auf Seite 123	Startet eine API-Sitzung und verbindet den Client mit Speicher.
„**dsmInitEx**" auf Seite 127	Startet eine API-Sitzung unter Verwendung der zusätzlichen Parameter, die eine erweiterte Überprüfung ermöglichen.
„dsmLogEvent" auf Seite 131	Protokolliert eine Benutzernachricht in der Serverprotokolldatei und/oder im lokalen Fehlerprotokoll.
„dsmLogEventEx" auf Seite 132	Protokolliert eine Benutzernachricht in der Serverprotokolldatei und/oder im lokalen Fehlerprotokoll.
„dsmQueryAccess" auf Seite 133	Fragt den Server nach allen Zugriffsberechtigungsregeln für Sicherungsversionen oder archivierte Kopien Ihrer Objekte ab.
„dsmQueryApiVersion" auf Seite 134	Führt eine Abfrageanforderung für die API-Bibliotheksversion aus, auf die der Anwendungsclient zugreift.
„dsmQueryApiVersionEx" auf Seite 135	Führt eine Abfrageanforderung für die API-Bibliotheksversion aus, auf die der Anwendungsclient zugreift.
„dsmQueryCliOptions" auf Seite 135	Fragt wichtige Optionswerte in der Benutzeroptionsdatei ab.
„dsmQuerySessInfo" auf Seite 136	Startet eine Anforderung zum Abfragen von Informationen an IBM Spectrum Protect, die sich auf die Operation der angegebenen Sitzung in **dsmHandle** beziehen.

Tabelle 19. API-Funktionsaufrufe (Forts.)

Funktionsaufruf und Position	Beschreibung
„dsmQuerySessOptions" auf Seite 137	Fragt wichtige Optionswerte ab, die in der angegebenen Sitzung in **dsmHandle** gültig sind.
„dsmRCMsg" auf Seite 138	Ruft den Nachrichtentext ab, der einem API-Rückkehrcode zugeordnet ist.
„dsmRegisterFS" auf Seite 139	Registriert einen neuen Dateibereich beim Server.
„dsmReleaseBuffer" auf Seite 140	Gibt einen von IBM Spectrum Protect zugeordneten Puffer zurück.
„dsmRenameObj" auf Seite 141	Benennt den übergeordneten oder untergeordneten Objektnamen um.
„dsmRequestBuffer" auf Seite 142	Ruft einen von IBM Spectrum Protect zugeordneten Puffer für die Pufferkopieneliminierung ab.
„dsmRetentionEvent" auf Seite 143	Sendet im Rahmen einer Aufbewahrungsereignisoperation, die für die in der Liste angegebenen Objekte ausgeführt werden soll, eine Liste mit Objekt-IDs an den Server.
„dsmSendBufferData" auf Seite 144	Sendet Daten aus einem von IBM Spectrum Protect zugeordneten Puffer.
„dsmSendData" auf Seite 145	Sendet einen Bytestrom mit Daten über einen Puffer an IBM Spectrum Protect.
„dsmSendObj" auf Seite 146	Startet eine Anforderung zum Senden eines einzelnen Objekts in Speicher.
„dsmSetAccess" auf Seite 150	Erteilt anderen Benutzern oder Knoten Zugriff auf Sicherungsversionen oder archivierte Kopien Ihrer Objekte, Zugriff auf alle Ihre Objekte oder Zugriff auf eine ausgewählte Gruppe von Objekten.
„dsmSetUp" auf Seite 151	Überschreibt Umgebungsvariablenwerte.
„dsmTerminate" auf Seite 153	Beendet eine Sitzung mit dem Server und bereinigt die IBM Spectrum Protect-Umgebung.
„dsmUpdateFS" auf Seite 153	Aktualisiert einen Dateibereich im Speicher.
„dsmUpdateObj" auf Seite 154	Aktualisiert die objInfo-Informationen, die einem aktiven Sicherungsobjekt zugeordnet sind, das sich bereits auf dem Server befindet, oder aktualisiert archivierte Objekte.
„dsmUpdateObjEx" auf Seite 156	Aktualisiert die objInfo-Informationen, die einem bestimmten Archivierungsobjekt zugeordnet sind, selbst dann, wenn mehrere Objekte mit demselben Namen vorhanden sind, oder aktualisiert aktive Sicherungsobjekte.

Zugehörige Verweise:

API-Rückkehrcodes

dsmBeginGetData

Mit dem Funktionsaufruf **dsmBeginGetData** wird eine Zurückschreibungs- oder Abrufoperationen für eine Liste mit Objekten im Speicher gestartet. Diese Liste mit Objekten ist in der Struktur **dsmGetList** enthalten. Die Anwendung erstellt diese Liste mit Werten aus der Abfrage, die einem Aufruf **dsmBeginGetData** voranging.

Der Aufrufende muss die Felder für die Zurückschreibungsreihenfolge aus der Objektabfrage verwenden, um die Liste, die in diesem Aufruf enthalten ist, zu sortieren. Damit wird sichergestellt, dass die Objekte so effizient wie möglich aus dem Speicher zurückgeschrieben werden, ohne dass Datenbänder zurückgespult oder erneut geladen werden müssen.

Beim Abrufen vollständiger Objekte ist das Maximum für *dsmGetList.numObjID* DSM_MAX_GET_OBJ. Beim Abrufen von Teilobjekten ist das Maximum DSM-_MAX_PARTIAL_GET_OBJ.

Geben Sie nach dem Aufruf **dsmBeginGetData** einen oder mehrere Aufrufe **dsmGetObj** an, um jedes Objekt in der Liste abzurufen. Nachdem das jeweilige Objekt abgerufen wurde und keine weiteren Daten für das Objekt erforderlich sind, wird der Aufruf **dsmEndGetObj** gesendet.

Nachdem alle Objekte abgerufen wurden oder wenn der Aufruf **dsmEndGetObj** abgebrochen wird, wird der Aufruf **dsmEndGetData** gesendet. Sie können dann den Zyklus erneut starten.

Syntax

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

Parameter

dsUint32_t dsmHandle (I)
Die Kennung, die diesen Aufruf einem vorherigen Aufruf **dsmInitEx** zuordnet.

dsBool_t mountWait (I)
Ein boolescher Wert von 'true' oder 'false' gibt an, ob der Anwendungsclient auf das Laden von Offlinedatenträgern wartet, wenn die erforderlichen Daten momentan offline sind. Hat 'mountWait' den Wert 'true', wartet die Anwendung auf das Laden der erforderlichen Datenträger durch den Server. Die Anwendung wartet, bis die Datenträger geladen sind oder die Anforderung abgebrochen wird.

dsmGetType getType (I)
Ein Aufzählungstyp, der aus **gtBackup** und **gtArchive** besteht und angibt, welcher Typ von Objekt abgerufen werden soll.

dsmGetList *dsmGetObjListP (I)
Die Struktur, die Informationen zu den Objekten oder Teilobjekten enthält, die zurückgeschrieben oder abgerufen werden sollen. Die Struktur verweist auf eine Liste mit Objekt-IDs und - im Fall einer Zurückschreibung oder eines Abrufs von Teilobjekten - auf eine Liste zugeordneter Offsets und Längen. Wenn Ihre Anwendung die Funktion für die Zurückschreibung oder den Abruf von Teilobjekten verwendet, setzen Sie das Feld **dsmGetList.stVersion** auf **dsmGetListPORVersion**. Bei einer Zurückschreibung oder einem Abruf von Teilobjek-

ten können Sie Daten während des Sendevorgangs nicht komprimieren. Um dies durchzusetzen, setzen Sie **ObjAttr.objCompressed** auf *bTrue*.

Abb. 19 auf Seite 77 und Anhang B, „Quellendateien für API-Typdefinitionen", auf Seite 171 enthalten weitere Informationen zu dieser Struktur.

„Zurückschreibung oder Abruf von Teilobjekten" auf Seite 70 enthält weitere Informationen zur Zurückschreibung oder zum Abruf von Teilobjekten.

Rückkehrcodes

Die Rückkehrcodenummern sind in runden Klammern ( ) angegeben.

Tabelle 20. Rückkehrcodes für dsmBeginGetData

Rückkehrcode	Erläuterung
DSM_RC_ABORT_INVALID_OFFSET (33)	Der Offset, der während des Abrufs eines Teilobjekts angegeben wurde, ist größer als die Länge des Objekts.
DSM_RC_ABORT_INVALID_LENGTH (34)	Die Länge, die während des Abrufs eines Teilobjekts angegeben wurde, ist größer als die Länge des Objekts oder der Offset addiert zur Länge liegt hinter dem Ende des Objekts.
DSM_RC_NO_MEMORY (102)	Es ist kein Arbeitsspeicher mehr verfügbar, um die Anforderung auszuführen.
DSM_RC_NUMOBJ_EXCEED (2029)	**dsmGetList.numObjId** ist größer als DSM_MAX_GET_OBJ.
DSM_RC_OBJID_NOTFOUND (2063)	Die Objekt-ID wurde nicht gefunden. Das Objekt wurde nicht zurückgeschrieben.
DSM_RC_WRONG_VERSION_PARM (2065)	Die API-Version des Anwendungsclients stimmt nicht mit der Version der IBM Spectrum Protect-Bibliothek überein.

dsmBeginQuery

Mit dem Funktionsaufruf **dsmBeginQuery** wird eine Anforderung zur Abfrage von Informationen zu Daten, Dateibereichen und Verwaltungsklassen an den Server gestartet.

Die Abfrage mit **dsmBeginQuery** betrifft speziell Folgendes:
- Archivierte Daten
- Gesicherte Daten
- Aktive gesicherte Daten
- Dateibereiche
- Verwaltungsklassen

Die Abfragedaten, die von dem Aufruf zurückgegeben werden, werden von einem oder mehreren Aufrufen **dsmGetNextQObj** abgerufen. Wenn die Abfrage beendet ist, wird der Aufruf **dsmEndQuery** gesendet.

Syntax

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

Parameter

dsUint32_t dsmHandle (I)

Die Kennung, die diesen Aufruf einem vorherigen Aufruf **dsmInitEx** zuordnet.

dsmQueryType queryType (I)

Gibt den Abfragetyp an, der ausgeführt werden soll. Weisen Sie eine der folgenden Optionen zu:

qtArchive

Es werden archivierte Objekte abgefragt.

qtBackup

Es werden gesicherte Objekte abgefragt.

qtBackupActive

Es werden nur aktive gesicherte Objekte für den vollständigen von Ihnen übergebenen Dateibereichsnamen abgefragt. Diese Abfrage wird als „Direktaufruf" bezeichnet und ist eine effiziente Möglichkeit, aktive Objekte im Speicher abzufragen.

Voraussetzungen: Sie müssen als Rootbenutzer auf einem Betriebssystem UNIX oder Linux angemeldet sein.

qtFilespace

Es werden registrierte Dateibereiche abgefragt.

qtMC

Es werden definierte Verwaltungsklassen abgefragt.

qtBackupGroups

Es werden geschlossene Gruppen abgefragt.

qtOpenGroups

Es werden offene Gruppe abgefragt.

qtProxyNodeAuth

Es werden Knoten abgefragt, an die dieser Knoten Daten weiterleiten kann.

qtProxyNodePeer

Es werden Peerknoten mit demselben Ziel abgefragt.

dsmQueryBuff *queryBuffer (I)

Gibt einen Verweis auf einen Puffer an, der einer bestimmten Datenstruktur zugeordnet ist. Diese Struktur ist dem von Ihnen übergebenen Abfragetyp zugeordnet. Diese Strukturen enthalten die Auswahlkriterien für jeden Abfragetyp. Füllen Sie die Felder in jeder Struktur aus, um den Geltungsbereich der Abfrage anzugeben, die ausgeführt werden soll. In jeder Struktur enthält das Feld `stVersion` die Strukturversionsnummer.

Die Datenstrukturen und ihre zugehörigen Felder umfassen die folgenden Elemente:

qryArchiveData

objName

Der vollständige Objektname. Sie können ein Platzhalterzeichen, wie beispielsweise einen Stern (*) oder ein Fragezeichen (?), verwenden, was sowohl für den übergeordneten als auch den untergeordneten Teil des Namens gilt. Ein Stern entspricht

null oder mehr Zeichen; ein Fragezeichen entspricht einem Zeichen. Das Feld objType von objName kann einen der folgenden Werte haben:

- DSM_OBJ_FILE
- DSM_OBJ_DIRECTORY
- DSM_OBJ_ANY_TYPE

Weitere Informationen zu übergeordneten und untergeordneten Namen finden Sie hier: „Übergeordnete und untergeordnete Namen" auf Seite 23.

Eigner

Der Name des Eigners des Objekts.

insDateLowerBound

Die Untergrenze für das Einfügedatum, an dem das Objekt archiviert wurde. Um die standardmäßige Untergrenze abzurufen, setzen Sie die Komponente für das Jahr auf DATE_MINUS_INFINITE.

insDateUpperBound

Die Obergrenze für das Einfügedatum, an dem das Objekt archiviert wurde. Um die standardmäßige Obergrenze abzurufen, setzen Sie die Komponente für das Jahr auf DATE_PLUS_INFINITE.

expDateLowerBound

Die Untergrenze für das Verfallsdatum. Die Standardwerte für beide Felder für das Verfallsdatum stimmen mit denen für die Felder für das Einfügedatum überein.

expDateUpperBound

Die Obergrenze für das Verfallsdatum.

descr

Die Archivierungsbeschreibung. Geben Sie einen Stern (*) ein, wenn alle Beschreibungen berücksichtigt werden sollen.

qryBackupData

objName

Der vollständige Objektname. Sie können ein Platzhalterzeichen, wie beispielsweise einen Stern (*) oder ein Fragezeichen (?), verwenden, was sowohl für den übergeordneten als auch den untergeordneten Teil des Namens gilt. Ein Stern entspricht null oder mehr Zeichen; ein Fragezeichen entspricht einem Zeichen. Das Feld objType von objName kann einen der folgenden Werte haben:

- DSM_OBJ_FILE
- DSM_OBJ_DIRECTORY
- DSM_OBJ_ANY_TYPE

Weitere Informationen zu übergeordneten und untergeordneten Namen finden Sie hier: „Übergeordnete und untergeordnete Namen" auf Seite 23.

Eigner

Der Name des Eigners des Objekts.

objState

Sie können nach einem der folgenden Objektzustände abfragen:

- DSM_ACTIVE
- DSM_INACTIVE
- DSM_ANY_MATCH

pitDate

Der Zeitpunktwert. Eine Abfrage mit diesem Feld gibt das letzte Objekt zurück, das vor diesem Datum und dieser Uhrzeit gesichert wurde. objState kann 'aktiv' oder 'inaktiv' angeben. Objekte, die vor dem in pitDate angegebenen Zeitpunkt gelöscht wurden, werden nicht zurückgegeben. Beispiel:

```
Mo - ABC(1), DEF, GHI sichern
Di - ABC(2) sichern, DEF löschen
Do - ABC(3) sichern
```

Wenn Sie die Abfrage am Freitag mit einem Zeitpunktwert von Mittwoch 12:00:00 Uhr aufrufen, werden folgende Informationen zurückgegeben:

```
ABC(2) - eine inaktive Kopie
GHI    - eine aktive Kopie
```

Der Aufruf gibt DEF nicht zurück, weil dieses Objekt vor dem Zeitpunktwert gelöscht wurde.

qryABackupData

objName

Der vollständige Objektname. Sie können ein Platzhalterzeichen, wie beispielsweise einen Stern (*) oder ein Fragezeichen (?), verwenden, was sowohl für den übergeordneten als auch den untergeordneten Teil des Namens gilt. Ein Stern entspricht null oder mehr Zeichen; ein Fragezeichen entspricht einem Zeichen. Das Feld objType von objName kann einen der folgenden Werte haben:

- DSM_OBJ_FILE
- DSM_OBJ_DIRECTORY
- DSM_OBJ_ANY_TYPE

Weitere Informationen zu übergeordneten und untergeordneten Namen finden Sie hier: „Übergeordnete und untergeordnete Namen" auf Seite 23.

qryFSData

fsName

Geben Sie in diesem Feld den Namen eines bestimmten Dateibereichs ein oder geben Sie einen Stern (*) ein, um Informationen zu allen registrierten Dateibereichen abzurufen.

qryMCData

mcName

Geben Sie den Namen einer bestimmten Verwaltungsklasse ein oder geben Sie eine leere Zeichenfolge („ ”) ein, um Informationen zu allen Verwaltungsklassen abzurufen.

Anmerkung: Sie können keinen Stern (*) verwenden.

mcDetail
Gibt an, ob Informationen zu den Sicherungs- und Archivierungskopiengruppen der Verwaltungsklasse zurückgegeben werden. Die folgenden Werte sind gültig:

- `bTrue`
- `bFalse`

qryBackupGroup:

groupType
Der Gruppentyp ist `DSM_GROUPTYPE_PEER`.

fsName
Der Dateibereichsname.

Eigner
Die Eigner-ID.

groupLeaderObjId
Die Objekt-ID des Gruppenleiters.

objType
Der Objekttyp.

qryProxyNodeAuth:

targetNodeName
Der Zielknotenname.

peerNodeName
Der Peerknotenname.

hlAddress
Die Peeradresse des übergeordneten Namens.

llAddress
Die Peeradresse des untergeordneten Namens.

qryProxyNodePeer:

targetNodeName
Der Zielknotenname.

peerNodeName
Der Peerknotenname.

hlAddress
Die Peeradresse des übergeordneten Namens.

llAddress
Die Peeradresse des untergeordneten Namens.

Rückkehrcodes

In der folgenden Tabelle werden die Rückkehrcodes für den Funktionsaufruf **dsmBeginQuery** beschrieben.

Tabelle 21. Rückkehrcodes für dsmBeginQuery

Rückkehrcode	Rückkehrcodenummer	Erläuterung
DSM_RC_NO_MEMORY	102	Es ist nicht genug Speicher verfügbar, um die Anforderung auszuführen.
DSM_RC_FILE_SPACE_NOT_FOUND	124	Der angegebene Dateibereich wurde nicht gefunden.
DSM_RC_NO_POLICY_BLK	2007	Die Servermaßnahmeninformationen waren nicht verfügbar.
DSM_RC_INVALID_OBJTYPE	2010	Ungültiger Objekttyp.
DSM_RC_INVALID_OBJOWNER	2019	Ungültiger Objekteignername.
DSM_RC_INVALID_OBJSTATE	2024	Ungültige Objektbedingung.
DSM_RC_WRONG_VERSION_PARM	2065	Die API-Version des Anwendungsclients stimmt nicht mit der Version der IBM Spectrum Protect-Bibliothek überein.

dsmBeginTxn

Mit dem Funktionsaufruf **dsmBeginTxn** werden eine oder mehrere IBM Spectrum Protect-Transaktionen gestartet, die eine vollständige Aktion starten; entweder werden alle Aktionen erfolgreich ausgeführt oder es wird keine Aktion erfolgreich ausgeführt. Eine Aktion kann entweder ein einzelner Aufruf oder eine Serie von Aufrufen sein. Beispielsweise kann ein Aufruf **dsmSendObj**, auf den eine Reihe von Aufrufen **dsmSendData** folgt, als eine einzige Aktion betrachtet werden. In ähnlicher Weise wird ein Aufruf **dsmSendObj** mit **dataBlkPtr**, der einen Datenbereich angibt, der das zu sichernde Objekt enthält, ebenso als eine einzige Aktion betrachtet.

Versuchen Sie, mehrere Objekte für Datenübertragungsoperationen in einer einzigen Transaktion zusammenzufassen. Das Zusammenfassen von Objekten hat deutliche Leistungsverbesserungen im IBM Spectrum Protect-System zur Folge. Sowohl aus der Perspektive des Clients als auch aus der Perspektive des Servers bedeutet das Starten und Beenden jeder Transaktion einen gewissen Systemaufwand.

Die Ausführung einer einzigen Transaktion unterliegt bestimmten Einschränkungen. Diese Einschränkungen umfassen:

- Eine maximale Anzahl Objekte, die in einer einzigen Transaktion gesendet oder gelöscht werden können. Dieser Grenzwert ist in den Daten angegeben, die **dsmQuerySessInfo** im Feld *ApiSessInfo.maxObjPerTxn* zurückgibt. Dies entspricht der Serveroption *TxnGroupMax*.
- Alle Objekte, die in einer einzigen Transaktion an den Server gesendet werden (Sicherung oder Archivierung), müssen dasselbe Kopienziel haben, das in der Verwaltungsklassenbindung für das Objekt definiert ist. Dieser Wert ist in den Daten angegeben, die **dsmBindMC** im Feld **mcBindKey.backup_copy_dest** oder **mcBindKey.archive_copy_dest** zurückgibt.

Bei der API kann entweder der Anwendungsclient diese Einschränkungen überwachen und steuern oder die API kann diese Einschränkungen überwachen. Wenn

die API Einschränkungen überwacht, wird der Anwendungsclient über entsprechende Rückkehrcodes von den API-Aufrufen benachrichtigt, wenn eine oder mehrere Einschränkungen erfüllt sind.

Geben Sie für einen Aufruf **dsmBeginTxn** auch immer einen Aufruf **dsmEndTxn** an, um die Gruppe von Aktionen in einem Paar aus Aufrufen **dsmBeginTxn** und **dsmEndTxn** zu optimieren.

Syntax

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

Parameter

dsUint32_t dsmHandle (I)
Die Kennung, die diesen Aufruf einem vorherigen Aufruf **dsmInitEx** zuordnet.

Rückkehrcodes

Die Rückkehrcodenummern sind in runden Klammern ( ) angegeben.

Tabelle 22. Rückkehrcodes für dsmBeginTxn

Rückkehrcode	Erläuterung
DSM_RC_ABORT_NODE_NOT_AUTHORIZED (36)	FROMNODE oder FROMOWNER ist für TXN-Operationen nicht zulässig.

dsmBindMC

Mit den Funktionsaufruf **dsmBindMC** wird eine Verwaltungsklasse dem übergebenen Objekt zugeordnet bzw. an es gebunden. Das Objekt wird über die Einschluss-/Ausschlussliste übergeben, auf die in der Optionsdatei verwiesen wird. Wird in der Einschlussliste keine Übereinstimmung für eine bestimmte Verwaltungsklasse gefunden, wird die Standardverwaltungsklasse zugeordnet. Über die Ausschlussliste können Objekte von einer Sicherung ausgeschlossen werden, aber nicht von einer Archivierung.

Der Anwendungsclient kann mithilfe von Parametern, die in der Struktur `mcBindKey` zurückgegeben werden, bestimmen, ob dieses Objekt gesichert oder archiviert werden soll oder ob wegen unterschiedlicher Kopienziele eine neue Transaktion gestartet werden muss. Weitere Informationen finden Sie unter **dsmBeginTxn**.

Sie müssen **dsmBindMC** aufrufen, bevor Sie **dsmSendObj** aufrufen, da jedem Objekt eine Verwaltungsklasse zugeordnet werden muss. Dieser Aufruf kann innerhalb oder außerhalb einer Transaktion ausgeführt werden. Gibt beispielsweise in einer Transaktion mit mehreren Objekten **dsmBindMC** an, dass das Objekt ein anderes Kopienziel wie das vorherige Objekt hat, muss die Transaktion beendet und eine neue Transaktion gestartet werden. In diesem Fall ist kein weiterer Aufruf **dsmBindMC** erforderlich, da bereits ein derartiger Aufruf für dieses Objekt ausgeführt wurde.

Syntax

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

Parameter

dsUint32_t dsmHandle (I)
Die Kennung, die diesen Aufruf einem vorherigen Aufruf **dsmInitEx** zuordnet.

dsmObjName *objNameP (I)
Ein Verweis auf die Struktur, die den Dateibereichsnamen, den übergeordneten Objektnamen, den untergeordneten Objektnamen und den Objekttyp enthält.

dsmSendType sendType (I)
Gibt an, ob die Bindung dieser Verwaltungsklasse für Archivierungs- oder Sicherungssendeoperationen gilt. Gültige Werte für diesen Aufruf umfassen:

Name	Beschreibung
stBackup	Ein Sicherungsobjekt
stArchive	Ein Archivierungsobjekt
stBackupMountWait	Ein Sicherungsobjekt
stArchiveMountWait	Ein Archivierungsobjekt

Für den Aufruf **dsmBindMC** sind `stBackup` und `stBackupMountWait` sowie `stArchive` und `stArchiveMountWait` funktional entsprechend.

mcBindKey *mcBindKeyP (O)
Dies ist die Adresse einer Struktur `mcBindKey`, in der die Verwaltungsklasseninformationen zurückgegeben werden. Der Anwendungsclient kann mithilfe der zurückgegebenen Informationen bestimmen, ob dieses Objekt für eine Transaktion mit mehreren Objekten geeignet ist oder ob eine Verwaltungsklassenabfrage für die an das Objekt gebundene Verwaltungsklasse ausgeführt werden soll.

Rückkehrcodes

Die Rückkehrcodenummern sind in runden Klammern ( ) angegeben.

Tabelle 23. Rückkehrcodes für dsmBindMC

Rückkehrcode	Erläuterung
DSM_RC_NO_MEMORY (102)	Es ist kein Arbeitsspeicher mehr verfügbar, um die Anforderung auszuführen.
DSM_RC_INVALID_PARM (109)	Einer der übergebenen Parameter hat einen ungültigen Wert.
DSM_RC_TL_EXCLUDED (185)	Das Sicherungsobjekt wurde ausgeschlossen und kann nicht gesendet werden.
DSM_RC_INVALID_OBJTYPE (2010)	Ungültiger Objekttyp.
DSM_RC_INVALID_SENDTYPE (2022)	Ungültiger Sendetyp.
DSM_RC_WRONG_VERSION_PARM (2065)	Die API-Version des Anwendungsclients stimmt nicht mit der Version der IBM Spectrum Protect-Bibliothek überein.

dsmChangePW

Mit dem Funktionsaufruf **dsmChangePW** wird ein IBM Spectrum Protect-Kennwort geändert. Bei einem Mehrbenutzerbetriebssystem wie UNIX oder Linux kann dieser Aufruf nur vom Rootbenutzer oder vom berechtigten Benutzer verwendet werden.

Bei den Windows-Betriebssystemen können Sie das Kennwort in der Datei dsm.opt angeben. In diesem Fall wird die Datei dsm.opt nicht durch **dsmChangePW** aktualisiert. Nachdem der Aufruf **dsmChangePW** erfolgt ist, müssen Sie die Datei dsm.opt separat aktualisieren.

Dieser Aufruf muss erfolgreich verarbeitet werden, wenn **dsmInitEx** DSM_RC_VERIFIER_EXPIRED zurückgibt. Die Sitzung wird beendet, wenn der Aufruf **dsmChangePW** in dieser Situation fehlschlägt.

Wird **dsmChangePW** aus einem anderen Grund aufgerufen, bleibt die Sitzung unabhängig vom Rückkehrcode geöffnet.

Syntax

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

Parameter

dsUint32_t dsmHandle (I)
Die Kennung, die diesen Aufruf einem vorherigen Aufruf **dsmInitEx** zuordnet.

char *oldPW (I)
Das alte Kennwort des Aufrufenden. Die maximale Länge ist DSM_MAX_VERIFIER_LENGTH.

char *newPW (I)
Das neue Kennwort des Aufrufenden. Die maximale Länge ist DSM_MAX_VERIFIER_LENGTH.

Rückkehrcodes

Die Rückkehrcodenummern sind in runden Klammern ( ) angegeben.

Tabelle 24. Rückkehrcodes für dsmChangePW

Rückkehrcode	Erläuterung
DSM_RC_ABORT_BAD_VERIFIER (6)	Es wurde ein falsches Kennwort eingegeben.
DSM_RC_AUTH_FAILURE (137)	Authentifizierungsfehler. Das alte Kennwort ist falsch.
DSM_RC_NEWPW_REQD (2030)	Für das neue Kennwort muss ein Wert eingegeben werden.
DSM_RC_OLDPW_REQD (2031)	Für das alte Kennwort muss ein Wert eingegeben werden.
DSM_RC_PASSWD_TOOLONG (2103)	Das angegebene Kennwort ist zu lang.
DSM_RC_NEED_ROOT (2300)	Der Aufrufende der API muss ein Rootbenutzer oder ein berechtigter Benutzer sein.

dsmCleanUp

Der Funktionsaufruf **dsmCleanUp** wird verwendet, wenn **dsmSetUp** aufgerufen wurde. Der Funktionsaufruf **dsmCleanUp** sollte aufgerufen werden, nachdem **dsmTerminate** aufgerufen wurde. Nachdem **dsmCleanUp** aufgerufen wurde, können keine anderen Aufrufe erfolgen.

Es gibt keine spezifischen Rückkehrcodes für diesen Aufruf.

Syntax

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

Parameter

dsBool_t mtFlag (I)
Dieser Parameter gibt an, dass die API in einem Einzelthread- oder in einem Multithread-Modus verwendet wurde. Gültige Werte umfassen:

- DSM_SINGLETHREAD
- DSM_MULTITHREAD

dsmDeleteAccess

Mit dem Funktionsaufruf **dsmDeleteAccess** werden aktuelle Berechtigungsregeln für Sicherungsversionen oder archivierte Kopien Ihrer Objekte gelöscht. Wenn Sie eine Berechtigungsregel löschen, entziehen Sie den Zugriff, den ein Benutzer auf alle in der Regel angegebenen Dateien hat.

Bei Verwendung von **dsmDeleteAccess** können Sie jeweils nur eine einzige Regel löschen. Rufen Sie die Regel-ID über den Befehl **dsmQueryAccess** ab.

Es gibt keine spezifischen Rückkehrcodes für diesen Aufruf.

Syntax

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

Parameter

dsUint32_t dsmHandle (I)
Die Kennung, die diesen Aufruf einem vorherigen Aufruf **dsmInitEx** zuordnet.

dsUint32_t ruleNum (I)
Die Regel-ID für die Zugriffsregel, die gelöscht wird. Dieser Wert wird mit einem Funktionsaufruf **dsmQueryAccess** abgerufen.

dsmDeleteFS

Mit dem Funktionsaufruf **dsmDeleteFS** wird ein Dateibereich aus dem Speicher gelöscht. Um einen Dateibereich löschen zu können, muss Ihnen der IBM Spectrum Protect-Administrator die korrekten Berechtigungen erteilen. Rufen Sie **dsmQuerySessInfo** auf, um zu bestimmen, ob Sie über die erforderlichen Berechtigungen verfügen. Dieser Funktionsaufruf gibt eine Datenstruktur des Typs *ApiSessInfo* zurück, die zwei Felder, *archDel* und *backDel*, enthält.

Anmerkung:

- Auf einem Betriebssystem UNIX oder Linux kann nur ein Rootbenutzer oder ein berechtigter Benutzer einen Dateibereich löschen.
- Wenn der Dateibereich, der gelöscht werden muss, Sicherungsversionen enthält, müssen Sie die Berechtigung zum Löschen von Sicherungen (***backDel*** = BACKDEL_YES) haben. Wenn der Dateibereich Archivierungskopien enthält, müssen Sie die Berechtigung zum Löschen von Archivierungen (*archDel* = ARCHDEL_YES) haben. Wenn der Dateibereich sowohl Sicherungsversionen als auch Archivierungskopien enthält, müssen Sie beide Typen von Löschberechtigung haben.
- Bei der Verwendung eines Archivierungsmanagerservers kann ein Dateibereich nicht wirklich entfernt werden. Dieser Funktionsaufruf gibt selbst dann *rc=0* zurück, wenn der Dateibereich nicht wirklich gelöscht wurde. Die einzige Möglichkeit zur Überprüfung, ob der Dateibereich gelöscht wurde, ist die Ausgabe einer Dateibereichsabfrage an den Server.
- Die IBM Spectrum Protect-Serverfunktion zum Löschen von Dateibereichen ist ein Hintergrundprozess. Wenn außer den Fehlern, die vor der Übergabe eines Rückkehrcodes erkannt wurden, noch weitere Fehler auftreten, werden diese im Protokoll des IBM Spectrum Protect-Servers aufgezeichnet.

Syntax

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

Parameter

dsUint32_t dsmHandle (I)
Die Kennung, die diesen Aufruf einem vorherigen Aufruf **dsmInitEx** zuordnet.

char *fsName (I)
Ein Verweis auf den Namen des Dateibereichs, der gelöscht werden soll. Das Platzhalterzeichen ist nicht zulässig.

unsigned char repository (I)
Gibt an, ob der zu löschende Dateibereich ein Sicherungsrepository und/oder ein Archivrepository ist. Gültige Werte für dieses Feld umfassen:

```
DSM_ARCHIVE_REP     /* Archivrepository     */
DSM_BACKUP_REP      /* Sicherungsrepository */
DSM_REPOS_ALL       /* alle Repository-Typen*/
```

Rückkehrcodes

Die Rückkehrcodenummern sind in runden Klammern ( ) angegeben.

Tabelle 25. Rückkehrcodes für dsmDeleteFS

Rückkehrcode	Erläuterung
DSM_RC_ABORT_NOT_AUTHORIZED (27)	Sie haben nicht die erforderliche Berechtigung zum Löschen des Dateibereichs.
DSM_RC_INVALID_REPOS (2015)	Ungültiger Wert für Repository.
DSM_RC_FSNAME_NOTFOUND (2060)	Dateibereichsname nicht gefunden.
DSM_RC_NEED_ROOT (2300)	Der Aufrufende der API muss ein Rootbenutzer sein.

dsmDeleteObj

Mit dem Funktionsaufruf **dsmDeleteObj** werden Sicherungsobjekte inaktiviert oder gelöscht oder Archivierungsobjekte im Speicher gelöscht. Beim Typ **dtBackup** wird nur die momentan aktive Sicherungskopie inaktiviert. Beim Typ **dtBackupID** wird das Objekt mit der jeweils angegebenen Objekt-ID vom Server gelöscht. Rufen Sie diese Funktion innerhalb einer Transaktion auf.

Weitere Informationen finden Sie unter **dsmBeginTxn**.

Bevor Sie den Aufruf **dsmDeleteObj** senden, senden Sie die in „IBM Spectrum Protect-System abfragen" auf Seite 33 beschriebene Abfragefolge, um die Informationen für **delInfo** abzurufen. Der Aufruf **dsmGetNextQObj** gibt für Sicherungsabfragen eine Datenstruktur mit dem Namen **qryRespBackupData** und für Archivabfragen eine Datenstruktur mit dem Namen **qryRespArchiveData** zurück. Diese Datenstrukturen enthalten Informationen, die Sie für **delInfo** benötigen.

Der Wert von **maxObjPerTxn** gibt die maximale Anzahl Objekte an, die in einer einzigen Transaktion gelöscht werden können. Rufen Sie **dsmQuerySessInfo** auf, um diesen Wert abzurufen.

Tipp: Ihr Knoten muss die entsprechende, von Ihrem Administrator festgelegte Berechtigung haben. Um Archivierungsobjekte löschen zu können, müssen Sie die Berechtigung zum Löschen von Archivierungen haben. Um ein Sicherungsobjekt inaktivieren zu können, benötigen Sie keine Berechtigung zum Löschen von Sicherungen.

Syntax

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

Parameter

dsUint32_t dsmHandle (I)
Die Kennung, die diesen Aufruf einem vorherigen Aufruf **dsmInitEx** zuordnet.

dsmDelType delType (I)
Gibt an, welcher Typ von Objekt (Sicherungs- oder Archivobjekt) gelöscht werden soll. Gültige Werte umfassen:

Name	Beschreibung
dtArchive	Das zu löschende Objekt wurde zuvor archiviert.

Name	Beschreibung
dtBackup	Das zu inaktivierende Objekt wurde zuvor gesichert.
dtBackupID	Das zu löschende Objekt wurde zuvor gesichert. **Einschränkung:** Wird dieser Löschtyp (**delType**) zusammen mit *objID* verwendet, wird das Sicherungsobjekt vom Server gelöscht. Ein Objekt kann nur von seinem Eigner gelöscht werden. Jede Version (aktiv oder inaktiv) eines Objekts kann gelöscht werden. Der Server gleicht die Versionen ab. Wird eine aktive Version eines Objekts gelöscht, wird die erste inaktive Version aktiv. Wird eine inaktive Version eines Objekts gelöscht, werden alle älteren Versionen vorverlegt. Der Knoten muss mit der Berechtigung **backDel** registriert sein.

dsmDelInfo delInfo (I)
Eine Struktur, deren Felder das Objekt identifizieren. Abhängig davon, ob das Objekt ein Sicherungsobjekt oder ein Archivierungsobjekt ist, sind die Felder unterschiedlich. Die Struktur zum Inaktivieren eines Sicherungsobjekts, `delBack`, enthält den Objektnamen und die Objektkopiengruppe. Die Struktur für ein Archivierungsobjekt, `delArch`, enthält die Objekt-ID.

Die Struktur zum Entfernen eines Sicherungsobjekts, `delBackID`, enthält die Objekt-ID.

Rückkehrcodes

Die Rückkehrcodenummern sind in runden Klammern ( ) angegeben.

Tabelle 26. Rückkehrcodes für dsmDeleteObj

Rückkehrcode	Erläuterung
DSM_RC_FS_NOT_REGISTERED (2061)	Dateibereichsname ist nicht registriert.
DSM_RC_WRONG_VERSION_PARM (2065)	Die API-Version des Anwendungsclients stimmt nicht mit der Version der IBM Spectrum Protect-Bibliothek überein.

dsmEndGetData

Mit dem Funktionsaufruf **dsmEndGetData** wird eine **dsmBeginGetData**-Sitzung, mit der Objekte aus Speicher abgerufen werden, beendet.

Der Funktionsaufruf **dsmEndGetData** wird gestartet, nachdem alle Objekte, die zurückgeschrieben werden sollen, verarbeitet wurden, oder er beendet den Abrufprozess vorzeitig. Rufen Sie **dsmEndGetData** auf, um eine **dsmBeginGetData**-Sitzung zu beenden, bevor Sie die weitere Verarbeitung fortsetzen können.

Abhängig davon, wann **dsmEndGetData** aufgerufen wird, muss die API unter Umständen die Verarbeitung eines Teildatenstroms beenden, bevor der Prozess gestoppt werden kann. Der Aufrufende sollte daher keine sofortige Rückkehr von diesem Aufruf erwarten. Verwenden Sie **dsmTerminate**, wenn die Anwendung sofort die Sitzung schließen und die Zurückschreibung beenden muss.

Es gibt keine spezifischen Rückkehrcodes für diesen Aufruf.

Syntax

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

Parameter

dsUint32_t dsmHandle (I)
Die Kennung, die diesen Aufruf einem vorherigen Aufruf **dsmInitEx** zuordnet.

dsmEndGetDataEx

Mit dem Funktionsaufruf **dsmEndGetDataEx** wird die Gesamtanzahl gesendeter LAN-unabhängiger Byte angegeben. Er ist eine Erweiterung des Funktionsaufrufs **dsmEndGetData**.

Syntax

Es gibt keine spezifischen Rückkehrcodes für diesen Aufruf.

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

Parameter

dsmEndGetDataExIn_t *dsmEndGetDataExInP (I)
Übergibt den dsmHandle für "'Objekt abrufen' beenden", der die Sitzung identifiziert und nachfolgenden Aufrufen zuordnet.

dsmEndGetDataExOut_t *dsmEndGetDataExOutP (O)
Diese Struktur enthält diesen Eingabeparameter:

totalLFBytesRecv
Die Gesamtanzahl empfangener LAN-unabhängiger Byte.

dsmEndGetObj

Mit dem Funktionsaufruf **dsmEndGetObj** wird eine **dsmGetObj**-Sitzung, mit der Daten für ein bestimmtes Objekt abgerufen werden, beendet.

Starten Sie den Aufruf **dsmEndGetObj**, nachdem ein Datenende für das Objekt empfangen wurde. Dies gibt an, dass alle Daten empfangen wurden oder dass keine weiteren Daten für dieses Objekt empfangen werden. Bevor Sie einen weiteren Aufruf **dsmGetObj** starten können, müssen Sie **dsmEndGetObj** aufrufen.

Abhängig davon, wann **dsmEndGetObj** aufgerufen wird, muss die API unter Umständen die Verarbeitung eines Teildatenstroms beenden, bevor der Prozess gestoppt werden kann. Erwarten Sie keine sofortige Rückkehr von diesem Aufruf.

Syntax

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

Parameter

dsUint32_t dsmHandle (I)
Die Kennung, die diesen Aufruf einem vorherigen Aufruf **dsmInitEx** zuordnet.

Rückkehrcodes

Die Rückkehrcodenummern sind in runden Klammern ( ) angegeben.

Tabelle 27. Rückkehrcodes für dsmEndGetObj

Rückkehrcode	Erläuterung
DSM_RC_NO_MEMORY (102)	Es ist kein Arbeitsspeicher mehr verfügbar, um die Anforderung auszuführen.

dsmEndQuery

Mit dem Funktionsaufruf **dsmEndQuery** wird das Ende einer Aktion **dsmBeginQuery** angegeben. Der Anwendungsclient sendet einen Aufruf **dsmEndQuery**, um eine Abfrage abzuschließen. Dieser Aufruf wird entweder gesendet, nachdem alle Abfrageantworten über **dsmGetNextQObj** abgerufen wurden, oder er wird gesendet, um eine Abfrage zu beenden, bevor alle Daten zurückgegeben werden.

Tipp: IBM Spectrum Protect sendet in diesem Fall weiterhin Abfragedaten vom Server an den Client, die API löscht jedoch alle verbleibenden Daten.

Nachdem ein Aufruf **dsmBeginQuery** gesendet wurde, muss ein Aufruf **dsmEndQuery** gesendet werden, bevor irgendeine andere Aktivität gestartet werden kann.

Es gibt keine spezifischen Rückkehrcodes für diesen Aufruf.

Syntax

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

Parameter

dsUint32_t dsmHandle (I)
Die Kennung, die diesen Aufruf einem vorherigen Aufruf **dsmInitEx** zuordnet.

dsmEndSendObj

Mit dem Funktionsaufruf **dsmEndSendObj** wird das Ende der Daten angegeben, die in den Speicher gesendet werden.

Geben Sie den Funktionsaufruf **dsmEndSendObj** ein, um das Ende der Daten für die Aufrufe **dsmSendObj** und **dsmSendData** anzugeben. Erfolgt dies nicht, ist ein fehlerhaftes Protokoll die Folge. Eine Ausnahme von dieser Regel ist das Aufrufen von **dsmEndTxn** zum Beenden der Transaktion. Damit werden alle Daten, die für die Transaktion gesendet wurden, gelöscht.

Syntax

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

Parameter

dsUint32_t dsmHandle (I)
Die Kennung, die diesen Aufruf einem vorherigen Aufruf **dsmInitEx** zuordnet.

Rückkehrcodes

Die Rückkehrcodenummern sind in runden Klammern ( ) angegeben.

Tabelle 28. Rückkehrcodes für dsmEndSendObj

Rückkehrcode	Erläuterung
DSM_RC_NO_MEMORY (102)	Es ist kein Arbeitsspeicher mehr verfügbar, um diese Anforderung auszuführen.

dsmEndSendObjEx

Mit dem Funktionsaufruf **dsmEndSendObjEx** werden zusätzliche Informationen in Bezug auf die Anzahl verarbeiteter Byte bereitgestellt. Die Informationen umfassen: Gesamtanzahl gesendeter Byte, Komprimierungsangaben, LAN-unabhängige Byte und Angaben zur Deduplizierung.

Der Funktionsaufruf **dsmEndSendObjEx** ist eine Erweiterung des Funktionsaufrufs **dsmEndSendObj**.

Syntax

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

Parameter

dsmEndSendObjExIn_t *dsmEndSendObjExInP (I)
Dieser Parameter übergibt den dsmHandle für "'Objekt senden' beenden", der die Sitzung identifiziert und nachfolgenden Aufrufen zuordnet.

dsmEndSendObjExOut_t *dsmEndSendObjExOutP (O)
Dieser Parameter übergibt die Informationen für "'Objekt senden' beenden":

Name	Beschreibung
totalBytesSent	Die Gesamtanzahl Byte, die von der Anwendung gelesen wird.
objCompressed	Ein Flag, das anzeigt, ob das Objekt komprimiert wurde.
totalCompressedSize	Die Gesamtgröße in Byte nach der Komprimierung.
totalLFBytesSent	Die Gesamtanzahl gesendeter LAN-unabhängiger Byte.
objDeduplicated	Ein Flag, das anzeigt, ob das Objekt von der API dedupliziert wurde.
totalDedupSize	Gesamtanzahl gesendeter Byte nach der Deduplizierung.

Rückkehrcodes

Die Rückkehrcodenummern sind in runden Klammern ( ) angegeben.

Tabelle 29. Rückkehrcodes für dsmEndSendObjEx

Rückkehrcode	Erläuterung
DSM_RC_NO_MEMORY (102)	Es ist kein Arbeitsspeicher mehr verfügbar, um diese Anforderung auszuführen.

dsmEndTxn

Mit dem Funktionsaufruf **dsmEndTxn** wird eine IBM Spectrum Protect-Transaktion beendet. Geben Sie den Funktionsaufruf **dsmEndTxn** zusammen mit **dsmBeginTxn** an, um den Aufruf oder die Gruppe von Aufrufen, die als eine Transaktion betrachtet werden, zu identifizieren. Der Anwendungsclient kann im Aufruf **dsmEndTxn** angeben, ob die Transaktion festgeschrieben oder beendet werden muss.

Führen Sie alle folgenden Aufrufe innerhalb der Grenzen einer Transaktion aus:

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

Syntax

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

Parameter

dsUint32_t dsmHandle (I)
Die Kennung, die diesen Aufruf einem vorherigen Aufruf **dsmInitEx** zuordnet.

dsUint8_t vote (I)
Gibt an, ob der Anwendungsclient alle Aktionen, die zwischen dem vorherigen Aufruf **dsmBeginTxn** und diesem Aufruf ausgeführt werden, festschreibt. Die folgenden Werte sind gültig:

```
DSM_VOTE_COMMIT     /* akt. Transaktion festschreiben*/
DSM_VOTE_ABORT      /* akt. Transakt. rückg. machen  */
```

Verwenden Sie DSM_VOTE_ABORT nur, wenn Ihre Anwendung einen Grund zum Stoppen der Transaktion findet.

dsUint16_t *reason (O)
Wenn der Aufruf **dsmEndTxn** mit einem Fehler beendet wird oder der Wert von vote nicht akzeptiert wird, gibt der Ursachencode dieses Parameters an, warum das Votum fehlgeschlagen ist. Der Rückkehrcode für den Aufruf kann null sein, während der Ursachencode ungleich null sein kann. Daher muss der Anwendungsclient immer sowohl den Rückkehrcode als auch den Ursachencode (if (rc || reason)) auf Fehler überprüfen, bevor die Ausführung als erfolgreich betrachtet werden kann.

Wenn die Anwendung als Votum DSM_VOTE_ABORT angibt, ist der Ursachencode DSM_RS_ABORT_BY_CLIENT (3). Eine Liste der möglichen Ursachencodes finden Sie in Anhang A, „Quellendatei mit API-Rückkehrcodes: dsmrc.h", auf Seite 159. Die Zahlen 1 bis 50 in der Liste der Rückkehrcodes sind für die Ursachencodes reserviert. Wenn der Server die Transaktion beendet, ist der Rückkehrcode DSM_RC_CHECK_REASON_CODE. In diesem Fall enthält der Wert des Ursachencodes weitere Informationen zu der Ursache des Abbruchs.

Rückkehrcodes

Die Rückkehrcodenummern sind in runden Klammern ( ) angegeben.

Tabelle 30. Rückkehrcodes für dsmEndTxn

Rückkehrcode	Erläuterung
DSM_RC_ABORT_CRC_FAILED (236)	Das Befehlserkennungszeichen (CRC), das vom Server empfangen wurde, entspricht nicht dem CRC, das vom Client errechnet wurde.
DSM_RC_INVALID_VOTE (2011)	Der für vote angegebene Wert ist ungültig.
DSM_RC_CHECK_REASON_CODE (2302)	Die Transaktion wurde abgebrochen. Überprüfen Sie das Feld für die Ursache.
DSM_RC_ABORT_STGPOOL_COPY_CONT_NO (241)	Der Schreibvorgang in einen der Kopienspeicherpools ist fehlgeschlagen und die Option COPYCONTINUE für den IBM Spectrum Protect-Speicherpool ist auf NO gesetzt. Die Transaktion wird beendet.
DSM_RC_ABORT_RETRY_SINGLE_TXN (242)	Dieser Abbruchcode gibt an, dass die aktuelle Transaktion aufgrund eines Fehlers während einer Speicheroperation abgebrochen wurde. Der Fehler kann behoben werden, indem jede Datei in einer eigenen Transaktion gesendet wird. Dieser Fehler ist typisch bei folgenden Bedingungen: • Der nächste Speicherpool hat eine andere Kopienspeicherpoolliste. • Die Operation wechselt mitten in einer Transaktion zu diesem Pool. .

dsmEndTxnEx

Mit dem Funktionsaufruf **dsmEndTxnEx** werden Informationen zur Objekt-ID des Gruppenleiters zur Verwendung im Funktionsaufruf **dsmGroupHandler** bereitgestellt. Er ist eine Erweiterung des Funktionsaufrufs **dsmEndTxn**.

Syntax

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

Parameter

dsmEndTxnExIn_t *dsmEndTxnExInP (I)
Diese Struktur enthält die folgenden Parameter:

dsmHandle
Die Kennung, die die Sitzung identifiziert und nachfolgenden IBM Spectrum Protect-Aufrufen zuordnet.

dsUint8_t vote (I)
Gibt an, ob der Anwendungsclient alle Aktionen, die zwischen dem vorherigen Aufruf **dsmBeginTxn** und diesem Aufruf ausgeführt werden, festschreibt. Gültige Werte sind:

```
DSM_VOTE_COMMIT     /* akt. Transaktion festschreiben*/
DSM_VOTE_ABORT      /* akt. Transakt. rückg. machen  */
```

Verwenden Sie DSM_VOTE_ABORT nur, wenn von Ihrer Anwendung ein Grund zum Stoppen der Transaktion gefunden wurde.

dsmEndTxnExOut_t *dsmEndTxnExOutP (0)
Diese Struktur enthält die folgenden Parameter:

dsUint16_t *reason (0)
Wenn der Aufruf **dsmEndTxnEx** mit einem Fehler beendet wird oder der Wert von *vote* nicht akzeptiert wird, gibt der Ursachencode dieses Parameters an, warum das Votum fehlgeschlagen ist.

Tipp: Der Rückkehrcode für den Aufruf kann null sein, während der Ursachencode ungleich null sein kann. Daher muss der Anwendungsclient immer sowohl den Rückkehrcode als auch den Ursachencode (`if (rc || reason)`) auf Fehler überprüfen, bevor die Ausführung als erfolgreich betrachtet werden kann.

Wenn die Anwendung als Votum DSM_VOTE_ABORT angibt, ist der Ursachencode DSM_RS_ABORT_BY_CLIENT (3). Eine Liste der möglichen Ursachencodes finden Sie in Anhang A, „Quellendatei mit API-Rückkehrcodes: `dsmrc.h`", auf Seite 159. Die Zahlen 1 bis 50 in der Liste der Rückkehrcodes sind für die Ursachencodes reserviert. Wenn der Server die Transaktion beendet, ist der Rückkehrcode DSM_RC_CHECK_REASON_CODE. In diesem Fall enthält der Wert des Ursachencodes weitere Informationen zu der Ursache des Abbruchs.

groupLeaderObjId
Die Objekt-ID des Gruppenleiters, die zurückgegeben wird, wenn das Flag DSM_ACTION_OPEN mit dem Aufruf **dsmGroupHandler** verwendet wird.

Rückkehrcodes

Die Rückkehrcodenummern sind in runden Klammern ( ) angegeben.

Tabelle 31. Rückkehrcodes für dsmEndTxnEx

Rückkehrcode	Erläuterung
DSM_RC_INVALID_VOTE (2011)	Der für 'vote' angegebene Wert ist ungültig.
DSM_RC_CHECK_REASON_CODE (2302)	Die Transaktion wurde abgebrochen. Überprüfen Sie das Feld für die Ursache.
DSM_RC_ABORT_STGPOOL_COPY_CONT_NO (241)	Der Schreibvorgang in einen der Kopienspeicherpools ist fehlgeschlagen und die Option COPYCONTINUE für den IBM Spectrum Protect-Speicherpool wurde auf NO gesetzt. Die Transaktion wird beendet.
DSM_RC_ABORT_RETRY_SINGLE_TXN (242)	Während einer Operation für gleichzeitiges Schreiben wird ein Objekt in der Transaktion an ein Ziel mit anderen Kopienspeicherpools gesendet. Beenden Sie die aktuelle Transaktion und senden Sie jedes Objekt erneut in einer eigenen Transaktion.

dsmGetData

Mit dem Funktionsaufruf **dsmGetData** wird ein Bytestrom mit Daten von IBM Spectrum Protect abgerufen und in den Puffer des Aufrufenden gestellt. Der Anwendungsclient ruft **dsmGetData** auf, wenn weitere Daten für den Empfang von einem vorherigen Aufruf **dsmGetObj** oder **dsmGetData** verfügbar sind.

Syntax

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

Parameter

dsUint32_t dsmHandle (I)
Die Kennung, die diesen Aufruf einem vorherigen Aufruf **dsmInitEx** zuordnet.

DataBlk *dataBlkPtr (I/O)
Verweist auf eine Struktur, die sowohl einen Zeiger auf den Puffer für die Daten enthält, die empfangen werden, als auch einen Zeiger auf die Größe des Puffers. Bei der Rückkehr enthält diese Struktur die Anzahl tatsächlich übertragener Byte. Die Typdefinition finden Sie in Anhang B, „Quellendateien für API-Typdefinitionen", auf Seite 171.

Rückkehrcodes

Die Rückkehrcodenummern sind in runden Klammern ( ) angegeben.

Tabelle 32. Rückkehrcodes für dsmGetData

Rückkehrcode	Erläuterung
DSM_RC_ABORT_INVALID_OFFSET (33)	Der Offset, der während des Abrufs eines Teilobjekts angegeben wurde, ist größer als die Länge des Objekts.
DSM_RC_ABORT_INVALID_LENGTH (34)	Die Länge, die während des Abrufs eines Teilobjekts angegeben wurde, ist größer als die Länge des Objekts oder der Offset addiert zur Länge liegt hinter dem Ende des Objekts.
DSM_RC_FINISHED (121)	Die Verarbeitung wurde beendet. Der letzte Puffer wurde empfangen. Überprüfen Sie `numBytes` auf das Datenvolumen und rufen Sie dann IBM Spectrum ProtectdsmEndGetObj auf.
DSM_RC_NULL_DATABLKPTR (2001)	Datenblockzeiger ist null.
DSM_RC_ZERO_BUFLEN (2008)	Puffergröße für Datenblockzeiger ist null.
DSM_RC_NULL_BUFPTR (2009)	Pufferzeiger für Datenblockzeiger ist null.
DSM_RC_WRONG_VERSION_PARM (2065)	Die API-Version des Anwendungsclients stimmt nicht mit der Version der IBM Spectrum Protect-Bibliothek überein.
DSM_RC_MORE_DATA (2200)	Es sind keine weiteren Daten zum Abrufen verfügbar.

dsmGetBufferData

Mit dem Funktionsaufruf **dsmGetBufferData** wird ein Bytestrom mit Daten über einen Puffer von IBM Spectrum Protect empfangen. Nach jedem Aufruf muss die Anwendung die Daten kopieren und den Puffer über einen Aufruf **dsmReleaseBuffer** freigeben. Wenn die Anzahl Puffer, die von der Anwendung angehalten wurden, dem im Aufruf **dsmInitEx** für 'numTsmBuffers' angegebenen Wert entspricht, blockt die Funktion **dsmGetBufferData** das Senden von Daten, bis ein Aufruf **dsmReleaseBuffer** gesendet wird.

Syntax

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

Parameter

getDataExIn_t * dsmGetBufferDataExInP (I)
Diese Struktur enthält den folgenden Eingabeparameter:

dsUint32_t dsmHandle
Die Kennung, die die Sitzung identifiziert und einem vorherigen Aufruf **dsmInitEx** zuordnet.

getDataExOut_t * dsmGetBufferDataExOutP (0)
Diese Struktur enthält die folgenden Ausgabeparameter:

dsUint8_t tsmBufferHandle(0)
Die Kennung, die den empfangenen Puffer identifiziert.

char *dataPtr(0)
Die Adresse, an die die Daten geschrieben wurden.

dsUint32_t numBytes(0)
Tatsächliche Anzahl Byte, die von IBM Spectrum Protect geschrieben wurde.

Rückkehrcodes

Die Rückkehrcodenummern sind in runden Klammern ( ) angegeben.

Tabelle 33. Rückkehrcodes für dsmGetBufferData

Rückkehrcode	Erläuterung
DSM_RC_BAD_CALL_SEQUENCE (2041)	Der Aufruf wurde nicht im korrekten Status ausgegeben.
DSM_RC_OBJ_ENCRYPTED (2049)	Diese Funktion kann nicht für verschlüsselte Objekte verwendet werden.
DSM_RC_OBJ_COMPRESSED (2048)	Diese Funktion kann nicht für komprimierte Objekte verwendet werden.
DSM_RC_BUFF_ARRAY_ERROR (2045)	Ein Pufferarrayfehler ist aufgetreten.

dsmGetNextQObj

Mit dem Funktionsaufruf **dsmGetNextQObj** wird die nächste Abfrageantwort von einem vorherigen Aufruf **dsmBeginQuery** abgerufen und in den Puffer des Aufrufenden gestellt.

Der Aufruf **dsmGetNextQObj** wird ein oder mehrmals gesendet. Bei jedem Aufruf der Funktion wird entweder ein einzelner Abfragesatz abgerufen oder ein Fehler oder der Ursachencode DSM_RC_FINISHED zurückgegeben. Wird DSM_RC_FINISHED zurückgegeben, sind keine weiteren Daten zu verarbeiten. Wenn alle Abfragedaten abgerufen wurden oder keine weiteren Abfragedaten benötigt werden, senden Sie den Aufruf **dsmEndQuery**, um den Abfrageprozess zu beenden.

Der Parameter **dataBlkPtr** muss auf einen Puffer verweisen, der mit dem Strukturtyp **qryResp*Data** definiert ist. Der Kontext, in dem **dsmGetNextQObj** aufgerufen wird, bestimmt den Strukturtyp für die Abfrageantwort.

Syntax

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

Parameter

dsUint32_t dsmHandle (I)
Die Kennung, die diesen Aufruf einem vorherigen Aufruf **dsmInitEx** zuordnet.

DataBlk *dataBlkPtr (I/O)
Verweist auf eine Struktur, die sowohl einen Zeiger auf den Puffer für die Daten enthält, die empfangen werden sollen, als auch einen Zeiger auf die Größe des Puffers. Dieser Puffer ist die Antwortstruktur des Typs **qryResp*Data**. Bei der Rückkehr enthält diese Struktur die Anzahl übertragener Byte. Die Struktur, die jedem Typ von Abfrage zugeordnet ist, wird in der folgenden Tabelle beschrieben. Weitere Informationen zur Typdefinition bei **DataBlk** finden Sie hier: Anhang B, „Quellendateien für API-Typdefinitionen", auf Seite 171.

Tabelle 34. DataBlk-Zeigerstruktur.

Abfrage	Antwortstruktur	Felder von besonderem Interesse
qtArchive	**qryRespArchiveData**	**sizeEstimate** Enthält den Wert, der in einem vorherigen Aufruf **dsmSendObj** übergeben wurde. **mediaClass** Kann den Wert MEDIA_FIXED haben, wenn sich das Objekt auf Platte befindet, oder den Wert MEDIA_LIBRARY, wenn sich das Objekt auf Band befindet. **clientDeduplicated** Gibt an, ob dieses Objekt vom Client dedupliziert wird.

Tabelle 34. DataBlk-Zeigerstruktur (Forts.).

Abfrage	Antwortstruktur	Felder von besonderem Interesse
qtBackup	**qryRespBackupData**	**restoreOrderExt** Hat den Typ dsUint16_t. Sortieren Sie anhand dieses Felds, wenn in einem Aufruf **dsmBeginGetData** mehrere Objekte zurückgeschrieben werden. Das API-Muster, dapiqry.c, enthält ein Beispiel für Sortiercode für diesen Aufruf. Ein Beispiel für das Sortieren finden Sie hier: Abb. 16 auf Seite 72. **sizeEstimate** Enthält den Wert, der in einem vorherigen Aufruf **dsmSendObj** übergeben wurde. **mediaClass** Kann den Wert MEDIA_FIXED haben, wenn sich das Objekt auf Platte befindet, oder den Wert MEDIA_LIBRARY, wenn sich das Objekt auf Band befindet. **clientDeduplicated** Gibt an, ob dieses Objekt vom Client dedupliziert wird.
qtBackupActive	**qryARespBackupData**	
qtBackupGroups	**qryRespBackupData**	**dsBool_t isGroupLeader** Gibt, wenn wahr, an, dass es sich bei diesem Objekt um einen Gruppenleiter handelt.
qtOpenGroups	**qryRespBackupData**	**dsBool_t isOpenGroup;** Gibt, wenn wahr, an, dass diese Gruppe offen und nicht vollständig ist.

Tabelle 34. DataBlk-Zeigerstruktur (Forts.).

Abfrage	Antwortstruktur	Felder von besonderem Interesse
qtFilespace	**qryRespFSData**	**backStartDate** Enthält die Zeitmarke des Servers, die angibt, wann der Dateibereich mit der Aktion **backStartDate** aktualisiert wird. **backCompleteDate** Enthält die Zeitmarke des Servers, die angibt, wann der Dateibereich mit der Aktion **backCompleteDate** aktualisiert wird. **lastReplStartDate** Enthält die Zeitmarke für den letzten Start der Replikation auf dem Server. **lastReplCmpltDate** Enthält die Zeitmarke für die letzte Beendigung der Replikation, auch wenn es einen Fehler gab. **lastBackOpDateFromServer** Enthält die letzte Speicherungszeitmarke, die auf dem Server gespeichert wurde. **lastBackOpDateFromLocal** Enthält die letzte Speicherungszeitmarke, die auf dem Client gespeichert wurde.
qtMC	**qryRespMCData** **qryRespMCDetailData**	
qtProxyNodeAuth	**qryRespProxyNodeData** **targetNodeName** **peerNodeName** **hlAddress** **llAddress**	
qtProxyNodePeer	**qryRespProaxyNodeData** **targetNodeName** **peerNodeName** **hlAddress** **llAddress**	

Rückkehrcodes

In der folgenden Tabelle werden die Rückkehrcodes für den Funktionsaufruf **dsmGetNextQObj** beschrieben.

Tabelle 35. Rückkehrcodes für den Funktionsaufruf **dsmGetNextQObj**.

Rückkehrcode	Rückkehrcodenummer	Beschreibung
DSM_RC_ABORT_NO_MATCH	2	Für die Abfrage wurde keine Übereinstimmung angefordert.

Tabelle 35. Rückkehrcodes für den Funktionsaufruf **dsmGetNextQObj** *(Forts.).*

Rückkehrcode	Rückkehrcodenummer	Beschreibung
DSM_RC_FINISHED	121	Die Verarbeitung ist beendet (**dsmEndQuery** starten). Es sind keine weiteren Daten zu verarbeiten.
DSM_RC_UNKNOWN_FORMAT	122	Die Datei, die von IBM Spectrum Protect zurückgeschrieben oder abgerufen werden sollte, hat ein unbekanntes Format.
DSM_RC_COMM_PROTOCOL_ERROR	136	Übertragungsprotokollfehler.
DSM_RC_NULL_DATABLKPTR	2001	Zeiger verweist nicht auf einen Datenblock.
DSM_RC_INVALID_MCNAME	2025	Ungültiger Verwaltungsklassenname.
DSM_RC_BAD_CALL_SEQUENCE	2041	Die Aufruffolge ist ungültig.
DSM_RC_WRONG_VERSION_PARM	2065	Die Version der Anwendungsclient-API stimmt nicht mit der Version der IBM Spectrum Protect-Bibliothek überein.
DSM_RC_MORE_DATA	2200	Es sind keine weiteren Daten zum Abrufen verfügbar.
DSM_RC_BUFF_TOO_SMALL	2210	Der Puffer ist zu klein.

dsmGetObj

Mit dem Funktionsaufruf **dsmGetObj** werden die angeforderten Objektdaten aus dem IBM Spectrum Protect-Datenstrom abgerufen und in den Puffer des Aufrufenden gestellt. Der Aufruf **dsmGetObj** verwendet die Objekt-ID, um das nächste Objekt oder das nächste Teilobjekt aus dem Datenstrom abzurufen.

Die Daten für das angegebene Objekt werden in den Puffer gestellt, auf den durch **DataBlk** verwiesen wird. Sind weitere Daten verfügbar, müssen Sie einen oder mehrere Aufrufe **dsmGetData** senden, um die verbleibenden Daten abzurufen, bis der Rückkehrcode DSM_RC_FINISHED zurückgegeben wird. Überprüfen Sie das Feld numBytes in **DataBlk**, um festzustellen, ob noch Daten im Puffer verblieben sind.

Die Objekte sollten in der Reihenfolge angefordert werden, in der sie im Aufruf **dsmBeginGetData** im Parameter **dsmGetList** aufgelistet sind. Eine Ausnahme von dieser Regel ist der Fall, wenn der Anwendungsclient ein Objekt in dem Datenstrom überspringen muss, um zu einem Objekt weiter hinten in der Liste zu gelangen. Handelt es sich bei dem Objekt, das durch die Objekt-ID angegeben wird, nicht um das nächste Objekt in dem Datenstrom, wird der Datenstrom verarbeitet, bis das Objekt lokalisiert oder der Datenstrom beendet ist. Diese Funktion sollte mit Bedacht eingesetzt werden, da möglicherweise ein großes Datenvolumen zum Lokalisieren des angeforderten Objekts verarbeitet und gelöscht werden muss.

Voraussetzung: Wenn **dsmGetObj** einen Fehlercode (NOT FINISHED oder MORE_DATA) zurückgibt, muss die Sitzung beendet werden, um die Zurückschreibungsoperation zu stoppen. Dies ist insbesondere dann wichtig, wenn Verschlüsselung verwendet und ein Rückkehrcode RC_ENC_WRONG_KEY empfangen wird. Sie müssen eine neue Sitzung mit dem korrekten Schlüssel starten.

Syntax

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

Parameter

dsUint32_t dsmHandle (I)
Die Kennung, die diesen Aufruf einem vorherigen Aufruf **dsmInitEx** zuordnet.

ObjID *objIdP (I)
Ein Verweis auf die ID des Objekts, das zurückgeschrieben werden soll.

DataBlk *dataBlkPtr (I/O)
Ein Verweis auf den Puffer, in den die zurückgeschriebenen Daten gestellt werden.

Rückkehrcodes

Die Rückkehrcodenummern sind in runden Klammern ( ) angegeben.

Tabelle 36. Rückkehrcodes für dsmGetObj

Rückkehrcode	Erläuterung
DSM_RC_ABORT_INVALID_OFFSET (33)	Der Offset, der während des Abrufs eines Teilobjekts angegeben wird, ist größer als die Länge des Objekts.
DSM_RC_ABORT_INVALID_LENGTH (34)	Die Länge, die während des Abrufs eines Teilobjekts angegeben wird, ist größer als die Länge des Objekts oder der Offset addiert zur Länge liegt hinter dem Ende des Objekts.
DSM_RC_FINISHED (121)	Die Verarbeitung ist beendet (**dsmEndGetObj** starten).
DSM_RC_WRONG_VERSION_PARM (2065)	Die API-Version des Anwendungsclients stimmt nicht mit der Version der IBM Spectrum Protect-Bibliothek überein.
DSM_RC_MORE_DATA (2200)	Es sind keine weiteren Daten zum Abrufen verfügbar.
RC_ENC_WRONG_KEY (4580)	Der im Aufruf **dsmInitEx** angegebene Schlüssel oder der gespeicherte Schlüssel stimmt nicht mit dem Schlüssel überein, der zum Verschlüsseln dieses Objekts verwendet wurde. Beenden Sie die Sitzung und geben Sie den korrekten Schlüssel an.

dsmGroupHandler

Mit dem Funktionsaufruf **dsmGroupHandler** wird abhängig von der bereitgestellten Eingabe eine Aktion für eine logische Dateigruppe ausgeführt. Der Client fasst eine Reihe einzelner Objekte in einer Gruppe zusammen, um diese auf dem IBM Spectrum Protect-Server als logische Gruppe zu referenzieren und zu verwalten.

Weitere Informationen finden Sie in „Dateigruppierung” auf Seite 67.

Syntax

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

Parameter

dsmGroupHandlerIn_t *dsmGroupHandlerInP (I)
Übergibt Gruppenattribute an die API.

groupType
Der Typ der Gruppe. Gültige Werte umfassen:
- DSM_GROUPTYPE_PEER - Peergruppe

actionType
Die Aktion, die ausgeführt werden soll. Gültige Werte umfassen:
- DSM_GROUP_ACTION_OPEN - erstellt eine neue Gruppe
- DSM_GROUP_ACTION_CLOSE - schreibt eine offene Gruppe fest und speichert sie
- DSM_GROUP_ACTION_ADD - fügt einer Gruppe ein Mitglied hinzu
- DSM_GROUP_ACTION_ASSIGNTO - ordnet ein Mitglied einer anderen Gruppe zu
- DSM_GROUP_ACTION_REMOVE - entfernt ein Mitglied aus der Gruppe

memberType
Der Gruppentyp des Objekts. Gültige Werte umfassen:
- DSM_MEMBERTYPE_LEADER - Gruppenleiter
- DSM_MEMBERTYPE_MEMBER - Gruppenmitglied

***uniqueGroupTagP**
Eine eindeutige Zeichenfolge-ID, die einer Gruppe zugeordnet ist.

leaderObjId
Die Objekt-ID für den Gruppenleiter.

***objNameP**
Ein Verweis auf den Objektnamen des Gruppenleiters.

memberObjList
Eine Liste der Objekte, die entfernt oder zugeordnet werden sollen.

dsmGroupHandlerOut_t *dsmGroupHandlerOutP (0)
Übergibt die Adresse der Struktur, die die API ausführt. Die Strukturversionsnummer wird zurückgegeben.

Rückkehrcodes

Die Rückkehrcodenummern sind in runden Klammern ( ) angegeben.

Tabelle 37. Rückkehrcodes für dsmGroupHandler

Rückkehrcode	Erläuterung
DSM_RC_ABORT_INVALID_GROUP_ACTION (237)	Es wurde versucht, eine ungültige Operation für einen Gruppenleiter oder ein Gruppenmitglied auszuführen.

dsmInit

Mit dem Funktionsaufruf **dsmInit** wird eine API-Sitzung gestartet und der Client mit IBM Spectrum Protect-Speicher verbunden. Für den Anwendungsclient kann jeweils nur eine einzige Sitzung zu einem bestimmten Zeitpunkt aktiv sein. Um eine weitere Sitzung mit anderen Parametern zu öffnen, beenden Sie die aktuelle Sitzung zunächst mit dem Aufruf **dsmTerminate**.

Um die knotenübergreifende Abfrage sowie die Zurückschreibung oder den Abruf zu ermöglichen, verwenden Sie die Zeichenfolgeoptionen *-fromnode* und *-fromowner*. Weitere Informationen finden Sie in „Knoten- und eignerübergreifender Zugriff auf Objekte" auf Seite 25.

Syntax

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

Parameter

dsUint32_t *dsmHandle (O)
Die Kennung, die diese Initialisierungssitzung identifiziert und nachfolgenden IBM Spectrum Protect-Aufrufen zuordnet.

dsmApiVersion *dsmApiVersionP (I)
Ein Verweis auf die Datenstruktur, die die Version der API identifiziert, die der Anwendungsclient für diese Sitzung verwendet. Die Struktur enthält die Werte der drei Konstanten DSM_API_VERSION, DSM_API_RELEASE und DSM_API-_LEVEL, die in der Datei dsmapitd.h definiert sind. Ein vorheriger Aufruf **dsmQueryApiVersion** muss ausgeführt werden, um sicherzustellen, dass zwischen der API-Version des Anwendungsclients und der Version der API-Bibliothek, die auf der Workstation des Benutzers installiert ist, Kompatibilität besteht.

char *clientNodeNameP (I)
Dieser Parameter ist ein Verweis auf den Knoten für die IBM Spectrum Protect-Sitzung. Allen Sitzungen muss ein Knotenname zugeordnet sein. Mit der Konstanten DSM_MAX_NODE_LENGTH in der Datei dsmapitd.h wird die maximale Größe festgelegt, die für einen Knotennamen zulässig ist.

Beim Knotennamen muss die Groß-/Kleinschreibung nicht beachtet werden.

Wenn dieser Parameter auf NULL und die Option *passwordaccess* auf *prompt* gesetzt wird, versucht die API, den Knotennamen zunächst aus der übergebenen Optionszeichenfolge abzurufen. Ist er dort nicht vorhanden, versucht die API anschließend, den Knotennamen aus der Konfigurationsdatei oder der Optionsdatei abzurufen. Schlagen diese Versuche, den Knotennamen zu finden, fehl, verwendet die UNIX- oder Linux-API den Systemhostnamen, während die APIs unter anderen Betriebssystemen den Code DSM_RC_REJECT_I-D_UNKNOWN zurückgeben.

Dieser Parameter muss NULL sein, wenn die Option *passwordaccess* in der Datei dsm.sys auf *generate* gesetzt ist. Die API verwendet den Systemhostnamen.

char *clientOwnerNameP (I)
Dieser Parameter ist ein Verweis auf den Eigner der IBM Spectrum Protect-Sitzung. Wenn das Betriebssystem, unter dem die Sitzung gestartet wird, ein Mehrbenutzerbetriebssystem ist, hat der Eignername NULL (der Rootbenutzer) die Berechtigung zum Sichern, Archivieren, Zurückschreiben oder Abrufen von Objekten, die zu der Anwendung gehören, unabhängig vom Eigner des Objekts.

Beim Eignernamen muss die Groß-/Kleinschreibung beachtet werden.

Dieser Parameter muss NULL sein, wenn die Option *passwordaccess* in der Datei dsm.sys auf *generate* gesetzt ist. Die API verwendet dann die Anmelde-Benutzer-ID.

Anmerkung: Wenn bei einem Mehrbenutzerbetriebssystem die Option *passwordaccess* auf *prompt* wird, muss der Eignername nicht mit der ID des aktiven Benutzers der Sitzung übereinstimmen, die die Anwendung ausführt.

char *clientPasswordP (I)
Dieser Parameter ist ein Verweis auf das Kennwort des Knotens, in dem die IBM Spectrum Protect-Sitzung ausgeführt wird. Mit der Konstanten DSM-_MAX_VERIFIER_LENGTH in der Datei dsmapitd.h wird die maximale Größe festgelegt, die für ein Kennwort zulässig ist.

Beim Kennwort muss die Groß-/Kleinschreibung nicht beachtet werden.

Außer wenn die Kennwortdatei zuerst gestartet wird, wird der Wert dieses Parameters ignoriert, wenn die Option *passwordaccess* auf *generate* gesetzt ist.

char *applicationType (I)
Dieser Parameter identifiziert die Anwendung, die die Sitzung ausführt. Der Wert wird vom Anwendungsclient definiert.

Jedes Mal, wenn ein API-Anwendungsclient eine Sitzung mit dem Server startet, wird der Anwendungstyp (bzw. die Plattform) des Clients auf dem Server aktualisiert. Der Wert für den Anwendungstyp sollte eine Abkürzung für das Betriebssystem enthalten, da dieser Wert in das Feld **platform** auf dem Server eingegeben wird. Die maximale Länge für die Zeichenfolge ist DSM_MAX-_PLATFORM_LENGTH.

Der aktuelle Wert für den Anwendungstyp kann mithilfe von **dsmQuerySessInfo** angezeigt werden.

char *configfile (I)
Dieser Parameter verweist auf eine Zeichenfolge, die den vollständig qualifizierten Namen einer API-Konfigurationsdatei enthält. Optionen, die in der API-Konfigurationsdatei angegeben werden, überschreiben die entsprechende Angabe in der Clientoptionsdatei. Optionsdateien werden beim Installieren von IBM Spectrum Protect (Client oder API) installiert.

char *options (I)
Verweist auf eine Zeichenfolge, die Benutzeroptionen wie die folgenden enthalten kann:

- *Compressalways*
- *Servername* (nur UNIX oder Linux)
- *TCPServeraddr*
- *Fromnode*
- *Fromowner*
- *EnableClientEncryptKey*

Der Anwendungsclient kann die Werte für diese Optionen, die in der Konfigurationsdatei definiert sind, mithilfe der Optionsliste überschreiben.

Die Optionen haben das folgende Format:

1. Jede Option, die in der Optionsliste angegeben ist, beginnt mit einem Gedankenstrich (-) gefolgt vom Optionsschlüsselwort.
2. Auf das Schlüsselwort wiederum folgt ein Gleichheitszeichen (=) gefolgt vom Optionsparameter.
3. Wenn der Optionsparameter ein Leerzeichen enthält, schließen Sie den Parameter in Hochkommas oder Anführungszeichen ein.
4. Wenn mehrere Optionen angegeben werden, trennen Sie die einzelnen Optionen jeweils durch ein Leerzeichen voneinander.

Wenn Optionen NULL sind, werden die Werte für alle Optionen aus der Benutzeroptionsdatei oder der API-Konfigurationsdatei übernommen.

Rückkehrcodes

Die Rückkehrcodenummern sind in runden Klammern ( ) angegeben.

Tabelle 38. Rückkehrcodes für dsmInit

Rückkehrcode	Erläuterung
DSM_RC_ABORT_SYSTEM_ERROR (1)	Der Server hat einen Systemfehler erkannt und die Clients benachrichtigt.
DSM_RC_REJECT_VERIFIER_EXPIRED (52)	Das Kennwort ist abgelaufen und muss aktualisiert werden.
DSM_RC_REJECT_ID_UNKNOWN (53)	Der Knotenname konnte nicht gefunden werden.
DSM_RC_AUTH_FAILURE (137)	Es ist ein Authentifizierungsfehler aufgetreten.
DSM_RC_NO_STARTING_DELIMITER (148)	Das Muster enthält keinen Startbegrenzer.
DSM_RC_NEEDED_DIR_DELIMITER (149)	Direkt vor und hinter der Metazeichenfolge („...") für den „Verzeichnisabgleich" ist ein Verzeichnisbegrenzer erforderlich, es wurde jedoch keiner gefunden.
DSM_RC_NO_PASS_FILE (168)	Die Kennwortdatei ist nicht verfügbar.
DSM_RC_UNMATCHED_QUOTE (177)	Die Optionszeichenfolge enthält ein Anführungszeichen ohne Entsprechung.
DSM_RC_NLS_CANT_OPEN_TXT (0610)	Die Nachrichtentextdatei kann nicht geöffnet werden.
DSM_RC_INVALID_OPT (400)	Ein Eintrag in der Optionszeichenfolge ist ungültig.
DSM_RC_INVALID_DS_HANDLE (2014)	Ungültige DSM-Kennung.
DSM_RC_NO_OWNER_REQD (2032)	Der Eignerparameter muss NULL sein, wenn die Option *passwordaccess* auf *generate* gesetzt ist.
DSM_RC_NO_NODE_REQD (2033)	Der Knotenparameter muss NULL sein, wenn die Option *passwordaccess* auf *generate* gesetzt ist.
DSM_RC_WRONG_VERSION (2064)	Die API-Version für den Anwendungsclient hat einen höheren Wert als die Version von IBM Spectrum Protect.
DSM_RC_PASSWD_TOOLONG (2103)	Das angegebene Kennwort ist zu lang.
DSM_RC_NO_OPT_FILE (2220)	Es konnte keine Konfigurationsdatei lokalisiert werden.
DSM_RC_INVALID_KEYWORD (2221)	Ein in einer Optionszeichenfolge angegebenes Schlüsselwort ist ungültig.
DSM_RC_PATTERN_TOO_COMPLEX (2222)	Das Einschluss-/Ausschlussmuster ist zu komplex und kann daher nicht von IBM Spectrum Protect interpretiert werden.
DSM_RC_NO_CLOSING_BRACKET (2223)	Im Muster fehlt die rechte eckige Klammer.
DSM_RC_INVALID_SERVER (2225)	Für eine Mehrplatzsystemumgebung wurde der Server in der Systemkonfigurationsdatei nicht gefunden.
DSM_RC_NO_HOST_ADDR (2226)	Nicht genügend Informationen, um die Verbindung zum Host herzustellen.
DSM_RC_MACHINE_SAME (2227)	Der Knotenname, der in der Optionsdatei definiert ist, darf nicht mit dem Systemhostnamen übereinstimmen.
DSM_RC_NO_API_CONFIGFILE (2228)	Die Konfigurationsdatei kann nicht geöffnet werden.
DSM_RC_NO_INCLEXCL_FILE (2229)	Die Einschluss-/Ausschlussdatei wurde nicht gefunden.
DSM_RC_NO_SYS_OR_INCLEXCL (2230)	Die Datei dsm.sys oder die Einschluss-/Ausschlussdatei wurde nicht gefunden.

Zugehörige Konzepte:

Übersicht über die Clientoptionsdatei

Verarbeitungsoptionen

---

dsmInitEx

Mit dem Funktionsaufruf **dsmInitEx** wird eine API-Sitzung gestartet, wobei die zusätzlichen Parameter für die erweiterte Prüfung verwendet werden.

Syntax

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

Parameter

dsUint32_t *dsmHandleP (O)
Die Kennung, die diese Initialisierungssitzung identifiziert und nachfolgenden IBM Spectrum Protect-Aufrufen zuordnet.

dsmInitExIn_t *dsmInitExInP
Diese Struktur enthält die folgenden Eingabeparameter:

dsmApiVersion *dsmApiVersionP (I)
Dieser Parameter ist ein Verweis auf die Datenstruktur, die die Version der API identifiziert, die der Anwendungsclient für diese Sitzung verwendet. Die Struktur enthält die Werte für die vier Konstanten DSM_API_VERSION, DSM_API_RELEASE, DSM_API_LEVEL und DSM_API_SUBLEVEL, die in der Datei `dsmapitd.h` definiert werden. Rufen Sie **dsmQueryApiVersionEx** auf und stellen Sie sicher, dass die API-Version des Anwendungsclients und die Version der API-Bibliothek, die auf der Workstation des Benutzers installiert ist, kompatibel sind.

char *clientNodeNameP (I)
Dieser Parameter ist ein Verweis auf den Knoten für die IBM Spectrum Protect-Sitzung. Alle Sitzungen müssen einem Knotennamen zugeordnet sein. Die Konstante DSM_MAX_NODE_LENGTH in der Datei `dsmapitd.h` legt die maximale Größe für einen Knotennamen fest.

Beim Knotennamen muss die Groß-/Kleinschreibung nicht beachtet werden.

Wenn dieser Parameter auf NULL und die Option **passwordaccess** auf `prompt` gesetzt wird, versucht die API, den Knotennamen zunächst aus der übergebenen Optionszeichenfolge abzurufen. Ist er dort nicht vorhanden, versucht die API anschließend, den Knotennamen aus der Konfigurationsdatei oder der Optionsdatei abzurufen. Schlagen diese Versuche, den Knotennamen zu finden, fehl, verwendet die UNIX- oder Linux-API den Systemhostnamen, während die APIs unter anderen Betriebssystemen `DSM_RC_REJECT_ID_UNKNOWN` zurückgeben.

Dieser Parameter muss NULL sein, wenn die Option **passwordaccess** in der Datei `dsm.sys` auf `generate` gesetzt ist. Die API verwendet dann den Systemhostnamen.

char *clientOwnerNameP (I)
Dieser Parameter ist ein Verweis auf den Eigner der IBM Spectrum Protect-Sitzung. Wenn das Betriebssystem eine Mehrbenutzerplattform ist, hat der Eignername NULL (der Rootbenutzer) die Berechtigung zum Sichern, Ar-

chivieren, Zurückschreiben oder Abrufen von Objekten, die zu der Anwendung gehören, unabhängig vom Eigner des Objekts.

Beim Eignernamen muss die Groß-/Kleinschreibung beachtet werden.

Dieser Parameter muss NULL sein, wenn die Option **passwordaccess** in der Datei dsm.sys auf generate gesetzt ist. Die API verwendet dann die Anmelde-Benutzer-ID.

Tipp: Wenn bei einer Mehrbenutzerplattform die Option **passwordaccess** auf prompt gesetzt wird, muss der Eignername nicht mit der ID des aktiven Benutzers der Sitzung übereinstimmen, die die Anwendung ausführt.

char *clientPasswordP (I)
Ein Verweis auf das Kennwort des Knotens, in dem die IBM Spectrum Protect-Sitzung ausgeführt wird. Mit der Konstanten DSM_MAX_VERIFIER_LENGTH in der Datei dsmapitd.h wird die maximale Größe festgelegt, die für ein Kennwort zulässig ist.

Beim Kennwort muss die Groß-/Kleinschreibung nicht beachtet werden.

Außer wenn die Kennwortdatei zuerst gestartet wird, wird der Wert dieses Parameters ignoriert, wenn die Option **passwordaccess** auf generate gesetzt ist.

char *userNameP;
Ein Verweis auf den Namen des Benutzers mit Verwaltungsaufgaben, der über Clientberechtigung für diesen Knoten verfügt.

char *userPasswordP;
Ein Verweis auf das Kennwort für den Parameter **userName**, sofern ein Wert angegeben ist.

char *applicationType (I)
Identifiziert die Anwendung, die die IBM Spectrum Protect-Sitzung ausführt. Der Anwendungsclient identifiziert den Wert.

Jedes Mal, wenn ein API-Anwendungsclient eine Sitzung mit dem Server startet, wird der Anwendungstyp (bzw. das Betriebssystem) des Clients auf dem Server aktualisiert. Der Wert wird in das Feld **platform** auf dem Server eingegeben. Ziehen Sie die Verwendung einer Betriebssystem-ID im Wert in Betracht. Die maximale Länge der Zeichenfolge wird in der Konstanten DSM_MAX_PLATFORM_LENGTH definiert.

Der aktuelle Wert für den Anwendungstyp kann mithilfe von **dsmQuerySessInfo** angezeigt werden.

char *configfile (I)
Verweist auf eine Zeichenfolge, die den vollständig qualifizierten Namen einer API-Konfigurationsdatei enthält. Optionen, die in der API-Konfigurationsdatei angegeben werden, überschreiben die entsprechende Angabe in der Clientoptionsdatei. Optionsdateien werden beim Installieren von IBM Spectrum Protect (Client oder API) installiert.

char *options (I)
Verweist auf eine Zeichenfolge, die Benutzeroptionen wie die folgenden enthalten kann:
- Compressalways
- Servername (nur UNIX- und Linux-Systeme)
- TCPServeraddr (nicht für UNIX-Systeme)
- Fromnode
- Fromowner

Der Anwendungsclient kann die Werte für diese Optionen, die in der Konfigurationsdatei definiert sind, mithilfe der Optionsliste überschreiben.

Optionen weisen das folgende Format auf:

1. Jede Option, die in der Optionsliste angegeben ist, beginnt mit einem Gedankenstrich (-) gefolgt vom Optionsschlüsselwort.
2. Auf das Schlüsselwort folgt ein Gleichheitszeichen (=) und dann der Optionsparameter.
3. Wenn der Optionsparameter ein Leerzeichen enthält, schließen Sie den Parameter in Hochkommas oder Anführungszeichen ein.
4. Wenn mehrere Optionen angegeben werden, trennen Sie die einzelnen Optionen jeweils durch ein Leerzeichen voneinander.

Wenn Optionen NULL sind, werden die Werte für alle Optionen aus der Benutzeroptionsdatei oder der API-Konfigurationsdatei übernommen.

dirDelimiter
Der Verzeichnisbegrenzer, der als Präfix vor Dateibereichsnamen, übergeordneten und untergeordneten Namen angegeben wird. Sie müssen den Parameter **dirDelimiter** nur angeben, wenn die Anwendung die Systemstandardwerte überschreibt. In einer UNIX- oder Linux-Umgebung bildet der Schrägstrich (/) den Standardwert. In einer Windows-Umgebung bildet der umgekehrte Schrägstrich (\) den Standardwert.

useUnicode
Ein boolesches Flag, das angibt, ob Unicode aktiviert ist. Das Flag **useUnicode** muss auf `false` gesetzt sein, um die plattformübergreifende Interoperabilität zwischen UNIX- und Windows-Systemen zu ermöglichen.

bCrossPlatform
Ein boolesches Flag, das gesetzt werden muss (`bTrue`), um plattformübergreifende Interoperabilität zwischen UNIX- und Windows-Systemen zu ermöglichen. Wenn das Flag `bCrossPlatform` gesetzt ist, stellt die API sicher, dass für die Dateibereiche nicht Unicode verwendet wird und die Anwendung nicht Unicode verwendet. Eine Windows-Anwendung, die Unicode verwendet, ist nicht mit Anwendungen kompatibel, die Nicht-Unicode-Codierungen verwenden. Das Flag `bCrossPlatform` darf nicht für eine Windows-Anwendung gesetzt werden, die Unicode verwendet.

UseTsmBuffers
Gibt an, ob die Pufferkopieneliminierung verwendet werden soll.

numTsmBuffers
Anzahl Puffer, wenn `useTsmBuffers=bTrue` gesetzt ist.

bEncryptKeyEnabled
Gibt an, ob Verschlüsselung mit anwendungsverwaltetem Schlüssel verwendet wird.

encryptionPasswordP
Das Verschlüsselungskennwort.

Einschränkung: Wenn `encryptkey=save` gesetzt und ein Verschlüsselungsschlüssel vorhanden ist, wird der in **encryptionPasswordP** angegebene Wert ignoriert.

dsmAppVersion *appVersionP (I)
Dieser Parameter ist ein Verweis auf die Datenstruktur, die die Versionsinformationen der Anwendung enthält, die eine API-Sitzung startet. Die

Struktur enthält die Werte für die vier Konstanten applicationVersion, applicationRelease, applicationLevel und applicationSubLevel, die in der Datei `tsmapitd.h` definiert werden.

dsmInitExOut_t *dsmInitExOut P
Diese Struktur enthält die Ausgabeparameter.

dsUint32_t *dsmHandle (0)
Die Kennung, die diese Initialisierungssitzung identifiziert und nachfolgenden API-Aufrufen zuordnet.

infoRC
Zusätzliche Informationen zum Rückkehrcode. Überprüfen Sie sowohl den Funktionsrückkehrcode als auch den Wert von **infoRC**. Wenn **infoRC** den Wert DSM_RC_REJECT_LASTSESS_CANCELED (69) hat, gibt IBM Spectrum Protect an, dass der Administrator die letzte Sitzung abgebrochen hat.

Rückkehrcodes

Die Rückkehrcodenummern sind in runden Klammern ( ) angegeben.

Tabelle 39. Rückkehrcodes für **dsmInitEx**

Rückkehrcode	Erläuterung
DSM_RC_ABORT_SYSTEM_ERROR (1)	Der IBM Spectrum Protect-Server hat einen Systemfehler erkannt und die Clients benachrichtigt.
DSM_RC_REJECT_VERIFIER_EXPIRED (52)	Das Kennwort ist abgelaufen und muss aktualisiert werden. Der nächste Aufruf muss **dsmChangePW** sein und in diesem Aufruf muss die Kennung zurückgegeben werden.
DSM_RC_REJECT_ID_UNKNOWN (53)	Der Knotenname wurde nicht gefunden.
DSM_RC_TA_COMM_DOWN (103)	Die Datenübertragungsverbindung ist inaktiv.
DSM_RC_AUTH_FAILURE (137)	Es ist ein Authentifizierungsfehler aufgetreten.
DSM_RC_NO_STARTING_DELIMITER (148)	Das Muster enthält keinen Startbegrenzer.
DSM_RC_NEEDED_DIR_DELIMITER (149)	Direkt vor und hinter der Metazeichenfolge („...") für den „Verzeichnisabgleich" ist ein Verzeichnisbegrenzer erforderlich, es wurde jedoch keiner gefunden.
DSM_RC_NO_PASS_FILE (168)	Die Kennwortdatei ist nicht verfügbar.
DSM_RC_UNMATCHED_QUOTE (177)	Die Optionszeichenfolge enthält ein Anführungszeichen ohne Entsprechung.
DSM_RC_NLS_CANT_OPEN_TXT (0610)	Die Nachrichtentextdatei kann nicht geöffnet werden.
DSM_RC_INVALID_OPT (2013)	Ein Eintrag in der Optionszeichenfolge ist ungültig.
DSM_RC_INVALID_DS_HANDLE (2014)	Ungültige DSM-Kennung.
DSM_RC_NO_OWNER_REQD (2032)	Der Eignerparameter muss NULL sein, wenn die Option **passwordaccess** auf `generate` gesetzt ist.
DSM_RC_NO_NODE_REQD (2033)	Der Knotenparameter muss NULL sein, wenn **passwordaccess** auf 'generate' gesetzt ist.
DSM_RC_WRONG_VERSION (2064)	Die API-Version des Anwendungsclients hat einen höheren Wert als die Version von IBM Spectrum Protect.
DSM_RC_PASSWD_TOOLONG (2103)	Das angegebene Kennwort ist zu lang.
DSM_RC_NO_OPT_FILE (2220)	Es wurde keine Konfigurationsdatei gefunden.
DSM_RC_INVALID_KEYWORD (2221)	Ein in einer Optionszeichenfolge angegebenes Schlüsselwort ist ungültig.

Tabelle 39. Rückkehrcodes für **dsmInitEx** *(Forts.)*

Rückkehrcode	Erläuterung
DSM_RC_PATTERN_TOO_COMPLEX (2222)	Das Einschluss-/Ausschlussmuster ist zu komplex und kann daher nicht von IBM Spectrum Protect interpretiert werden.
DSM_RC_NO_CLOSING_BRACKET (2223)	Im Muster fehlt die rechte eckige Klammer.
DSM_RC_INVALID_SERVER (2225)	Für eine Mehrplatzsystemumgebung wurde der Server in der Systemkonfigurationsdatei nicht gefunden.
DSM_RC_NO_HOST_ADDR (2226)	Nicht genügend Informationen, um die Verbindung zum Host herzustellen.
DSM_RC_MACHINE_SAME (2227)	Der Knotenname, der in der Optionsdatei definiert ist, darf nicht mit dem Systemhostnamen übereinstimmen.
DSM_RC_NO_API_CONFIGFILE (2228)	Die Konfigurationsdatei kann nicht geöffnet werden.
DSM_RC_NO_INCLEXCL_FILE (2229)	Die Einschluss-/Ausschlussdatei wurde nicht gefunden.
DSM_RC_NO_SYS_OR_INCLEXCL (2230)	Die Datei `dsm.sys` oder die Einschluss-/Ausschlussdatei wurde nicht gefunden.

Zugehörige Konzepte:

Übersicht über die Clientoptionsdatei

Verarbeitungsoptionen

dsmLogEvent

Mit dem Funktionsaufruf **dsmLogEvent** wird eine Benutzernachricht (ANE4991I) in der Serverprotokolldatei und/oder im lokalen Fehlerprotokoll protokolliert. Eine Struktur des Typs **logInfo** wird in dem Aufruf übergeben. Dieser Aufruf muss während des Zustands **InSession** innerhalb einer Sitzung ausgeführt werden. Er darf nicht in einer Sende-, Abruf- oder Abfrageoperation ausgeführt werden. Um Nachrichten abzurufen, die auf dem Server protokolliert sind, verwenden Sie den Befehl **query actlog** über den Verwaltungsclient.

Siehe das Zusammenfassungszustandsdiagramm, Abb. 20 auf Seite 81.

Syntax

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

Parameter

dsUint32_t dsmHandle(I)
Die Kennung, die diesen Aufruf einem vorherigen Aufruf **dsmInitEx** zuordnet.

logInfo *logInfoP (I)
Übergibt die Nachricht und das Ziel. Der Anwendungsclient ist für das Zuordnen von Speicher für die Struktur zuständig.

Die Struktur **logInfo** enthält folgende Felder:

message
Der Text der Nachricht, die protokolliert werden soll. Dabei muss es sich um eine Zeichenfolge handeln, die auf null endet. Die maximale Länge ist DSM_MAX_RC_MSG_LENGTH.

dsmLogtype
Gibt an, wo die Nachricht protokolliert werden soll. Gültige Werte umfassen: **logServer**, **logLocal**, **logBoth**.

Rückkehrcodes

Die Rückkehrcodenummern sind in runden Klammern ( ) angegeben.

Tabelle 40. Rückkehrcodes für dsmLogEvent

Rückkehrcode	Erläuterung
DSM_RC_STRING_TOO_LONG (2120)	Die Nachrichtenzeichenfolge ist zu lang.

dsmLogEventEx

Mit dem Funktionsaufruf **dsmLogEventEx** wird eine Benutzernachricht in der Serverprotokolldatei und/oder im lokalen Fehlerprotokoll protokolliert. Dieser Aufruf muss während des Zustands **InSession** innerhalb einer Sitzung ausgeführt werden. Der Aufruf kann nicht in einem Sende-, Abruf- oder Abfrageaufruf durchgeführt werden.

Zusammenfassungszustandsdiagramm: Eine Übersicht über die Sitzungsinteraktionen finden Sie im Zusammenfassungszustandsdiagramm hier:

Abb. 20 auf Seite 81

Die IBM Spectrum Protect-Nachrichtennummer wird durch die Dringlichkeit bestimmt. Um Nachrichten anzuzeigen, die auf dem Server protokolliert sind, verwenden Sie den Befehl **query actlog** über den Verwaltungsclient. Verwenden Sie die IBM Spectrum Protect-Clientoption `errorlogretention`, um die Clientfehlerprotokolldatei zu bereinigen, wenn die Anwendung zahlreiche Clientnachrichten in das Clientprotokoll schreibt (`dsmLogType` ist entweder `logLocal` oder `logBoth`). Weitere Informationen finden Sie in der Dokumentation für den IBM Spectrum Protect-Server.

Syntax

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

Parameter

dsUint32_t dsmHandle(I)
Die Kennung, die diesen Aufruf einem vorherigen Aufruf **dsmInitEx** zuordnet.

dsmLogExIn_t *dsmLogExInP
Diese Struktur enthält die Eingabeparameter.

dsmLogSeverity severity;
Dieser Parameter gibt die Ereignisdringlichkeit an. Gültige Werte sind:

```
logSevInfo,       /* Information   ANE4990 */
logSevWarning,    /* Warnung       ANE4991 */
logSevError,      /* Fehler        ANE4992 */
logSevSevere      /* schwerwiegend ANE4993 */
```

char appMsgID[8];
Dieser Parameter ist eine Zeichenfolge zum Identifizieren der spezifischen

Anwendungsnachricht. Ein geeignetes Format sind drei Zeichen, gefolgt von vier Zahlen, beispielsweise DSM0250.

dsmLogType logType;
Dieser Parameter gibt an, wohin das Ereignis übertragen werden soll. Für diesen Parameter gibt es die folgenden gültigen Werte:

- logServer
- logLocal
- logBoth

char *message;
Dieser Parameter gibt den Text der zu protokollierenden Ereignisnachricht an. Bei dem Text muss es sich um eine Zeichenfolge handeln, die auf null endet. Die maximale Länge ist DSM_MAX_RC_MSG_LENGTH.

Einschränkung: Nachrichten, die an den Server übertragen werden, müssen in Englisch abgefasst sein. Nachrichten in anderen Sprachen als Englisch werden nicht korrekt angezeigt.

dsmLogExOut_t *dsmLogExOutP
Diese Struktur enthält die Ausgabeparameter. Momentan sind keine Ausgabeparameter definiert.

Rückkehrcodes

Die Rückkehrcodenummern sind in runden Klammern ( ) angegeben.

Tabelle 41. Rückkehrcodes für dsmLogEventEx

Rückkehrcode	Erläuterung
DSM_RC_STRING_TOO_LONG (2120)	Die Nachrichtenzeichenfolge ist zu lang.

dsmQueryAccess

Mit dem Funktionsaufruf **dsmQueryAccess** wird der Server nach allen Zugriffsberechtigungsregeln für Sicherungsversionen oder archivierte Kopien Ihrer Objekte abgefragt. Ein Verweis auf ein Array von Zugriffsregeln wird an den Aufruf übergeben und das fertige Array zurückgegeben. Es wird ein Verweis auf die Anzahl Regeln übergeben, um anzugeben, wie viele Regeln das Array enthält.

Es gibt keine spezifischen Rückkehrcodes für diesen Aufruf.

Syntax

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

Parameter

dsUint32_t dsmHandle (I)
Die Kennung, die diesen Aufruf einem vorherigen Aufruf **dsmInitEx** zuordnet.

qryRespAccessData **accessListP (O)
Ein Verweis auf einen Array von Elementen qryRespAccessData, die die API-Bibliothek zuordnet. Jedes Element entspricht einer Zugriffsregel. Die Anzahl

Elemente in dem Array wird im Parameter **numberOfRules** zurückgegeben. Die Informationen, die in jedem Element qryRespAccessData zurückgegeben werden, umfassen Folgendes:

Name	Beschreibung
ruleNumber	Die ID für die Zugriffsregel. Diese kennzeichnet die Regel zum Löschen.
AccessType	Der Sicherungs- oder Archivierungstyp.
Node	Der Knoten, für den Zugriff erteilt wurde.
Owner	Der Benutzer, dem Zugriff erteilt wurde.
objName	Die übergeordneten oder untergeordneten Dateibereichsdeskriptoren.

dsUint32_t *numberOfRules (0)
Gibt die Anzahl Regeln in dem Array accessList zurück.

dsmQueryApiVersion

Mit dem Funktionsaufruf **dsmQueryApiVersion** wird eine Abfrageanforderung für die API-Bibliotheksversion ausgeführt, auf die der Anwendungsclient zugreift.

Alle Aktualisierungen an der API erfolgen in aufwärtskompatiblem Format. Jeder Anwendungsclient mit einer API-Version oder einem API-Release kleiner-gleich der Version bzw. des Release der API-Bibliothek auf der Workstation des Endbenutzers wird ohne Änderung ausgeführt. Sollte der Aufruf **dsmQueryApiVersion** eine ältere Version oder ein älteres Versionsrelease als die bzw. das des Anwendungsclients zurückgeben, müssen Sie vor dem Fortfahren beachten, dass einige API-Aufrufe möglicherweise auf eine Art und Weise erweitert wurden, die nicht von der älteren API-Version des Endbenutzers unterstützt wird.

Die API-Versionsnummer der Anwendung ist in der Datei dsmapitd.h als die Konstanten DSM_API_VERSION, DSM_API_RELEASE und DSM_API_LEVEL gespeichert.

Es gibt keine spezifischen Rückkehrcodes für diesen Aufruf.

Syntax

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

Parameter

dsmApiVersion *apiVersionP (0)
Dieser Parameter ist ein Verweis auf die Struktur, die die Version, das Release und die Stufe der API-Bibliothek enthält. Wenn die Bibliothek beispielsweise die Version 1.1.0 hat, enthalten die Felder der Struktur nach der Rückkehr des Aufrufs die folgenden Werte:

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

dsmQueryApiVersionEx

Mit dem Funktionsaufruf **dsmQueryApiVersionEx** wird eine Abfrageanforderung für die API-Bibliotheksversion ausgeführt, auf die der Anwendungsclient zugreift.

Alle Aktualisierungen an der API erfolgen in aufwärtskompatiblem Format. Jeder Anwendungsclient mit einer API-Version oder einem API-Release kleiner-gleich der Version bzw. des Release der API-Bibliothek auf der Workstation des Endbenutzers wird ohne Änderung ausgeführt. Die Zusammenfassung der Codeänderungen in der Datei README_api_enu gibt Auskunft über die Ausnahmen in Bezug auf die Aufwärtskompatibilität. Wenn der Aufruf **dsmQueryApiVersionEx** eine andere Version oder ein anderes Versionsrelease zurückgibt als die bzw. das des Anwendungsclients, müssen Sie vor dem Fortfahren beachten, dass einige API-Aufrufe möglicherweise auf eine Art und Weise erweitert wurden, die nicht von der älteren API-Version des Endbenutzers unterstützt wird.

Die API-Versionsnummer der Anwendung ist in der Headerdatei dsmapitd.h als die Konstanten DSM_API_VERSION, DSM_API_RELEASE, DSM_API_LEVEL und DSM_API_SUBLEVEL gespeichert.

Es gibt keine spezifischen Rückkehrcodes für diesen Aufruf.

Syntax

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

Parameter

dsmApiVersionEx *apiVersionP (O)
Dieser Parameter ist ein Verweis auf die Struktur, die die Version, das Release, die Stufe und die Unterstufe der API-Bibliothek enthält. Wenn die Bibliothek beispielsweise die Version 5.5.0.0 hat, enthalten die Felder der Struktur nach der Rückkehr des Aufrufs die folgenden Werte:

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

dsmQueryCliOptions

Mit dem Funktionsaufruf **dsmQueryCliOptions** werden wichtige Optionswerte in der Benutzeroptionsdatei abgefragt. Eine Struktur des Typs **optStruct** wird in dem Aufruf übergeben und enthält die Informationen. Dieser Aufruf wird ausgeführt, bevor **dsmInitEx** aufgerufen wird; er legt die Konfiguration vor der Sitzung fest.

Es gibt keine spezifischen Rückkehrcodes für diesen Aufruf.

Syntax

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

Parameter

optStruct *optstructP (I/O)
Dieser Parameter übergibt die Adresse der Struktur, die die API ausführt. Der Anwendungsclient ist für das Zuordnen von Speicher für die Struktur zuständig. Bei der erfolgreichen Rückkehr enthalten die Felder in der Struktur die entsprechenden Informationen.

Die folgenden Informationen werden in der **optStruct**-Struktur zurückgegeben:

Name	Beschreibung
dsmiDir	Der Wert der Umgebungsvariablen DSMI_DIR.
dsmiConfig	Die in der Umgebungsvariablen DSMI_CONFIG angegebene Clientoptionsdatei.
serverName	Der Name des IBM Spectrum Protect-Servers.
commMethod	Die ausgewählte Übertragungsmethode. Siehe #defines für DSM_COMM_* in der Datei dsmapitd.h.
serverAddress	Die Adresse des Servers auf der Basis der Übertragungsmethode.
nodeName	Der Name des Clientknotens (Maschine).
compression	Dieses Feld enthält Informationen, die die Komprimierungsoption betreffen.
passwordAccess	Gültige Werte sind: *bTrue* für 'generate' und *bFalse* für 'prompt'.

Zugehörige Konzepte:

Verarbeitungsoptionen

dsmQuerySessInfo

Mit dem Funktionsaufruf **dsmQuerySessInfo** wird eine Anforderung zum Abfragen von Informationen an IBM Spectrum Protect gestartet, die sich auf die Operation der angegebenen Sitzung in **dsmHandle** beziehen. Eine Struktur des Typs **ApiSessInfo** wird in dem Aufruf übergeben; sie enthält alle verfügbaren eingegebenen sitzungsbezogenen Informationen. Dieser Aufruf wird nach einem erfolgreichen Aufruf **dsmInitEx** gestartet.

Die Informationen, die in der Struktur des Typs **ApiSessInfo** zurückgegeben werden, umfassen Folgendes:
- Serverinformationen: Portnummer, Datum und Uhrzeit sowie Typ
- Clientstandardwerte: Anwendungstyp, Löschberechtigungen, Begrenzer und Transaktionsgrenzen
- Sitzungsdaten: Anmelde-ID und Eigner
- Maßnahmendaten: Domäne, aktive Maßnahmengruppe und Aufbewahrungszeitraum

Informationen zum Inhalt der übergebenen Struktur und jedes Felds in der Struktur finden Sie in Anhang B, „Quellendateien für API-Typdefinitionen", auf Seite 171.

Syntax

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

Parameter

dsUint32_t dsmHandle (I)
Die Kennung, die diesen Aufruf einem vorherigen Aufruf **dsmInitEx** zuordnet.

ApiSessInfo *SessInfoP (I/O)
Dieser Parameter übergibt die Adresse der Struktur, die die API ausführt. Der Anwendungsclient ist für das Zuordnen von Speicher für die Struktur zuständig und für das Eingeben der Feldeinträge, die die Version der verwendeten Struktur angeben. Bei der erfolgreichen Rückkehr enthalten die Felder in der Struktur die entsprechenden Informationen. Bei **adsmServerName** handelt es sich um den Namen, der dem IBM Spectrum Protect-Server im Befehl **define**

server zugeordnet wird. Ist das Feld 'archiveRetentionProtection' auf wahr gesetzt, ist der Server für den Aufbewahrungsschutz aktiviert.

Rückkehrcodes

Die Rückkehrcodenummern sind in runden Klammern ( ) angegeben.

Tabelle 42. Rückkehrcodes für dsmQuerySessInfo

Rückkehrcode	Erläuterung
DSM_RC_NO_SESS_BLK (2006)	Keine Serversitzungsblockinformationen.
DSM_RC_NO_POLICY_BLK (2007)	Keine Servermaßnahmeninformationen verfügbar.
DSM_RC_WRONG_VERSION_PARM (2065)	Die API-Version des Anwendungsclients stimmt nicht mit der Version der IBM Spectrum Protect-Bibliothek überein.

dsmQuerySessOptions

Mit dem Funktionsaufruf **dsmQuerySessOptions** werden wichtige Optionswerte, die in der in **dsmHandle** angegebenen Sitzung gültig sind, abgefragt. Eine Struktur des Typs `optStruct` wird in dem Aufruf übergeben und enthält die Informationen.

Dieser Aufruf wird nach einem erfolgreichen Aufruf **dsmInitEx** gestartet. Die Werte, die zurückgegeben werden, können sich, abhängig von Werten, die an den Aufruf **dsmInitEx** übergeben werden, von den Werten, die in einem Aufruf **dsmQueryCliOptions** zurückgegeben werden, unterscheiden; dies betrifft in erste Linie `optString` und `optFile`. Informationen zur Vorrangstellung der einzelnen Optionen finden Sie in „Erläuterungen zu Konfigurations- und Optionsdateien” auf Seite 1.

Es gibt keine spezifischen Rückkehrcodes für diesen Aufruf.

Syntax

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

Parameter

dsUint32_t dsmhandle(I)
Die Kennung, die diesen Aufruf einem vorherigen Aufruf **dsmInitEx** zuordnet.

optStruct *optstructP (I/O)
Dieser Parameter übergibt die Adresse der Struktur, die die API ausführt. Der Anwendungsclient ist für das Zuordnen von Speicher für die Struktur zuständig. Bei der erfolgreichen Rückkehr enthalten die Felder in der Struktur die entsprechenden Informationen.

In der Struktur des Typs 'optStruct' werden die folgenden Informationen zurückgegeben:

Name	Beschreibung
dsmiDir	Der Wert der Umgebungsvariablen `DSMI_DIR`.
dsmiConfig	Die Datei dsm.opt, die in der Umgebungsvariablen `DSMI_CONFIG` angegeben ist.
serverName	Der Name in der IBM Spectrum Protect-Serverzeilengruppe in der Optionsdatei.

Name	Beschreibung
commMethod	Die ausgewählte Übertragungsmethode. Siehe #defines für DSM_COMM_* in der Datei dsmapitd.h.
serverAddress	Die Adresse des Servers auf der Basis der Übertragungsmethode.
nodeName	Der Name des Knotens (der Maschine) des Clients.
compression	Der Wert für die Option 'compression' (`bTrue=on` und `bFalse=off`).
compressAlways	Der Wert für die Option `compressalways` (`bTrue=on` und `bFalse=off`).
passwordAccess	Wert `bTrue` für 'generate' und `bFalse` für 'prompt'.

Zugehörige Konzepte:

Verarbeitungsoptionen

dsmRCMsg

Mit dem Funktionsaufruf **dsmRCMsg** wird der Nachrichtentext abgerufen, der einem API-Rückkehrcode zugeordnet ist.

Der Parameter **msg** zeigt den Nachrichtenpräfixrückkehrcode in runden Klammern ( ) gefolgt vom Nachrichtentext. Ein Aufruf **dsmRCMsg** könnte beispielsweise Folgendes zurückgeben:

```
ANS0264E (RC2300) Nur der Root
darf dsmChangePW oder
dsmDeleteFS ausführen.
```

Für einige Sprachen, bei denen Zeichen in ANSI- und OEM-Codepages unterschiedlich sind, kann es notwendig sein, Zeichenfolgen von ANSI in OEM zu konvertieren (z. B. Einzelbytezeichensätze für Osteuropa). Nachfolgend ein Beispiel:

```
dsmRCMsg(dsmHandle, rc, msgBuf);
#ifdef WIN32
#ifndef WIN64
CharToOemBuff(msgBuf, msgBuf, strlen(msgBuf));
#endif
#endif
printf("
```

Syntax

```
dsInt16_t dsmRCMsg  (dsUint32_t       dsmHandle,
   dsInt16_t        dsmRC,
   char        *msg);
```

Parameter

dsUint32_t dsmHandle (I)
Die Kennung, die diesen Aufruf einem vorherigen Aufruf **dsmInitEx** zuordnet.

dsInt16_t dsmRC (I)
Der API-Rückkehrcode des zugeordneten Nachrichtentextes. Die API-Rückkehrcodes sind in der Datei dsmrc.h aufgelistet. Weitere Informationen finden Sie in Anhang A, „Quellendatei mit API-Rückkehrcodes: `dsmrc.h`", auf Seite 159.

char *msg (O)
Dieser Parameter gibt den Nachrichtentext an, der dem Rückkehrcode **dsmRC** zugeordnet ist. Der Aufrufende ist für das Zuordnen von ausreichend Speicherbereich verantwortlich.

Die maximale Länge für **msg** ist als `DSM_MAX_RC_MSG_LENGTH` definiert.

Auf Plattformen mit Unterstützung in der Landessprache und mehreren länderspezifischen Nachrichtendateien gibt die API eine Nachrichtenzeichenfolge in der Landessprache zurück.

Rückkehrcodes

Die Rückkehrcodenummern sind in runden Klammern ( ) angegeben.

Tabelle 43. Rückkehrcodes für dsmRCMsg

Rückkehrcode	Erläuterung
DSM_RC_NULL_MSG (2002)	Der Parameter **msg** für den Aufruf dsmRCMsg ist ein NULL-Zeiger.
DSM_RC_INVALID_RETCODE (2021)	Der Rückkehrcode, der an den Aufruf **dsmRCMsg** übergeben wurde, ist ungültig.
DSM_RC_NLS_CANT_OPEN_TXT (0610)	Die Nachrichtentextdatei kann nicht geöffnet werden.

dsmRegisterFS

Mit dem Funktionsaufruf **dsmRegisterFS** wird ein neuer Dateibereich beim IBM Spectrum Protect-Server registriert. Bevor Sie Daten in einem Dateibereich sichern können, müssen Sie diesen zuvor registrieren.

Anwendungsclients dürfen nicht dieselben Dateibereichsnamen wie ein Client für Sichern/Archivieren verwenden.

- Führen Sie unter UNIX oder Linux den Befehl **df** für diese Namen aus.
- Bei Windows handelt es sich bei diesen Namen im Allgemeinen um die Datenträgerkennsätze, die den einzelnen Laufwerken auf Ihrem System zugeordnet sind.

Syntax

```
dsInt16_t dsmRegisterFS  (dsUint32_t         dsmHandle,
   regFSData     *regFilespaceP);
```

Parameter

dsUint32_t dsmHandle (I)
Die Kennung, die diesen Aufruf einem vorherigen Aufruf **dsmInitEx** zuordnet.

regFSData *regFilespaceP (I)
Mit diesem Parameter werden der Name des Dateibereichs und die zugehörigen Informationen, die für die Registrierung beim IBM Spectrum Protect-Server erforderlich sind, übergeben.

Tipp: Das Feld *fstype* umfasst das Präfix, **„API:"**. Diese Zeichenfolge wird für alle Dateibereichsabfragen angezeigt. Übergibt beispielsweise der Benutzer *myfstype* für *fstype* im Aufruf **dsmRegisterFS**, wird als tatsächlicher Wert für die Zeichenfolge auf dem Server bei der Abfrage `API:myfstype` zurückgegeben. Dieses Präfix dient zur Unterscheidung zwischen API-Objekten und Objekten des Clients für Sichern/Archivieren.

Der verwendbare Bereich für **fsInfo** ist jetzt DSM_MAX_USER_FSINFO-_LENGTH.

Rückkehrcodes

Die Rückkehrcodenummern sind in runden Klammern ( ) angegeben.

Tabelle 44. Rückkehrcodes für dsmRegisterFS

Rückkehrcode	Erläuterung
DSM_RC_INVALID_FSNAME (2016)	Ungültiger Dateibereichsname.
DSM_RC_INVALID_DRIVE_CHAR (2026)	Laufwerkbuchstabe ist kein alphabetischen Zeichen.
DSM_RC_NULL_FSNAME (2027)	Leerer Dateibereichsname.
DSM_RC_FS_ALREADY_REGED (2062)	Dateibereich ist bereits registriert.
DSM_RC_WRONG_VERSION_PARM (2065)	Die API-Version des Anwendungsclients stimmt nicht mit der Version der IBM Spectrum Protect-Bibliothek überein.
DSM_RC_FSINFO_TOOLONG (2106)	Dateibereichsinformationen sind zu lang.

dsmReleaseBuffer

Mit der Funktion **dsmReleaseBuffer** wird ein Puffer an IBM Spectrum Protect zurückgegeben. **dsmReleaseBuffer** wird von der Anwendung aufgerufen, nachdem ein Aufruf **dsmGetDataEx** erfolgt ist und die Anwendung alle Daten aus dem Puffer übertragen hat und zum Freigeben des Puffers bereit ist. **dsmReleaseBuffer** erfordert den Aufruf **dsmInitEx** unter Angabe von ***btrue*** für *UseTsmBuffers* und einem Wert ungleich null für *numTsmBuffers*. **dsmReleaseBuffer** sollte auch aufgerufen werden, wenn der Aufruf **dsmTerminate** unmittelbar bevorsteht und die Anwendung Datenpuffer noch nicht freigegeben hat.

Syntax

```
dsInt16_t  dsmReleaseBuffer  (releaseBufferIn_t     *dsmReleaseBufferInP,
                              releaseBufferOut_t    *dsmReleaseBufferOutP) ;
```

Parameter

releaseBufferIn_t *dsmReleaseBufferInP (I)
Diese Struktur enthält die folgenden Eingabeparameter:

dsUint32_t dsmHandle (I)
Die Kennung, die diesen Aufruf einem vorherigen Aufruf **dsmInitEx** zuordnet.

dsUint8_t tsmBufferHandle(I)
Die Kennung, die diesen Puffer identifiziert.

char *dataPtr(I)
Die Adresse, an die die Anwendung geschrieben wird.

Rückkehrcodes

Die Rückkehrcodenummern sind in runden Klammern ( ) angegeben.

Tabelle 45. Rückkehrcodes für dsmReleaseBuffer

Rückkehrcode	Erläuterung
DSM_RC_BAD_CALL_SEQUENCE	Der Aufruf wurde nicht im korrekten Status ausgegeben.
DSM_RC_INVALID_TSMBUFFER	Die Kennung oder der Wert von **dataPtr** ist ungültig.
DSM_RC_BUFF_ARRAY_ERROR	Ein Pufferarrayfehler ist aufgetreten.

dsmRenameObj

Mit dem Funktionsaufruf **dsmRenameObj** wird der übergeordnete oder untergeordnete Objektname umbenannt. Übergeben Sie bei Sicherungsobjekten den aktuellen Objektnamen und Änderungen für über- oder untergeordnete Objektnamen. Übergeben Sie bei Archivierungsobjekten den aktuellen Objektdateibereichsnamen und die Objekt-ID sowie Änderungen für über- oder untergeordnete Objektnamen. Verwenden Sie diesen Funktionsaufruf in Aufrufen **dsmBeginTxn** und **dsmEndTxn**.

Das Mischflag legt fest, ob ein doppelter Sicherungsobjektname mit den vorhandenen Sicherungen gemischt wird. Wenn der neue Name einem vorhandenen Objekt entspricht und der Mischvorgang wahr ist, wird das aktuelle Objekt in den neuen Namen konvertiert und wird zu der aktiven Version des neuen Namens, während das vorhandene aktive Objekt mit diesem Namen zur ersten inaktiven Kopie des Objekts wird. Wenn der neue Name einem vorhandenen Objekt entspricht und der Mischvorgang falsch ist, gibt die Funktion den Rückkehrcode DSM_RC_ABORT-_DUPLICATE_OBJECT zurück.

Einschränkung: Ein Objekt kann nur von seinem Eigner umbenannt werden.

Im Funktionsaufruf **dsmRenameObj** wird auf diese Mischbedingungen getestet:

- Das aktuelle Objekt in der Struktur **dsmObjName** und das neue übergeordnete oder untergeordnete Objekt müssen in Bezug auf Eigner, Kopiengruppe und Verwaltungsklasse übereinstimmen.
- Das aktuelle Objekt in der Struktur **dsmObjName** muss nach dem momentan aktiven Objekt mit dem neuen Namen gesichert worden sein.
- Es darf nur eine einzige aktive Kopie des aktuellen Objekts in der Struktur **dsmObjName** und es dürfen keine inaktiven Kopien vorhanden sein.

Syntax

```
dsInt16_t dsmRenameObj  (dsmRenameIn_t     *dsmRenameInP,
                         dsmRenameOut_t    *dsmRenameOutP);
```

Parameter

dsUint32_t dsmHandle (I)
Die Kennung, die diesen Aufruf einem vorherigen Aufruf **dsmInitEx** zuordnet.

dsmRenameIn_t *dsmRenameInP
Diese Struktur enthält die Eingabeparameter.

dsUint8_t repository (I);
Dieser Parameter gibt an, ob der zu löschende Dateibereich im Sicherungsrepository oder im Archivrepository vorhanden ist.

dsmObjName *objNameP (I);
Dieser Parameter ist ein Verweis auf die Struktur, die die aktuellen Werte für den Dateibereichsnamen, den übergeordneten Objektnamen, den untergeordneten Objektnamen und den Objekttyp enthält.

char newHl [DSM_MAX_HL_LENGTH + 1];
Dieser Parameter gibt den neuen übergeordneten Namen an.

char newLl [DSM_MAX_LL_LENGTH + 1];
Dieser Parameter gibt den neuen untergeordneten Namen an.

dsBool_t merge;
Dieser Parameter legt fest, ob ein Sicherungsobjekt mit doppelten benannten Objekten gemischt wird. Gültige Werte sind 'true' und 'false'.

ObjID;
Die Objekt-ID für Archivierungsobjekte.

dsmRenameOut_t *dsmRnameOutP
Diese Struktur enthält die Ausgabeparameter.

Anmerkung: Es gibt keine Ausgabeparameter.

Rückkehrcodes

Die Rückkehrcodenummern sind in runden Klammern ( ) angegeben.

Tabelle 46. Rückkehrcodes für dsmRenameObj

Rückkehrcode	Erläuterung
DSM_RC_ABORT_MERGE_ERROR (45)	Der Server hat einen Mischfehler erkannt.
DSM_RC_ABORT_DUPLICATE_OBJECT (32)	Das Objekt ist bereits vorhanden und der Mischvorgang ist falsch.
DSM_RC_ABORT_NO_MATCH (2)	Objekt nicht gefunden.
DSM_RC_REJECT_SERVER_DOWNLEVEL (58)	Diese Funktion kann nur auf einem IBM Spectrum Protect-Server der Version 3.7.4.0 oder höher verwendet werden.

dsmRequestBuffer

Mit der Funktion **dsmRequestBuffer** wird ein Puffer an IBM Spectrum Protect zurückgegeben. **dsmRequestBuffer** wird von der Anwendung aufgerufen, nachdem ein Aufruf **dsmGetDataEx** erfolgt ist und die Anwendung alle Daten aus dem Puffer übertragen hat und zum Freigeben des Puffers bereit ist.

dsmReleaseBuffer erfordert den Aufruf **dsmInitEx** unter Angabe von *btrue* für *UseTsmBuffers* und einem Wert ungleich null für *numTsmBuffers*. **dsmReleaseBuffer** sollte auch aufgerufen werden, wenn der Aufruf **dsmTerminate** unmittelbar bevorsteht und die Anwendung IBM Spectrum Protect-Puffer noch nicht freigegeben hat.

Syntax

```
dsInt16_t  dsmRequestBuffer   (getBufferIn_t      *dsmRequestBufferInP,
                               getBufferOut_t     *dsmRequestBufferOutP) ;
```

Parameter

getBufferIn_t * dsmRequestBufferInP (I)
Diese Struktur enthält den folgenden Eingabeparameter:

dsUint32_t dsmHandle
Die Kennung, die die Sitzung identifiziert und einem vorherigen Aufruf **dsmInitEx** zuordnet.

getBufferOut_t *dsmRequestBufferOut P (0)
Diese Struktur enthält die Ausgabeparameter.

dsUint8_t tsmBufferHandle(0)
Die Kennung, die diesen Puffer identifiziert.

char *dataPtr(0)
Die Adresse, an die die Anwendung geschrieben wird.

dsUint32_t *bufferLen(0)
Die maximale Anzahl Byte, die in diesen Puffer geschrieben werden kann.

Rückkehrcodes

Die Rückkehrcodenummern sind in runden Klammern ( ) angegeben.

Tabelle 47. Rückkehrcodes für dsmRequestBuffer

Rückkehrcode	Erläuterung
DSM_RC_BAD_CALL_SEQUENCE (33)	Der Aufruf wurde nicht im korrekten Status ausgegeben.
DSM_RC_SENDDATA_WITH_ZERO_SIZE (34)	Wenn das gesendete Objekt die Länge 0 hat, sind keine Aufrufe **dsmReleaseBuffer** zulässig.
DSM_RC_BUFF_ARRAY_ERROR (121)	Es konnte kein gültiger Puffer abgerufen werden.

dsmRetentionEvent

Mit dem Funktionsaufruf **dsmRetentionEvent** wird im Rahmen einer Aufbewahrungsereignisoperation, die für die in der Liste angegebenen Objekte ausgeführt werden soll, eine Liste mit Objekt-IDs an den IBM Spectrum Protect-Server gesendet. Verwenden Sie diesen Funktionsaufruf in Aufrufen **dsmBeginTxn** und **dsmEndTxn**.

Anmerkung: Diese Funktion kann nur auf einem Server der Version 5.2.2.0 oder höher verwendet werden.

Die maximale Anzahl Objekte in einem Aufruf ist auf den Wert im Feld *maxObjPerTxn* begrenzt, das in der Struktur des Typs *ApisessInfo* von einem Aufruf **dsmQuerySessInfo** zurückgegeben wird.

Ein Ereignis für ein Objekt kann nur vom Eigner dieses Objekts gesendet werden.

Gültige Ereignisse sind:

eventRetentionActivate
Kann nur für Objekte ausgegeben werden, die an eine ereignisgesteuerte Verwaltungsklasse gebunden sind. Durch das Senden dieses Ereignisses wird das Ereignis für dieses Objekt aktiviert und der Status für die Aufbewahrungsdauer für dieses Objekt ändert sich von DSM_ARCH_RETINIT_PENDING in DSM_ARCH_RETINIT_STARTED.

eventHoldObj
Dieses Ereignis gibt eine temporäre Sperre in Bezug auf die Aufbewahrungsdauer oder das Löschen für ein Objekt aus, sodass das Objekt erst verfallen oder gelöscht werden kann, nachdem eine Freigabe erfolgt ist.

eventReleaseObj
Dieses Ereignis kann nur für ein Objekt ausgegeben werden, für das das Feld ***objectHeld*** den Wert DSM_ARCH_HELD_TRUE enthält; mit diesem Ereignis wird die temporäre Sperre für das Objekt entfernt und die ursprüngliche Maßnahme für Aufbewahrungsdauer wieder aufgenommen.

Bevor Sie den Aufruf ***dsmRetentionEvent*** senden, senden Sie die in „IBM Spectrum Protect-System abfragen" auf Seite 33 beschriebene Abfragefolge, um die Informationen für das Objekt abzurufen. Der Aufruf **dsmGetNextQObj** gibt für Archivabfragen eine Datenstruktur mit dem Namen **qryRespArchiveData** zurück. Diese Datenstruktur enthält die Informationen, die für **dsmRetentionEvent** benötigt werden.

Syntax

```
extern dsInt16_t DSMLINKAGE dsmRetentionEvent(
  dsmRetentionEventIn_t          *ddsmRetentionEventInP,
  dsmRetentionEventOut_t         *dsmRetentionEventOutP
   );
```

Parameter

dsmRetentionEventIn_t *dsmRetentionEventP
Diese Struktur enthält die folgenden Eingabeparameter:

dsUint16_t stVersion;
Dieser Parameter gibt die Strukturversion an.

dsUint32_t dsmHandle (I)
Die Kennung, die diesen Aufruf einem vorherigen Aufruf **dsmInitEx** zuordnet.

dsmEventType_t evenType (I);
Dieser Parameter gibt den Ereignistyp an. Die Bedeutung der gültigen Werte (**eventRetentionActivate**, **eventHoldObj** und **eventReleaseObj**) finden Sie am Anfang dieses Abschnitts.

dsmObjList_t objList;
Dieser Parameter gibt eine Liste der als Signal zu sendenden Objekt-IDs an.

Rückkehrcodes

Die Rückkehrcodenummern sind in runden Klammern ( ) angegeben.

Tabelle 48. Rückkehrcodes für dsmRetentionEvent

Rückkehrcode	Erläuterung
DSM_RC_ABORT_NODE_NOT_AUTHORIZED (36)	Der Knoten oder Benutzer hat nicht die korrekte Berechtigung.
DSM_RC_ABORT_TXN_LIMIT_EXCEEDED (249)	Zu viele Objekte in der Transaktion.
DSM_RC_ABORT_OBJECT_ALREADY_HELD (250)	Das Objekt ist bereits gesperrt, daher kann keine weitere temporäre Sperre ausgegeben werden.
DSM_RC_REJECT_SERVER_DOWNLEVEL (58)	Diese Funktion kann nur auf einem Server der Version 5.2.2.0 oder höher verwendet werden.

dsmSendBufferData

Mit dem Funktionsaufruf **dsmSendBufferData** wird ein Bytestrom mit Daten über einen in einem vorherigen Aufruf **dsmReleaseBuffer** bereitgestellten Puffer an IBM Spectrum Protect gesendet. Der Anwendungsclient kann jeden Typ von Daten zum Speichern auf dem Server übergeben. In der Regel (aber nicht ausschließlich) handelt es sich bei diesen Daten um Dateidaten. Sie können **dsmSendBufferData** mehrmals aufrufen, wenn der Bytestrom mit Daten, den Sie senden, groß ist. Der Puffer wird unabhängig davon, ob der Aufruf erfolgreich ausgeführt wird oder fehlschlägt, freigegeben.

Einschränkung: Bei Verwendung der Option *useTsmBuffers* wird das Objekt nicht komprimiert, selbst wenn es für die Komprimierung eingeschlossen ist.

Syntax

```
dsInt16_t  dsmSendBufferData   (sendBufferDataIn_t        *dsmSendBufferDataExInP,
                                sendBufferDataOut_t       *dsmSendBufferDataOutP) ;
```

Parameter

sendBufferDataIn_t * dsmSendBufferDataInP (I)
Diese Struktur enthält die folgenden Eingabeparameter:

dsUint32_t dsmHandle (I)
Die Kennung, die diesen Aufruf einem vorherigen Aufruf **dsmInitEx** zuordnet.

dsUint8_t tsmBufferHandle(I)
Die Kennung, die den zu sendenden Puffer identifiziert.

char *dataPtr(I)
Die Adresse, an die Anwendungsdaten geschrieben wurden.

dsUint32_t numBytes(I)
Die tatsächliche Anzahl Byte, die von der Anwendung geschrieben wurde (dieser Wert sollte immer kleiner als der in **dsmReleaseBuffer** angegebene Wert sein).

Rückkehrcodes

Die Rückkehrcodenummern sind in runden Klammern ( ) angegeben.

Tabelle 49. Rückkehrcodes für dsmSendBufferData

Rückkehrcode	Erläuterung
DSM_RC_BAD_CALL_SEQUENCE (2041)	Der Aufruf wurde nicht im korrekten Status ausgegeben.
DSM_RC_INVALID_TSMBUFFER (2042)	Die Kennung oder der Wert von **dataPtr** ist ungültig.
DSM_RC_BUFF_ARRAY_ERROR (2045)	Ein Pufferarrayfehler ist aufgetreten.
DSM_RC_TOO_MANY_BYTES (2043)	Der Wert von ***numBytes*** ist größer als der im Aufruf **dsmReleaseBuffer** bereitgestellte Puffer.

dsmSendData

Mit dem Funktionsaufruf **dsmSendData** wird ein Bytestrom mit Daten über einen Puffer an IBM Spectrum Protect gesendet. Der Anwendungsclient kann jeden Typ von Daten zum Speichern auf dem Server übergeben. In der Regel (aber nicht ausschließlich) handelt es sich bei diesen Daten um Dateidaten. Sie können **dsmSendData** mehrmals aufrufen, wenn der Bytestrom mit Daten, der gesendet werden soll, groß ist.

Einschränkung: Der Anwendungsclient kann den Puffer, der in **dsmSendData** angegeben ist, erst nach der Rückkehr des Aufrufs **dsmSendData** wiederverwenden.

Tipp: Wenn IBM Spectrum Protect den Code 157 (DSM_RC_WILL_ABORT) zurückgibt, starten Sie einen Aufruf **dsmEndSendObj** und dann einen Aufruf **dsmEndTxn** mit dem Votum DSM_VOTE_COMMIT. Die Anwendung empfängt dann den Rückkehrcode 2302 (DSM_RC_CHECK_REASON_CODE) und übergibt den Ursachencode zurück an den Anwendungsbenutzer. Damit wird dem Benutzer mitgeteilt, warum der Server die Transaktion beendet.

Syntax

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

Parameter

dsUint32_t dsmHandle (I)
Die Kennung, die diesen Aufruf einem vorherigen Aufruf **dsmInitEx** zuordnet.

DataBlk *dataBlkPtr (I/O)
Dieser Parameter verweist auf eine Struktur, die sowohl einen Zeiger auf den Puffer enthält, aus dem die Daten gesendet werden sollen, als auch einen Zeiger auf die Größe des Puffers. Bei der Rückkehr enthält diese Struktur die Anzahl tatsächlich übertragener Byte. Die Typdefinition finden Sie in Anhang B, „Quellendateien für API-Typdefinitionen", auf Seite 171.

Rückkehrcodes

Die Rückkehrcodenummern sind in runden Klammern ( ) angegeben.

Tabelle 50. Rückkehrcodes für dsmSendData

Rückkehrcode	Erläuterung
DSM_RC_NO_COMPRESS_MEMORY (154)	Es ist nicht genügend Speicher zum Ausführen der Datenkomprimierung oder -erweiterung verfügbar.
DSM_RC_COMPRESS_GREW (155)	Während der Komprimierung nahm die Größe der komprimierten Daten im Vergleich zu den ursprünglichen Daten zu.
DSM_RC_WILL_ABORT (157)	Ein unbekannter und unerwarteter Fehler ist aufgetreten, der das Anhalten der Transaktion zur Folge hatte.
DSM_RC_WRONG_VERSION_PARM (2065)	Die API-Version des Anwendungsclients stimmt nicht mit der Version der IBM Spectrum Protect-Bibliothek überein.
DSM_RC_NEEDTO_ENDTXN (2070)	Die Transaktion muss beendet werden.
DSM_RC_OBJ_EXCLUDED (2080)	Das Objekt wird von der Einschluss-/Ausschlussliste ausgeschlossen.
DSM_RC_OBJ_NOBCG (2081)	Das Objekt hat keine Sicherungskopiengruppe und wird nicht an den Server gesendet.
DSM_RC_OBJ_NOACG (2082)	Das Objekt hat keine Archivierungskopiengruppe und wird nicht an den Server gesendet.
DSM_RC_SENDDATA_WITH_ZERO_SIZE (2107)	Ein Objekt kann keine Daten mit einer Größenschätzung (*sizeEstimate*) von null Byte senden.

dsmSendObj

Mit dem Funktionsaufruf **dsmSendObj** wird eine Anforderung zum Senden eines einzelnen Objekts in Speicher gestartet. Aufgrund von Leistungsaspekten können mehrere Aufrufe **dsmSendObj** und zugehörige Aufrufe **dsmSendData** innerhalb der Grenzen einer Transaktion ausgeführt werden.

Mit dem Aufruf **dsmSendObj** werden die Daten für das Objekt als Bytestrom verarbeitet, der in Hauptspeicherpuffer übergeben wurde. Der Parameter **dataBlkPtr** im Aufruf **dsmSendObj** ermöglicht dem Anwendungsclient eine der folgenden Aktionen:

- Übergabe der Daten und der Attribute (die Attribute werden über **objAttrPtr** übergeben) des Objekts in einem einzigen Aufruf.

- Angabe eines Teils der Objektdaten über den Aufruf **dsmSendObj** und Angabe der verbleibenden Daten über einen oder mehrere Aufrufe **dsmSendData**.

Alternativ kann der Anwendungsclient auch nur die Attribute über den Aufruf **dsmSendObj** angeben und die Objektdaten über einen oder mehrere Aufrufe **dsmSendData** angeben. Bei diesem Verfahren müssen Sie **dataBlkPtr** im Aufruf **dsmSendObj** auf NULL setzen.

Tipp: Bei bestimmten Objekttypen werden Bytestromdaten möglicherweise nicht den Daten zugeordnet, beispielsweise ein Verzeichniseintrag ohne erweiterte Attribute.

Bevor **dsmSendObj** aufgerufen wird, muss ein vorhergehender Aufruf **dsmBindMC** ausgeführt werden, um eine Verwaltungsklasse korrekt an das Objekt zu binden, das gesichert oder archiviert werden soll. Die API behält diese Bindung bei, sodass sie dem Objekt, wenn es an den Server gesendet wird, die korrekte Verwaltungsklasse zuordnen kann. Wenn die Verwaltungsklasse, die in einem Aufruf **dsmSendObj** gebunden wird, standardmäßig den Objekttyp 'Verzeichnis' (DSM_OBJ_DIRECTORY) annehmen kann, entspricht der Standardwert möglicherweise nicht der Standardverwaltungsklasse. Stattdessen wird die Verwaltungsklasse mit dem längsten Aufbewahrungszeitraum verwendet. Sind mehrere Verwaltungsklassen mit diesem Aufbewahrungszeitraum vorhanden, wird die erste gefundene Verwaltungsklasse verwendet.

Führen Sie für alle Objektdaten, die in Speicher gesendet werden, anschließend einen Aufruf **dsmEndSendObj** aus. Sind keine Objektdaten zum Senden an den Server vorhanden oder waren alle Daten in dem Aufruf **dsmSendObj** enthalten, müssen Sie einen Aufruf **dsmEndSendObj** starten, bevor Sie einen weiteren Aufruf **dsmSendObj** starten können. Waren mehrere Operationen für das Senden von Daten über den Aufruf **dsmSendData** erforderlich, folgt der Aufruf **dsmEndSendObj** auf die letzte Sendeoperation, um die Statusänderung anzugeben.

Tipp: Wenn IBM Spectrum Protect den Code 157 (DSM_RC_WILL_ABORT) zurückgibt, starten Sie einen Aufruf **dsmEndTxn** mit dem Votum DSM_VOTE_COMMIT. Die Anwendung empfängt den Rückkehrcode 2302 (DSM_RC_CHECK_REASON_CODE) und übergibt den Ursachencode zurück an den Anwendungsbenutzer. Damit wird dem Benutzer mitgeteilt, warum der Server die Transaktion beendet.

Lautet der Ursachencode 11 (DSM_RS_ABORT_NO_REPOSIT_SPACE), ist es möglich, dass die Größenschätzung (*sizeEstimate*) für das tatsächliche Datenvolumen zu klein ist. Die Anwendung muss eine genauere Größenschätzung (*sizeEstimate*) durchführen und die Daten erneut senden.

Syntax

```
dsInt16_t dsmSendObj  (dsUint32_t       dsmHandle,
   dsmSendType            sendType,
   void        *sendBuff,
   dsmObjName             *objNameP,
   ObjAttr     *objAttrPtr,
   DataBlk     *dataBlkPtr);
```

Parameter

dsUint32_t dsmHandle (I)
Die Kennung, die diesen Aufruf einem vorherigen Aufruf **dsmInitEx** zuordnet.

dsmSendType sendType (I)
Dieser Parameter gibt den Sendetyp an, der ausgeführt wird. Gültige Werte umfassen:

Name	Beschreibung
stBackup	Ein Sicherungsobjekt, das an den Server gesendet wird.
stArchive	Ein Archivierungsobjekt, das an den Server gesendet wird.
stBackupMountWait	Ein Sicherungsobjekt, für das der Server warten soll, bis die erforderliche Einheit bereitgestellt wird , z. B. ein Band geladen wird.
stArchiveMountWait	Ein Archivierungsobjekt, für das der Server warten soll, bis die erforderliche Einheit bereitgestellt wird, z. B. ein Band geladen wird.

Anmerkung: Verwenden Sie die **MountWait**-Typen, wenn die Möglichkeit besteht, dass Ihr Anwendungsbenutzer unter Umständen Daten an ein Band sendet.

void *sendBuff (I)
Dieser Parameter ist ein Verweis auf eine Struktur, die andere Informationen enthält, die für den Sendetyp (**sendType**) in dem Aufruf spezifisch sind. Derzeit ist nur für einen Sendetyp (**sendType**) mit dem Wert **stArchive** eine Struktur zugeordnet. Diese Struktur hat den Namen **sndArchiveData** und enthält die Archivierungsbeschreibung.

dsmObjName *objNameP (I)
Dieser Parameter ist ein Verweis auf die Struktur, die den Dateibereichsnamen, den übergeordneten Objektnamen, den untergeordneten Objektnamen und den Objekttyp enthält. Weitere Informationen finden Sie in „Objektnamen und -IDs" auf Seite 22.

ObjAttr *objAttrPtr (I)
Dieser Parameter übergibt Objektattribute, die für die Anwendung von Interesse sind. Die Typdefinition finden Sie in Anhang B, „Quellendateien für API-Typdefinitionen", auf Seite 171.

Die Attribute sind:

- **owner** bezieht sich auf den Eigner des Objekts. Wenn das Objekt wieder aus dem IBM Spectrum Protect-Speicher abgerufen wird, ist es wichtig festzustellen, ob der Eigner als spezifischer Name oder als leere Zeichenfolge deklariert ist. Weitere Informationen finden Sie in „Zugriff auf Objekte als Sitzungseigner" auf Seite 24.
- **sizeEstimate** ist die bestmögliche Schätzung der Gesamtgröße des Datenobjekts, das an den Server gesendet werden soll. Diese Größe sollte so genau wie möglich geschätzt werden, da der Server dieses Attribut für die effiziente Bereichszuordnung und Objektpositionierung in seinen Speicherressourcen verwendet.

  Ist die von Ihnen angegebene Größenschätzung erheblich geringer als die tatsächliche Anzahl Byte, die gesendet wird, hat der Server möglicherweise Schwierigkeiten, ausreichend Speicherbereich zuzuordnen, und beendet die Transaktion mit dem Ursachencode 11 (DSM_RS_ABORT_NO_REPOSIT_SPACE).

  Anmerkung: Die Größenschätzung gilt für die Gesamtgröße des Datenobjekts und wird in Byte angegeben.

  Objekte mit einer geringeren Größe als DSM_MIN_COMPRESS_SIZE werden nicht komprimiert.

Wenn Ihr Objekt keine Bitdaten (sondern nur Attributinformationen dieses Aufrufs) enthält, sollte der Wert für **sizeEstimate** null sein.

Anmerkung: Ab Version 5.1.0 wird das Kopienziel innerhalb einer Transaktion für Objekte mit der Länge null nicht auf Konsistenz geprüft.

- **objCompressed** ist ein boolescher Wert, der angibt, ob die Objektdaten bereits komprimiert wurden oder nicht.

  Wenn das Objekt komprimiert ist (*compressed=bTrue*), versucht IBM Spectrum Protect nicht, das Objekt erneut zu komprimieren. Wenn das Objekt nicht komprimiert ist, entscheidet IBM Spectrum Protect, ob das Objekt komprimiert werden soll. Diese Entscheidung ist von den Werten für die Option `compression` abhängig, die vom Administrator und in den API-Konfigurationsquellen festgelegt wurden.

  Soll Ihre Anwendung eine Zurückschreibung oder einen Abruf von Teilobjekten ausführen, können die Daten während des Sendevorgangs nicht komprimiert werden. Um dies durchzusetzen, setzen Sie *ObjAttr.objCompressed* auf *bTrue*.
- Mit **objInfo** werden Informationen zu dem spezifischen Objekt gespeichert.

  Einschränkung: Die Informationen werden nicht automatisch gespeichert. Wenn dieses Attribut verwendet wird, müssen Sie das Attribut *objInfoLength* definieren, um die Länge für *objInfo* anzugeben.
- **mcNameP** enthält den Namen einer Verwaltungsklasse, die die mit **dsmBindMC** abgerufene Verwaltungsklasse überschreibt.
- **disableDeduplication** ist ein boolescher Wert. Wird er auf wahr gesetzt, wird das Objekt nicht vom Client dedupliziert.

DataBlk *dataBlkPtr (I/O)
Dieser Parameter verweist auf eine Struktur, die sowohl einen Zeiger auf den Puffer für die Daten enthält, die gesichert oder archiviert werden sollen, als auch einen Zeiger auf die Größe des Puffers. Dieser Parameter gilt nur für **dsmSendObj**. Wenn Sie den Sendevorgang für Daten in einem nachfolgenden Aufruf **dsmSendData** statt im Aufruf **dsmSendObj** starten möchten, setzen Sie den Pufferzeiger in der DataBlk-Struktur auf `NULL`. Bei der Rückkehr enthält diese Struktur die Anzahl tatsächlich übertragener Byte. Die Typdefinition finden Sie in Anhang B, „Quellendateien für API-Typdefinitionen", auf Seite 171.

Rückkehrcodes

Die Rückkehrcodenummern sind in runden Klammern ( ) angegeben.

Tabelle 51. Rückkehrcodes für dsmSendObj

Rückkehrcode	Erläuterung
DSM_RC_NO_COMPRESS_MEMORY (154)	Es ist nicht genügend Speicher zum Ausführen der Datenkomprimierung oder -erweiterung verfügbar.
DSM_RC_COMPRESS_GREW (155)	Während der Komprimierung nahm die Größe der komprimierten Daten im Vergleich zu den ursprünglichen Daten zu.
DSM_RC_WILL_ABORT (157)	Ein unbekannter und unerwarteter Fehler ist aufgetreten, der das Anhalten der Transaktion zur Folge hatte.
DSM_RC_TL_NOACG (186)	Die Verwaltungsklasse für diese Datei hat keine gültige Kopiengruppe für den Sendetyp.
DSM_RC_NULL_OBJNAME (2000)	Objektname ist null.
DSM_RC_NULL_OBJATTRPTR (2004)	Objektattributzeiger ist null.

Tabelle 51. Rückkehrcodes für dsmSendObj (Forts.)

Rückkehrcode	Erläuterung
DSM_RC_INVALID_OBJTYPE (2010)	Ungültiger Objekttyp.
DSM_RC_INVALID_OBJOWNER (2019)	Ungültiger Objekteigner.
DSM_RC_INVALID_SENDTYPE (2022)	Ungültiger Sendetyp.
DSM_RC_WILDCHAR_NOTALLOWED (2050)	Platzhalterzeichen nicht zulässig.
DSM_RC_FS_NOT_REGISTERED (2061)	Dateibereich nicht registriert.
DSM_RC_WRONG_VERSION_PARM (2065)	Die API-Version des Anwendungsclients stimmt nicht mit der Version der IBM Spectrum Protect-Bibliothek überein.
DSM_RC_NEEDTO_ENDTXN (2070)	Die Transaktion muss beendet werden.
DSM_RC_OBJ_EXCLUDED (2080)	Das Objekt wurde von der Einschluss-/Ausschlussliste ausgeschlossen.
DSM_RC_OBJ_NOBCG (2081)	Das Objekt hat keine Sicherungskopiengruppe und wird nicht an den Server gesendet.
DSM_RC_OBJ_NOACG (2082)	Das Objekt hat keine Archivierungskopiengruppe und wird nicht an den Server gesendet.
DSM_RC_DESC_TOOLONG (2100)	Die Beschreibung ist zu lang.
DSM_RC_OBJINFO_TOOLONG (2101)	Die Objektinformationen sind zu lang.
DSM_RC_HL_TOOLONG (2102)	Das übergeordnete Qualifikationsmerkmal ist zu lang.
DSM_RC_FILESPACE_TOOLONG (2104)	Der Dateibereichsname ist zu lang.
DSM_RC_LL_TOOLONG (2105)	Das untergeordnete Qualifikationsmerkmal ist zu lang.
DSM_RC_NEEDTO_CALL_BINDMC (2301)	**dsmBindMC** muss zuerst aufgerufen werden.

dsmSetAccess

Mit dem Funktionsaufruf **dsmSetAccess** wird anderen Benutzern oder Knoten Zugriff auf Sicherungsversionen oder archivierte Kopien Ihrer Objekte, Zugriff auf alle Ihre Objekte oder Zugriff auf eine ausgewählte Gruppe von Objekten erteilt. Wenn Sie einem anderen Benutzer Zugriff erteilen, kann dieser Benutzer Ihre Dateien abfragen, zurückschreiben oder abrufen. Dieser Befehl unterstützt Platzhalterzeichen für die folgenden Felder: *fs*, *hl*, *ll*, *node*, *owner*.

Anmerkung: Es ist nicht möglich, in einem einzigen Befehl sowohl Zugriff auf Sicherungsversionen als auch Zugriff auf Archivierungskopien zu erteilen. Sie müssen entweder Sicherung oder Archivierung angeben.

Syntax

```
dsInt16_t DSMLINKAGE dsmSetAccess
        (dsUint32_t          dsmHandle,
         dsmSetAccessType    accessType,
         dsmObjName          *objNameP,
         char                *node,
         char                *owner);
```

Parameter

dsUint32_t dsmHandle (I)
Die Kennung, die diesen Aufruf einem vorherigen Aufruf **dsmInitEx** zuordnet.

dsmAccessType accessType (I)
Dieser Parameter gibt den Typ von Objekten an, für die Zugriff erteilt werden soll. Gültige Werte umfassen:

Name	Beschreibung
atBackup	Gibt an, dass der Zugriff für Sicherungsobjekte erteilt wird.
atArchive	Gibt an, dass der Zugriff für Archivierungsobjekte erteilt wird.

dsmObjName *objNameP (I)
Dieser Parameter ist ein Verweis auf die Struktur, die den Dateibereichsnamen, den übergeordneten Objektnamen und den untergeordneten Objektnamen enthält.

Anmerkung: Um alle Dateibereiche anzugeben, verwenden Sie einen Stern (*) für den Dateibereichsnamen.

char *node (I)
Dieser Parameter ist ein Verweis auf den Knotennamen, für den Zugriff erteilt wird. Geben Sie für alle Knoten einen Stern (*) an.

char *owner (I)
Dieser Parameter ist ein Verweis auf den Benutzernamen in dem Knoten, für den Zugriff erteilt wurde. Geben Sie für alle Benutzer einen Stern (*) an.

Rückkehrcodes

Die Rückkehrcodenummern sind in runden Klammern ( ) angegeben.

Tabelle 52. Rückkehrcodes für dsmSetAccess

Rückkehrcode	Erläuterung
DSM_RC_INVALID_ACCESS_TYPE (2110)	Ungültiger Zugriffstyp angegeben.
DSM_RC_FILE_SPACE_NOT_FOUND (124)	Der angegebene Dateibereich wurde auf dem Server nicht gefunden.
DSM_RC_QUERY_COMM_FAILURE (2111)	Kommunikationsfehler während der Serverabfrage.
DSM_RC_NO_FILES_BACKUP (2112)	Für diesen Dateibereich wurden keine Dateien gesichert.
DSM_RC_NO_FILES_ARCHIVE (2113)	Für diesen Dateibereich wurden keine Dateien archiviert.
DSM_RC_INVALID_SETACCESS (2114)	Ungültige Formulierung beim Erteilen des Zugriffs.

dsmSetUp

Mit dem Funktionsaufruf **dsmSetUp** werden Umgebungsvariablenwerte überschrieben. Rufen Sie **dsmSetUp** auf, bevor Sie **dsmInitEx** aufrufen. Die Werte, die in der Struktur **envSetUp** übergeben wurden, überschreiben alle vorhandenen Umgebungsvariablen oder Standardwerte. Wenn Sie NULL für ein Feld angeben, werden die Werte aus der Umgebung übernommen. Wenn Sie keinen Wert definieren, werden die Werte aus den Standardwerden übernommen.

Voraussetzungen:

1. Wenn Sie **dsmSetUp** verwenden, rufen Sie immer **dsmTerminate** auf, bevor Sie **dsmCleanUp** aufrufen.
2. Die API-Instrumentierung kann nur aktiviert werden, wenn die API 'testflag INSTRUMENT:' in der Konfigurationsdatei definiert ist und die Aufrufe **dsmSetUp** oder **dsmCleanUp** in der Anwendung verwendet werden.

Syntax

```
dsInt16_t DSMLINKAGE dsmSetUp
        (dsBool_t    mtFlag,
         envSetUp    *envSetUpP);
```

Parameter

dsBool_t mtFlag (I)
Dieser Parameter gibt an, ob die API in einem Einzelthread- oder in einem Multithread-Modus verwendet wird. Gültige Werte umfassen:

```
DSM_SINGLETHREAD
DSM_MULTITHREAD
```

Voraussetzung: Das Flag für Multithread muss aktiviert sein, damit eine LAN-unabhängige Datenübertragung erfolgen kann.

envSetUp *envSetUpP(I)
Dieser Parameter ist ein Verweis auf die Struktur, die die Überschreibungswerte enthält. Geben Sie NULL an, wenn vorhandene Umgebungsvariablen nicht überschrieben werden sollen. Die Struktur **envSetUp** enthält folgende Felder:

Name	Beschreibung
dsmiDir	Ein vollständig qualifizierter Verzeichnispfad, der unter UNIX oder Linux eine Nachrichtendatei enthält. Dieses Feld gibt außerdem die Verzeichnisse für dsmtca und dsm.sys an.
dsmiConfig	Der vollständig qualifizierte Name der Clientoptionsdatei.
dsmiLog	Der vollständig qualifizierte Pfad des Fehlerprotokollverzeichnisses.
argv	Übergeben Sie den argv[0]-Namen des aufrufenden Programms, wenn die Anwendung mit der Berechtigung eines berechtigten Benutzers ausgeführt werden muss. Weitere Informationen finden Sie in „Option passwordaccess ohne TCA auf generate setzen" auf Seite 20.
logName	Der Dateiname für ein Fehlerprotokoll, sofern die Anwendung nicht dsierror.log verwendet.
inclExclCaseSensitive	Gibt an, ob bei Einschluss-/Ausschlussregeln die Groß-/Kleinschreibung beachtet werden muss oder nicht. Dieser Parameter kann nur unter Windows verwendet werden; für andere Betriebssysteme wird er ignoriert.

Rückkehrcodes

Die Rückkehrcodenummern sind in runden Klammern ( ) angegeben.

Tabelle 53. Rückkehrcodes für dsmSetUp

Rückkehrcode	Erläuterung
DSM_RC_ACCESS_DENIED (106)	Der Zugriff auf die angegebene Datei oder das angegebene Verzeichnis wird verweigert.
DSM_RC_INVALID_OPT (0400)	Es wurde eine ungültige Option gefunden.
DSM_RC_NO_HOST_ADDR (0405)	TCPSERVERADDRESS für diesen Server ist in der Zeilengruppe für den Servernamen in der Systemoptionsdatei nicht definiert.
DSM_RC_NO_OPT_FILE (0406)	Die über den Dateinamen angegebene Optionsdatei kann nicht gefunden werden.
DSM_RC_MACHINE_SAME (0408)	Der in der Optionsdatei definierte Knotenname (NODENAME) darf nicht mit dem Hostnamen (*HostName*) des Systems übereinstimmen.
DSM_RC_INVALID_SERVER (0409)	Die Systemoptionsdatei enthält nicht die Option SERVERNAME.

Tabelle 53. Rückkehrcodes für dsmSetUp (Forts.)

Rückkehrcode	Erläuterung
DSM_RC_INVALID_KEYWORD (0410)	In der Konfigurationsdatei, der Optionszeichenfolge, der Datei dsm.sys oder der Datei dsm.opt wurde für **dsmInitEx** ein ungültiges Optionsschlüsselwort gefunden.
DSM_RC_PATTERN_TOO_COMPLEX (0411)	Das ausgegebene Einschluss- oder Ausschlussmuster ist zu komplex und kann daher von IBM Spectrum Protect nicht korrekt interpretiert werden.
DSM_RC_NO_CLOSING_BRACKET (0412)	Das Einschluss- oder Ausschlussmuster wurde nicht korrekt erstellt. Die rechte eckige Klammer fehlt.
DSM_RC_NLS_CANT_OPEN_TXT (0610)	Das System kann die Nachrichtentextdatei nicht öffnen.
DSM_RC_NLS_INVALID_CNTL_REC (0612)	Das System kann die Nachrichtentextdatei nicht verwenden.
DSM_RC_NOT_ADSM_AUTHORIZED (0927)	Sie müssen der berechtigte Benutzer sein, um Multithreading und '*passwordaccess* generate' verwenden zu können.
DSM_RC_NO_INCLEXCL_FILE (2229)	Die Einschluss-/Ausschlussdatei wurde nicht gefunden.
DSM_RC_NO_SYS_OR_INCLEXCL (2230)	Die Datei dsm.sys oder die Einschluss-/Ausschlussdatei wurde nicht gefunden.

dsmTerminate

Mit dem Funktionsaufruf **dsmTerminate** wird eine Sitzung mit dem IBM Spectrum Protect-Server beendet und die IBM Spectrum Protect-Umgebung bereinigt.

Syntax

Es gibt keine spezifischen Rückkehrcodes für diesen Aufruf.

```
dsInt16_t dsmTerminate (dsUint32_t dsmHandle);
```

Parameter

dsUint32_t dsmHandle (I)
Die Kennung, die diesen Aufruf einem vorherigen Aufruf **dsmInitEx** zuordnet.

dsmUpdateFS

Mit dem Funktionsaufruf **dsmUpdateFS** wird ein Dateibereich im IBM Spectrum Protect-Speicher aktualisiert. Diese Aktualisierung stellt sicher, das der Administrator über einen aktuellen Satz Ihres Dateibereichs verfügt.

Syntax

```
dsInt16_t dsmUpdateFS  (dsUint32_t      dsmHandle,
   char         *fs,
   dsmFSUpd     *fsUpdP,
   dsUint32_t    fsUpdAct);
```

Parameter

dsUint32_t dsmHandle (I)
Die Kennung, die diesen Aufruf einem vorherigen Aufruf **dsmInitEx** zuordnet.

char *fs (I)
Dieser Parameter ist ein Verweis auf den Dateibereichsnamen.

dsmFSUpd *fsUpdP (I)
Dieser Parameter ist ein Verweis auf die Struktur mit den richtigen Feldern für die gewünschte Aktualisierung. Geben Sie nur Werte für die Felder an, die aktualisiert werden müssen.

dsUint32_t fsUpdAct (I)
Eine 2-Byte-Bitzuordnung, die angibt, welche Felder aktualisiert werden müssen. Die Bitmasken haben die folgenden Werte:

- DSM_FSUPD_FSTYPE
- DSM_FSUPD_FSINFO

  Tipp: Bei Windows-Betriebssystemen wird auch der Wert für den Laufwerkbuchstaben in **dsmDOSAttrib** aktualisiert, wenn **FSINFO** ausgewählt wird.
- DSM_FSUPD_OCCUPANCY
- DSM_FSUPD_CAPACITY
- DSM_FSUPD_BACKSTARTDATE
- DSM_FSUPD_BACKCOMPLETEDATE

Eine Beschreibung dieser Bitmasken finden Sie in den DSM_FSUPD-Definitionen hier: Anhang B, „Quellendateien für API-Typdefinitionen", auf Seite 171.

Rückkehrcodes

In der folgenden Tabelle werden Rückkehrcodes für den Funktionsaufruf **dsmUpdateFS** aufgelistet.

Tabelle 54. Rückkehrcodes für dsmUpdateFS

Rückkehrcode	Rückkehrcodenummer	Beschreibung
DSM_RC_FS_NOT_REGISTERED	2061	Dateibereichsname ist nicht registriert.
DSM_RC_WRONG_VERSION_PARM	2065	Die API-Version des Anwendungsclients stimmt nicht mit der Version der IBM Spectrum Protect-Bibliothek überein.
DSM_RC_FSINFO_TOOLONG	2106	Dateibereichsinformationen sind zu lang.

dsmUpdateObj

Mit dem Funktionsaufruf **dsmUpdateObj** werden die Metainformationen aktualisiert, die einem aktiven Sicherungs- oder Archivierungsobjekt zugeordnet sind, das sich bereits auf dem Server befindet. Der Aufruf hat keine Auswirkungen auf die Anwendungsbitdaten. Um ein Objekt aktualisieren zu können, müssen Sie einen bestimmten Namen ohne Platzhalterzeichen angeben. Um ein archiviertes Objekt zu aktualisieren, setzen Sie **dsmSendType** auf **stArchive**. Es wird nur das letzte benannte Archivierungsobjekt aktualisiert.

Sie können den Aufruf **dsmUpdateObj** nur im Sitzungsstatus starten; er kann nicht innerhalb einer Transaktion aufgerufen werden, da er seine eigene Transaktion ausführt. Außerdem können Sie jeweils nur ein einziges Objekt aktualisieren.

Einschränkung: Auf einem Betriebssystem UNIX oder Linux können Sie, wenn Sie das Feld für den Eigner ändern, das Objekt nur dann abfragen oder zurückschreiben, wenn Sie der Rootbenutzer sind.

Syntax

```
dsInt16_t  dsmUpdateObj
   (dsUint32_t      dsmHandle,
    dsmSendType     sendType,
    void           *sendBuff,
    dsmObjName     *objNameP,
    ObjAttr        *objAttrPtr,  /* objInfo */
    dsUint16_t      objUpdAct);  /* Aktionsbitvektor */
```

Parameter

Die Feldbeschreibungen sind mit denen von **dsmSendObj** bis auf die folgenden Ausnahmen identisch:

dsmObjName *objNameP (I)
Sie können kein Platzhalterzeichen verwenden.

ObjAttr *objAttrPtr (I)
Das Feld **objCompressed** wird für diesen Aufruf ignoriert.

Weitere Unterschiede sind:

- **owner**. Wenn Sie ein neues Feld **owner** angeben, ändert sich der Eigner.
- **sizeEstimate**. Wenn Sie einen Wert ungleich null angeben, muss das tatsächlich gesendete Datenvolumen in Byte angegeben werden. Der Wert wird für die zukünftige Verwendung in den IBM Spectrum Protect-Metadaten gespeichert.
- **objInfo**. Dieses Attribut enthält die neuen Informationen, die in das Feld **objInfo** gestellt werden sollen. Setzen Sie **objInfoLength** auf die Länge für die neuen Objektinformationen (**objInfo**).

dsUint16_t objUpdAct
Die Bitmasken und gültigen Aktionen für **objUpdAct** sind wie folgt:

DSM_BACKUPD_MC
Aktualisiert die Verwaltungsklasse für das Objekt.

DSM_BACKUPD_OBJINFO
Aktualisiert **objInfo**, **objInfoLength** und **sizeEstimate**.

DSM_BACKUPD_OWNER
Aktualisiert den Eigner des Objekts.

DSM_ARCHUPD_DESCR
Aktualisiert das Feld **Description**. Geben Sie den Wert für die neue Beschreibung über den Parameter **SendBuff** ein. Siehe das Musterprogramm zur korrekten Verwendung.

DSM_ARCHUPD_OBJINFO
Aktualisiert **objInfo**, **objInfoLength** und **sizeEstimate**.

DSM_ARCHUPD_OWNER
Aktualisiert den Eigner des Objekts.

Rückkehrcodes

Die Rückkehrcodenummern sind in runden Klammern ( ) angegeben.

Tabelle 55. Rückkehrcodes für dsmUpdateObj

Rückkehrcode	Erläuterung
DSM_RC_INVALID_ACTION (2232)	Ungültige Aktion.

Tabelle 55. Rückkehrcodes für dsmUpdateObj (Forts.)

Rückkehrcode	Erläuterung
DSM_RC_FS_NOT_REGISTERED (2061)	Dateibereich nicht registriert.
DSM_RC_BAD_CALL_SEQUENCE (2041)	Aufruffolge ist ungültig.
DSM_RC_WILDCHAR_NOTALLOWED (2050)	Es sind keine Platzhalterzeichen zulässig.
DSM_RC_ABORT_NO_MATCH (2)	Vorherige Abfrage stimmt nicht überein.

dsmUpdateObjEx

Mit dem Funktionsaufruf **dsmUpdateObjEx** werden die Metainformationen aktualisiert, die einem aktiven Sicherungs- oder Archivierungsobjekt zugeordnet sind, das sich auf dem Server befindet. Der Aufruf hat keine Auswirkungen auf die Anwendungsbitdaten. Um ein Objekt aktualisieren zu können, müssen Sie einen Namen ohne Platzhalterzeichen angeben; Sie können auch die Objekt-ID angeben, um ein bestimmtes archiviertes Objekt zu aktualisieren. Bei der Angabe des Namens können Sie keine Platzhalterzeichen verwenden. Um ein Sicherungsobjekt zu aktualisieren, setzen Sie den Parameter **dsmSendType** auf **stBackup**. Um ein archiviertes Objekt zu aktualisieren, setzen Sie den Parameter **dsmSendType** auf **stArchive**.

Sie können den Aufruf **dsmUpdateObjEx** nur im Sitzungsstatus starten; er kann nicht innerhalb einer Transaktion aufgerufen werden, da er seine eigene Transaktion ausführt. Sie können jeweils nur ein einziges Objekt aktualisieren.

Einschränkung: Auf einem Betriebssystem UNIX oder Linux können Sie, wenn Sie das Feld für den Eigner ändern, das Objekt nur dann abfragen oder zurückschreiben, wenn Sie der Rootbenutzer sind. Es kann nur die momentan aktive Version eines Sicherungsobjekts aktualisiert werden.

Syntax

```
dsInt16_t  dsmUpdateObjEx
   (dsmUpdateObjExIn_t   *dsmUpdateObjExInP,
    dsmUpdateObjExOut_t  *dsmUpdateObjExOutP);
```

Parameter

dsmUpdateObjExIn_t *dsmUpdateObjExInP
Diese Struktur enthält die folgenden Eingabeparameter:

dsUint16_t stVersion (I)
Die aktuelle Version der Struktur, die verwendet wird.

dsUint32_t dsmHandle (I)
Die Kennung, die diesen Aufruf einem vorherigen Aufruf **dsmInitEx** zuordnet.

dsmSendType sendType (I)
Der Sendetyp, der ausgeführt wird. Gültige Werte sind:

stBackup
Ein Sicherungsobjekt, das an den Server gesendet wird.

stArchive
Ein Archivierungsobjekt, das an den Server gesendet wird.

dsmObjName *objNameP (I)
Ein Verweis auf die Struktur, die den Dateibereichsnamen, den übergeord-

neten Objektnamen, den untergeordneten Objektnamen und den Objekttyp enthält. Sie können kein Platzhalterzeichen verwenden.

ObjAttr *objAttrPtr (I)
Übergibt Objektattribute an die Anwendung. Welche Werte aktualisiert werden, ist von den Flags im Feld **objUpdAct** abhängig. Das Attribut **objCompressed** wird für diesen Aufruf ignoriert.

Die Attribute sind:

- **owner**; wenn ein neuer Name eingegeben wird, ändert sich der Eigner.
- **sizeEstimate** ist das tatsächlich gesendete Datenvolumen in Byte. Der Wert wird für die zukünftige Verwendung in den IBM Spectrum Protect-Metadaten gespeichert.
- **objCompressed** ist ein boolescher Wert, der angibt, ob die Objektdaten bereits komprimiert wurden oder nicht.
- **objInfo** ist ein Attribut, das die neuen Informationen enthält, die in das Feld **objInfo** gestellt werden sollen. Setzen Sie **objInfoLength** auf die Länge für die neuen Objektinformationen (**objInfo**).
- **mcNameP** enthält den Namen einer Verwaltungsklasse, die die mit **dsmBindMC** abgerufene Verwaltungsklasse überschreibt.

dsUint32_t objUpdAct
Gibt die Bitmasken und Aktionen für **objUpdAct** an:

DSM_BACKUPD_MC
Aktualisiert die Verwaltungsklasse für das Objekt.

DSM_BACKUPD_OBJINFO
Aktualisiert die Objektinformationen (**objInfo**), die Länge der Objektinformationen (**objInfoLength**) und das gesendete Datenvolumen (**sizeEstimate**) für das Sicherungsobjekt.

DSM_BACKUPD_OWNER
Aktualisiert den Eigner des Sicherungsobjekts.

DSM_ARCHUPD_DESCR
Aktualisiert das Feld **Description** für das Archivierungsobjekt. Geben Sie den Wert für die neue Beschreibung über den Parameter **sendBuff** ein.

DSM_ARCHUPD_OBJINFO
Aktualisiert die Objektinformationen (**objInfo**), die Länge der Objektinformationen (**objInfoLength**) und das gesendete Datenvolumen (**sizeEstimate**) für das Archivierungsobjekt.

DSM_ARCHUPD_OWNER
Aktualisiert den Eigner des Archivierungsobjekts.

ObjID archObjId
Gibt die eindeutige Objekt-ID für ein bestimmtes Archivierungsobjekt an. Da mehrere Archivierungsobjekte denselben Namen haben können, kann über diesen Parameter ein bestimmtes Archivierungsobjekt identifiziert werden. Sie können die Objekt-ID mithilfe eines Abfragearchivaufrufs abrufen.

dsmUpdateObjExOut_t *dsmUpdateObjExOutP
Diese Struktur enthält den Ausgabeparameter:

dsUint16_t stVersion (I)
Die aktuelle Version der Struktur, die verwendet wird.

Rückkehrcodes

Die Rückkehrcodenummern sind in der folgenden Tabelle in runden Klammern ( ) angegeben.

Tabelle 56. Rückkehrcodes für dsmUpdateObjEx

Rückkehrcode	Erläuterung
DSM_RC_INVALID_ACTION (2012)	Ungültige Aktion.
DSM_RC_FS_NOT_REGISTERED (2061)	Dateibereich nicht registriert.
DSM_RC_BAD_CALL_SEQUENCE (2041)	Aufruffolge ist ungültig.
DSM_RC_WILDCHAR_NOTALLOWED (2050)	Es sind keine Platzhalterzeichen zulässig.
DSM_RC_ABORT_NO_MATCH (2)	Vorherige Abfrage stimmt nicht überein.

Anhang A. Quellendatei mit API-Rückkehrcodes: dsmrc.h

Die Headerdatei dsmrc.h enthält alle Rückkehrcodes, die die API an eine Anwendung zurückgeben kann.

Die hier bereitgestellten Informationen enthalten eine Zeitpunktkopie der Datei dsmrc.h, die mit der API verteilt wird. Die neueste Version können Sie der Datei im API-Verteilerpaket entnehmen.

```
/*****************************************************************
* Tivoli Storage Manager                                         *
* API-Clientkomponente                                           *
*                                                                *
* (C) Copyright IBM Corporation 1993, 2010                       *
*****************************************************************/

/*****************************************************************/
/* Headerdateiname:     dsmrc.h                                  */
/*                                                               */
/* Beschreibender Name: Rückkehrcodes von 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 /* erfolgreiche Ausführung */
#define DSM_RC_OK                       0 /* erfolgreiche Ausführung */

#define DSM_RC_UNSUCCESSFUL            -1 /* nicht erfolgr. Ausführ. */

/* dsmEndTxn Ursachencode */
#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    /* Teilobjektabruf         */
#define DSM_RS_ABORT_INVALID_LENGTH          34    /* Teilobjektabruf         */
#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


/* RÜCKKEHRCODE */

#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


/* Definitionen für Codes für Zurückweisung der Serveranmeldung        */
/* Diese Fehlercodes liegen zwischen 51 und 99 einschließlich.         */
#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 /* Clienttyp nicht lizenz.  */
#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 /* Verarbeitung vom Ben. abgebrochen */
#define DSM_RC_NO_MEMORY           102 /* kein RAM für Anforderungsausführ. */
#define DSM_RC_TA_COMM_DOWN       2021 /* nicht mehr verwendet              */
#define DSM_RC_FILE_NOT_FOUND      104 /* angegebene Datei nicht gefunden   */
#define DSM_RC_PATH_NOT_FOUND      105 /* angegebener Pfad nicht vorhanden  */
#define DSM_RC_ACCESS_DENIED       106 /* wg. falscher Berecht. zurückgew.  */
#define DSM_RC_NO_HANDLES          107 /* keine Dateikennungen mehr verfügb.*/
#define DSM_RC_FILE_EXISTS         108 /* Datei ist bereits vorhanden       */
#define DSM_RC_INVALID_PARM        109 /* ungült. Param. übergeben. KRITISCH*/
#define DSM_RC_INVALID_HANDLE      110 /* ungültige Dateikennung übergeben  */
#define DSM_RC_DISK_FULL           111 /* kein Plattenspeicherplatz verfügb.*/
#define DSM_RC_PROTOCOL_VIOLATION  113 /* fehlerh. Protokollaufruf. KRITISCH*/
#define DSM_RC_UNKNOWN_ERROR       114 /* unbekannter Systemfehler. KRITISCH*/
#define DSM_RC_UNEXPECTED_ERROR    115 /* unerwarteter Fehler. KRITISCH     */
#define DSM_RC_FILE_BEING_EXECUTED 116 /* Schreiben nicht zulässig          */
#define DSM_RC_DIR_NO_SPACE        117 /* Verz. kann nicht erweitert werden */
#define DSM_RC_LOOPED_SYM_LINK     118 /* zu viele symbolische Verbindungen
                                          beim Umsetzen des Pfads gefunden  */
#define DSM_RC_FILE_NAME_TOO_LONG  119 /* Dateiname zu lang                 */
#define DSM_RC_FILE_SPACE_LOCKED   120 /* Dateibereich vom System gesperrt  */
#define DSM_RC_FINISHED            121 /* Verarbeitung beendet              */
#define DSM_RC_UNKNOWN_FORMAT      122 /* unbekanntes Format                */
#define DSM_RC_NO_AUTHORIZATION    123 /* Serverantwort, wenn Client keine
                                          Berechtigung zum Lesen der Daten
                                          des Eigners eines anderen Hosts
                                          für Sichern/Archivieren hat       */
#define DSM_RC_FILE_SPACE_NOT_FOUND 124/* angeg. Dateibereich nicht gefunden*/
#define DSM_RC_TXN_ABORTED         125 /* Transaktion abgebrochen           */
#define DSM_RC_SUBDIR_AS_FILE      126 /* Unterverzeichnisname als Datei
                                          vorhanden                         */
#define DSM_RC_PROCESS_NO_SPACE    127 /* kein weiterer Plattenspeicherplatz
                                          für Prozess vorhanden             */
#define DSM_RC_PATH_TOO_LONG       128 /* ein erstellter Verzeichnispfad ist
                                          zu lang                           */
#define DSM_RC_NOT_COMPRESSED      129 /* Datei, die komprimiert sein sollte,
                                          ist nicht komprimiert             */
#define DSM_RC_TOO_MANY_BITS       130 /* Datei mit mehr Bit komprimiert,
                                          als Expander handhaben            */
#define DSM_RC_SYSTEM_ERROR        131 /* interner Systemfehler             */
#define DSM_RC_NO_SERVER_RESOURCES 132 /* keine Ressourcen für Server verf. */
#define DSM_RC_FS_NOT_KNOWN        133 /* Dateibereich ist dem Server nicht
                                          bekannt                           */
#define DSM_RC_NO_LEADING_DIRSEP   134 /* kein führendes Verzeichnistrennz. */
#define DSM_RC_WILDCARD_DIR        135 /* Platzhalterzeichen in Verzeichnis-
                                          pfad an unzulässiger Stelle       */
#define DSM_RC_COMM_PROTOCOL_ERROR 136 /* Übertragungsprotokollfehler       */
#define DSM_RC_AUTH_FAILURE        137 /* Authentifizierungsfehler          */
#define DSM_RC_TA_NOT_VALID        138 /* TA kein Root und/od. SUID-Programm*/
#define DSM_RC_KILLED              139 /* Prozess abgebrochen               */

#define DSM_RC_RETRY               143 /* dieselbe Operation wiederholen    */

#define DSM_RC_WOULD_BLOCK         145 /* Operation hätte das Blockieren des
                                          Systems zur Folge, um auf Eingabe
                                          zu warten.                        */
#define DSM_RC_TOO_SMALL           146 /* Bereich f. kompil. Muster zu klein*/
#define DSM_RC_UNCLOSED            147 /* keine rechte eckige Klammer in
                                          Muster                            */
#define DSM_RC_NO_STARTING_DELIMITER 148 /* Muster muss mit Verzeichnis-
                                            begrenzer beginnen              */
#define DSM_RC_NEEDED_DIR_DELIMITER 149 /* direkt vor und hinter der Meta-
                                           zeichenfolge ("...") für den
                                           Verzeichnisabgleich ist ein Ver-
                                           zeichnisbegrenzer erforderlich,
                                           es wurde jedoch keiner gefunden   */
#define DSM_RC_UNKNOWN_FILE_DATA_TYPE 150 /* strukturierter Dateidatentyp
                                             ist unbekannt                  */
#define DSM_RC_BUFFER_OVERFLOW     151 /* Datenpufferüberlauf               */


#define DSM_RC_NO_COMPRESS_MEMORY  154 /* kein Speicher für Erweitern/Kompr.*/
#define DSM_RC_COMPRESS_GREW       155 /* Komprimierung gewachsen           */
```

```
#define DSM_RC_INV_COMM_METHOD     156 /* Ungültige Übertragungsmethode ang.*/
#define DSM_RC_WILL_ABORT          157 /* Transaktion wird abgebrochen      */
#define DSM_RC_FS_WRITE_LOCKED     158 /* Schreibsperre für Dateibereich    */
#define DSM_RC_SKIPPED_BY_USER     159 /* Benutzeranforderung, Datei bei
                                           ABORT_DATA_OFFLINE zu überspringen*/
#define DSM_RC_TA_NOT_FOUND        160 /* TA in Verzeichnis nicht gefunden  */
#define DSM_RC_TA_ACCESS_DENIED    161 /* Zugriff auf TA verweigert         */
#define DSM_RC_FS_NOT_READY        162 /* Dateibereich nicht bereit         */
#define DSM_RC_FS_IS_BAD           163 /* Dateibereich ungültig             */
#define DSM_RC_FIO_ERROR           164 /* Dateiein-/ausgabefehler           */
#define DSM_RC_WRITE_FAILURE       165 /* Fehler beim Schreiben in Datei    */
#define DSM_RC_OVER_FILE_SIZE_LIMIT 166 /* Datei über System-/Benutzer-
                                           Grenzwert                         */
#define DSM_RC_CANNOT_MAKE         167 /* Datei/Verzeichnis konnte wegen
                                           eines ungültigen Namens nicht
                                           erstellt werden                   */
#define DSM_RC_NO_PASS_FILE        168 /* Kennwortdatei erforderlich und
                                           Benutzer ist nicht Rootbenutzer   */
#define DSM_RC_VERFILE_OLD         169 /* lokal gespeichertes Kennwort stimmt
                                           nicht mit dem auf Host überein    */
#define DSM_RC_INPUT_ERROR         173 /* Tastatureingabe kann nicht gelesen
                                           werden                            */
#define DSM_RC_REJECT_PLATFORM_MISMATCH 174 /* Plattformname stimmt nicht mit
                                              der Angabe überein, die laut
                                              Server die Plattform für den
                                              Client ist                     */
#define DSM_RC_TL_NOT_FILE_OWNER   175 /* Benutzer, der versucht, Datei zu
                                           sichern, ist nicht Dateieigner.   */
#define DSM_RC_COMPRESSED_DATA_CORRUPTED 176 /* komprim. Daten beschädigt   */
#define DSM_RC_UNMATCHED_QUOTE     177 /* fehlendes Anführungszeichen am
                                           Anfang oder Ende                  */

#define DSM_RC_SIGNON_FAILOVER_MODE 178 /* Übernahme auf Replikationsserver,
                                           aktiv im Übernahmemodus     */
#define DSM_RC_FAILOVER_MODE_FUNC_BLOCKED 179 /* Funktion blockiert, weil
                                             Sitzung im Übernahmemodus  */

/*------------------------------------------------------------------------*/
/* Rückkehrcodes 180-199 sind für Handhabung von Maßnahmengruppen reserviert */
/*------------------------------------------------------------------------*/
#define DSM_RC_PS_MULTBCG          181 /* Mehrere Sicherungskopiengruppen in
                                           1 Verwaltungsklasse               */
#define DSM_RC_PS_MULTACG          182 /* Mehrere Archivierungskopiengruppen
                                           in 1 Verwaltungsklasse            */
#define DSM_RC_PS_NODFLTMC         183 /* Standardverwaltungsklassenname
                                           nicht in Maßnahmengruppe          */
#define DSM_RC_TL_NOBCG            184 /* Sicherung erforderlich; keine
                                           Sicherungskopiengruppe            */
#define DSM_RC_TL_EXCLUDED         185 /* Sicherung erforderlich; von Ein-/
                                           Ausschlussfilter ausgeschlossen   */
#define DSM_RC_TL_NOACG            186 /* Archivierung erforderlich; keine
                                           Archivierungskopiengruppe         */
#define DSM_RC_PS_INVALID_ARCHMC   187 /* Ungültiger Verwaltungsklassenname
                                           in Archivierungsüberschreibung    */
#define DSM_RC_NO_PS_DATA          188 /* Keine Maßnahmengruppendaten auf dem
                                           Server                            */
#define DSM_RC_PS_INVALID_DIRMC    189 /* Ungültiges Verzeichnis MC in
                                           Optionsdatei angegeben            */
#define DSM_RC_PS_NO_CG_IN_DIR_MC  190 /* Keine Sicherungskopiengruppe in
                                           Verzeichnis MC. Verwaltungsklasse
                                           muss mit Option DirMC ang. werden. */

#define DSM_RC_WIN32_UNSUPPORTED_FILE_TYPE 280 /* Datei hat nicht Win32-Typ
                                             FILE_TYPE_DISK            */

/*------------------------------------------------------------------------*/
/* Rückkehrcodes für Trusted Communication Agent                          */
/*------------------------------------------------------------------------*/
#define DSM_RC_TCA_NOT_ROOT        161 /* Zugriff auf TA verweigert         */
#define DSM_RC_TCA_ATTACH_SHR_MEM_ERR  200 /* Fehler beim Zuordnen von
                                           gemeinsam genutztem Speicher      */
#define DSM_RC_TCA_SHR_MEM_BLOCK_ERR   200 /* Fehler bei Shared-Memory-Block */
#define DSM_RC_TCA_SHR_MEM_IN_USE      200 /* Fehler bei Shared-Memory-Block */
#define DSM_RC_TCA_SHARED_MEMORY_ERROR 200 /* Fehler bei Shared-Memory-Block */
#define DSM_RC_TCA_SEGMENT_MISMATCH    200 /* Fehler bei Shared-Memory-Block */
#define DSM_RC_TCA_FORK_FAILED     292 /* Fehler beim Abzweigen v. TA-Prozess*/
#define DSM_RC_TCA_DIED            294 /* TCA unerwartet inaktiviert        */
#define DSM_RC_TCA_INVALID_REQUEST 295 /* Ungültige Anford. an TCA gesendet  */
#define DSM_RC_TCA_SEMGET_ERROR    297 /* Fehler beim Abrufen von Semaphors  */
```

```
#define DSM_RC_TCA_SEM_OP_ERROR    298 /* Fehler bei Semaphorgruppe oder
                                          Warten                              */
#define DSM_RC_TCA_NOT_ALLOWED     299 /* TCA nicht zulässig (Multithread)   */

/*-------------------------------------------------------------------------*/
/* 400-430  für Optionen                                                   */
/*-------------------------------------------------------------------------*/
#define DSM_RC_INVALID_OPT           400 /* ungültige Option                */
#define DSM_RC_NO_HOST_ADDR          405 /* Nicht genügend Informationen, um
                                            Verbindung zu Server herzustellen*/
#define DSM_RC_NO_OPT_FILE           406 /* Keine Standardbenutzerkonf.-Datei*/
#define DSM_RC_MACHINE_SAME          408 /* -MACHINENAME = realer Name       */
#define DSM_RC_INVALID_SERVER        409 /* ungültiger Servername von Client */
#define DSM_RC_INVALID_KEYWORD       410 /* ungültiges Optionsschlüsselwort  */
#define DSM_RC_PATTERN_TOO_COMPLEX   411 /* Einschluss-/Ausschluss kann nicht
                                            abgeglichen werden               */
#define DSM_RC_NO_CLOSING_BRACKET    412 /* Fehlende rechte eckige Klammer
                                            für Einschluss/Ausschluss        */
#define DSM_RC_OPT_CLIENT_NOT_ACCEPTING 417/* Client akzeptiert diese Option
                                            nicht vom Server                 */
#define DSM_RC_OPT_CLIENT_DOES_NOT_WANT 418/* Client will diesen Wert
                                            nicht vom Server                 */
#define DSM_RC_OPT_NO_INCLEXCL_FILE  419   /* Einschluss-/Ausschlussdatei
                                            nicht gefunden                   */
#define DSM_RC_OPT_OPEN_FAILURE      420   /* Datei kann nicht geöffnet
                                            werden                           */
#define DSM_RC_OPT_INV_NODENAME      421   /* für Windows, wenn nodename=lokale
                                            Maschine, wenn CLUSTERNODE=YES */
#define DSM_RC_OPT_NODENAME_INVALID  423   /* ungült. generischer Knotenname */
#define DSM_RC_OPT_ERRORLOG_CONFLICT 424   /* logmax & Aufbewahrungsdauer
                                            angegeben                        */
#define DSM_RC_OPT_SCHEDLOG_CONFLICT 425   /* logmax & Aufbewahrungsdauer
                                            angegeben                        */
#define DSM_RC_CANNOT_OPEN_TRACEFILE 426   /* Tracedatei kann nicht geöffnet
                                            werden                           */
#define DSM_RC_CANNOT_OPEN_LOGFILE   427   /* Fehlerprotokolldatei kann nicht
                                            geöffnet werden                  */
#define DSM_RC_OPT_SESSINIT_LF_CONFLICT 428/* sessioninit=server und
                                                      enablelanfree=yes angegeben    */
#define DSM_RC_OPT_OPTION_IGNORE     429   /* Option wird ignoriert           */
#define DSM_RC_OPT_DEDUP_CONFLICT    430   /* Fehlerprotokolldatei kann nicht
                                            geöffnet werden                  */
#define DSM_RC_OPT_HSMLOG_CONFLICT 431/* logmax & Aufbewahrungsdauer angegeben */


/*-------------------------------------------------------------------------*/
/* 600 bis 610 für Datenträgerkennsatzcodes                                */
/*-------------------------------------------------------------------------*/
#define DSM_RC_DUP_LABEL           600 /* doppelten Datenträgerkennsatz
                                          gefunden                           */
#define DSM_RC_NO_LABEL            601 /* Laufwerk hat keinen Kennsatz       */

/*-------------------------------------------------------------------------*/
/* Rückkehrcodes für Nachrichtendateiverarbeitung                          */
/*-------------------------------------------------------------------------*/
#define DSM_RC_NLS_CANT_OPEN_TXT   610 /* Fehler beim Öffnen der Nachrichten-
                                          textdatei                          */
#define DSM_RC_NLS_CANT_READ_HDR   611 /* Fehler beim Lesen von Header       */
#define DSM_RC_NLS_INVALID_CNTL_REC 612 /* ungültiger Steuersatz             */
#define DSM_RC_NLS_INVALID_DATE_FMT 613 /* ungültiges Standarddatumsformat   */
#define DSM_RC_NLS_INVALID_TIME_FMT 614 /* ungültiges Standardzeitformat     */
#define DSM_RC_NLS_INVALID_NUM_FMT 615 /* ungültiges Standardzahlenformat    */

/*-------------------------------------------------------------------------*/
/* Rückkehrcodes 620-630 für Rückkehrcodes f. Protokollnachrichten reserviert*/
/*-------------------------------------------------------------------------*/
#define DSM_RC_LOG_CANT_BE_OPENED  620        /* Fehler beim Öffnen des
                                                 Fehlerprotokolls            */
#define DSM_RC_LOG_ERROR_WRITING_TO_LOG 621   /* Fehler beim Schreiben in
                                                 Protokolldatei              */
#define DSM_RC_LOG_NOT_SPECIFIED   622        /* keine Fehlerprotokolldatei
                                                 angegeben                   */


/*-------------------------------------------------------------------------*/
/* Rückkehrcode  900-999 NUR TSM-CLIENT                                    */
/*-------------------------------------------------------------------------*/
#define DSM_RC_NOT_ADSM_AUTHORIZED  927  /* Muss für ADSM berechtigt sein, um
                                            Aktion ausführen zu können: Root
```

```
                                        oder Kennwortberechtigung         */
#define DSM_RC_REJECT_USERID_UNKNOWN 940 /* Benutzer-ID auf Server unbekannt */
#define DSM_RC_FILE_IS_SYMLINK      959  /* Fehlerprotokoll oder Trace ist
                                        eine symbolische Verbindung      */

#define DSM_RC_DIRECT_STORAGE_AGENT_UNSUPPORTED 961 /* Direktverbindung zu
                                                       Speicheragent nicht
                                                       unterstützt           */
#define DSM_RC_FS_NAMESPACE_DOWNLEVEL 963 /* Langer Namensbereich wurde aus
                                        Netware-Datenträger entfernt    */
#define DSM_RC_CONTINUE_NEW_CONSUMER   972 /* Verarbeitung mit neuem
                                        Konsumenten fortsetzen          */
#define DSM_RC_CONTINUE_NEW_CONSUMER_NODEDUP   973 /* Verarbeitung mit neuem
                                               Konsumenten fortsetzen;
                                               keine Deduplizierung   */
#define DSM_RC_CONTINUE_NEW_CONSUMER_NOCOMPRESS 976 /* Verarbeitung mit neuem
                                                       Konsumenten fortsetzen;
                                                       keine Komprimierung   */


#define DSM_RC_SERVER_SUPPORTS_FUNC    994 /* Server unterstützt diese Funktion  */
#define DSM_RC_SERVER_AND_SA_SUPPORT_FUNC 995 /* Server und Speicheragent
                                           unterstützen die Funktion       */
#define DSM_RC_SERVER_DOWNLEVEL_FUNC   996 /* Serverversion ist für Funktion
                                        veraltet                        */
#define DSM_RC_STORAGEAGENT_DOWNLEVEL  997 /* Speicheragentenversion ist veraltet*/
#define DSM_RC_SERVER_AND_SA_DOWNLEVEL 998 /* Server- und Speicheragentenversion
                                        sind veraltet                   */


/* TCP/IP-Fehlercodes */
#define DSM_RC_TCPIP_FAILURE       -50 /* TCP/IP-Übertragungsfehler          */
#define DSM_RC_CONN_TIMEDOUT       -51 /* Zeitlimitüberschreitung bei
                                        TCP/IP-Verbindung                  */
#define DSM_RC_CONN_REFUSED        -52 /* TCP/IP-Verbindung von Host
                                        zurückgewiesen                     */
#define DSM_RC_BAD_HOST_NAME       -53 /* Ungültiger TCP/IP-Hostname angeg.  */
#define DSM_RC_NETWORK_UNREACHABLE -54 /* TCP/IP-Hostname nicht erreichbar   */
#define DSM_RC_WINSOCK_MISSING     -55 /* TCP/IP-Datei WINSOCK.DLL fehlt     */
#define DSM_RC_TCPIP_DLL_LOADFAILURE -56 /* Fehler von LoadLibrary           */
#define DSM_RC_TCPIP_LOADFAILURE   -57 /* Fehler von GetProcAddress          */
#define DSM_RC_TCPIP_USER_ABORT    -58 /* Abbruch durch Benutzer in
                                        TCP/IP-Schicht                     */

/*-----------------------------------------------------------------------------*/
/* Rückkehrcodes (-71)-(-90) sind für CommTSM-Fehlercodes reserviert         */
/*-----------------------------------------------------------------------------*/
#define DSM_RC_TSM_FAILURE         -71 /* TSM-Kommunikationsfehler           */
#define DSM_RC_TSM_ABORT           -72 /* Sitzung abnormal abgebrochen       */

/*comm3270-Fehlercodes - nicht mehr verwendet*/
#define DSM_RC_COMM_TIMEOUT       2021   /* nicht mehr verwendet              */
#define DSM_RC_EMULATOR_INACTIVE  2021   /* nicht mehr verwendet              */
#define DSM_RC_BAD_HOST_ID        2021   /* nicht mehr verwendet              */
#define DSM_RC_HOST_SESS_BUSY     2021   /* nicht mehr verwendet              */
#define DSM_RC_3270_CONNECT_FAILURE 2021 /* nicht mehr verwendet              */
#define DSM_RC_NO_ACS3ELKE_DLL    2021   /* nicht mehr verwendet              */
#define DSM_RC_EMULATOR_ERROR     2021   /* nicht mehr verwendet              */
#define DSM_RC_EMULATOR_BACKLEVEL 2021   /* nicht mehr verwendet              */
#define DSM_RC_CKSUM_FAILURE      2021   /* nicht mehr verwendet              */


/* Die folgenden Rückkehrcode gelten für EHLLAPI für Windows                 */
#define DSM_RC_3270COMMError_DLL        2021   /* nicht mehr verwendet       */
#define DSM_RC_3270COMMError_GetProc    2021   /* nicht mehr verwendet       */
#define DSM_RC_EHLLAPIError_DLL         2021   /* nicht mehr verwendet       */
#define DSM_RC_EHLLAPIError_GetProc     2021   /* nicht mehr verwendet       */
#define DSM_RC_EHLLAPIError_HostConnect 2021   /* nicht mehr verwendet       */
#define DSM_RC_EHLLAPIError_AllocBuff   2021   /* nicht mehr verwendet       */
#define DSM_RC_EHLLAPIError_SendKey     2021   /* nicht mehr verwendet       */
#define DSM_RC_EHLLAPIError_PacketChk   2021   /* nicht mehr verwendet       */
#define DSM_RC_EHLLAPIError_ChkSum      2021   /* nicht mehr verwendet       */
#define DSM_RC_EHLLAPIError_HostTimeOut 2021   /* nicht mehr verwendet       */
#define DSM_RC_EHLLAPIError_Send        2021   /* nicht mehr verwendet       */
#define DSM_RC_EHLLAPIError_Recv        2021   /* nicht mehr verwendet       */
#define DSM_RC_EHLLAPIError_General     2021   /* nicht mehr verwendet       */
#define DSM_RC_PC3270_MISSING_DLL       2021   /* nicht mehr verwendet       */
#define DSM_RC_3270COMM_MISSING_DLL     2021   /* nicht mehr verwendet       */
```

```
/* NETBIOS-Fehlercodes */
#define DSM_RC_NETB_ERROR         -151 /* Knoten konnte LAN nicht hinzugefügt
                                          werden                          */
#define DSM_RC_NETB_NO_DLL        -152 /* ACSNETB.DLL konnte nicht geladen
                                          werden                          */
#define DSM_RC_NETB_LAN_ERR       -155 /* LAN-Fehler erkannt              */
#define DSM_RC_NETB_NAME_ERR      -158 /* NetBIOS-Fehler im Hinzufügenamen  */
#define DSM_RC_NETB_TIMEOUT       -159 /* Zeitlimitüberschreitung in
                                          NetBIOS-Sendeoperation          */
#define DSM_RC_NETB_NOTINST       -160 /* NetBIOS nicht installiert - DOS   */
#define DSM_RC_NETB_REBOOT        -161 /* NetBIOS-Konfigurationsfehler -
                                          Warmstart für DOS durchführen   */

/* Fehlercodes für benannte Pipe */
#define DSM_RC_NP_ERROR                    -190

/* CPIC-Fehlercodes */
#define DSM_RC_CPIC_ALLOCATE_FAILURE       2021 /* nicht mehr verwendet*/
#define DSM_RC_CPIC_TYPE_MISMATCH          2021 /* nicht mehr verwendet*/
#define DSM_RC_CPIC_PIP_NOT_SPECIFY_ERR    2021 /* nicht mehr verwendet*/
#define DSM_RC_CPIC_SECURITY_NOT_VALID     2021 /* nicht mehr verwendet*/
#define DSM_RC_CPIC_SYNC_LVL_NO_SUPPORT    2021 /* nicht mehr verwendet*/
#define DSM_RC_CPIC_TPN_NOT_RECOGNIZED     2021 /* nicht mehr verwendet*/
#define DSM_RC_CPIC_TP_ERROR               2021 /* nicht mehr verwendet*/
#define DSM_RC_CPIC_PARAMETER_ERROR        2021 /* nicht mehr verwendet*/
#define DSM_RC_CPIC_PROD_SPECIFIC_ERR      2021 /* nicht mehr verwendet*/
#define DSM_RC_CPIC_PROGRAM_ERROR          2021 /* nicht mehr verwendet*/
#define DSM_RC_CPIC_RESOURCE_ERROR         2021 /* nicht mehr verwendet*/
#define DSM_RC_CPIC_DEALLOCATE_ERROR       2021 /* nicht mehr verwendet*/
#define DSM_RC_CPIC_SVC_ERROR              2021 /* nicht mehr verwendet*/
#define DSM_RC_CPIC_PROGRAM_STATE_CHECK    2021 /* nicht mehr verwendet*/
#define DSM_RC_CPIC_PROGRAM_PARAM_CHECK    2021 /* nicht mehr verwendet*/
#define DSM_RC_CPIC_UNSUCCESSFUL           2021 /* nicht mehr verwendet*/
#define DSM_RC_UNKNOWN_CPIC_PROBLEM        2021 /* nicht mehr verwendet*/
#define DSM_RC_CPIC_MISSING_LU             2021 /* nicht mehr verwendet*/
#define DSM_RC_CPIC_MISSING_TP             2021 /* nicht mehr verwendet*/
#define DSM_RC_CPIC_SNA6000_LOAD_FAIL      2021 /* nicht mehr verwendet*/
#define DSM_RC_CPIC_STARTUP_FAILURE        2021 /* nicht mehr verwendet*/

/*------------------------------------------------------------------------*/
/* Rückkehrcodes -300 bis -307 sind für die IPX/SPX-Kommunikation reserviert */
/*------------------------------------------------------------------------*/
#define DSM_RC_TLI_ERROR                   2021 /* nicht mehr verwendet*/
#define DSM_RC_IPXSPX_FAILURE              2021 /* nicht mehr verwendet*/
#define DSM_RC_TLI_DLL_MISSING             2021 /* nicht mehr verwendet*/
#define DSM_RC_DLL_LOADFAILURE             2021 /* nicht mehr verwendet*/
#define DSM_RC_DLL_FUNCTION_LOADFAILURE    2021 /* nicht mehr verwendet*/
#define DSM_RC_IPXCONN_REFUSED             2021 /* nicht mehr verwendet*/
#define DSM_RC_IPXCONN_TIMEDOUT            2021 /* nicht mehr verwendet*/
#define DSM_RC_IPXADDR_UNREACHABLE         2021 /* nicht mehr verwendet*/
#define DSM_RC_CPIC_MISSING_DLL            2021 /* nicht mehr verwendet*/
#define DSM_RC_CPIC_DLL_LOADFAILURE        2021 /* nicht mehr verwendet*/
#define DSM_RC_CPIC_FUNC_LOADFAILURE       2021 /* nicht mehr verwendet*/

/*=== Fehlercodes für Shared-Memory-Protokoll ===*/
#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 /* Objektnamenverweis ist NULL     */
#define DSM_RC_NULL_DATABLKPTR    2001 /* dataBlkPtr ist NULL             */
#define DSM_RC_NULL_MSG           2002 /* Nachrichtenparameter in dsmRCMsg
                                          ist NULL                        */

#define DSM_RC_NULL_OBJATTRPTR    2004 /* Objektattributverweis ist NULL  */

#define DSM_RC_NO_SESS_BLK        2006 /* keine Serversitzungsinformationen*/
#define DSM_RC_NO_POLICY_BLK      2007 /* keine Maßnahmenheaderinformat.  */
#define DSM_RC_ZERO_BUFLEN        2008 /* bufferLen für dataBlkPtr ist null*/
#define DSM_RC_NULL_BUFPTR        2009 /* bufferPtr für dataBlkPtr ist NULL*/

#define DSM_RC_INVALID_OBJTYPE    2010 /* ungültiger Objekttyp            */
#define DSM_RC_INVALID_VOTE       2011 /* ungültiges Votum                */
#define DSM_RC_INVALID_ACTION     2012 /* ungültige Aktion                */
#define DSM_RC_INVALID_DS_HANDLE  2014 /* ungültige ADSM-Kennung          */
#define DSM_RC_INVALID_REPOS      2015 /* ungültiger Wert für Repository  */
#define DSM_RC_INVALID_FSNAME     2016 /* Dateibereichsname muss mit
                                          Verzeichnisbegrenzer beginnen   */
```

```
#define DSM_RC_INVALID_OBJNAME     2017 /* ungültiger vollständiger Pfadname*/
#define DSM_RC_INVALID_LLNAME      2018 /* untergeordneter Name muss mit
                                           Verzeichnisbegrenzer beginnen    */
#define DSM_RC_INVALID_OBJOWNER    2019 /* ungültiger Objekteignername      */
#define DSM_RC_INVALID_ACTYPE      2020 /* ungültiger Aktionstyp            */
#define DSM_RC_INVALID_RETCODE     2021 /* dsmRC in dsmRCMsg ist ungültig   */
#define DSM_RC_INVALID_SENDTYPE    2022 /* ungültiger Sendetyp              */
#define DSM_RC_INVALID_PARAMETER   2023 /* ungültiger Parameter             */
#define DSM_RC_INVALID_OBJSTATE    2024 /* aktiv, inaktiv oder beliebige
                                           Übereinstimmung?                 */
#define DSM_RC_INVALID_MCNAME      2025 /* Verwaltungsklassenname nicht gef.*/
#define DSM_RC_INVALID_DRIVE_CHAR  2026 /* Laufwerkbuchstabe ist kein
                                           alphabetisches Zeichen           */
#define DSM_RC_NULL_FSNAME         2027 /* Dateibereichsname ist NULL       */
#define DSM_RC_INVALID_HLNAME      2028 /* untergeordneter Name muss mit
                                           Verzeichnisbegrenzer beginnen    */

#define DSM_RC_NUMOBJ_EXCEED       2029 /* Anzahl Objekte für BeginGetData
                                           überschritten                    */

#define DSM_RC_NEWPW_REQD          2030 /* neues Kennwort ist erforderlich  */
#define DSM_RC_OLDPW_REQD          2031 /* altes Kennwort ist erforderlich  */
#define DSM_RC_NO_OWNER_REQD       2032 /* Eigner nicht zulässig. Standard-
                                           wert zulassen                    */
#define DSM_RC_NO_NODE_REQD        2033 /* Knoten ohne password=generate
                                           nicht zulässig                   */
#define DSM_RC_KEY_MISSING         2034 /* Schlüsseldatei nicht gefunden    */
#define DSM_RC_KEY_BAD             2035 /* Inhalt d. Schlüsseldatei ungültig*/

#define DSM_RC_BAD_CALL_SEQUENCE   2041 /* Folge von DSM-Aufrufen nicht zulässig */
#define DSM_RC_INVALID_TSMBUFFER   2042 /* ungültiger Wert für tsmbuffhandle oder dataPtr */
#define DSM_RC_TOO_MANY_BYTES      2043 /* zu viele Byte in Puffer kopiert  */
#define DSM_RC_MUST_RELEASE_BUFFER 2044 /* Anwendung, die Puffer freigeben muss, kann nicht
                                           verlassen werden */
#define DSM_RC_BUFF_ARRAY_ERROR    2045 /* interner Pufferarrayfehler */
#define DSM_RC_INVALID_DATABLK     2046 /* bei Verwendung von TsmBuffers muss DataBlk null sein */
#define DSM_RC_ENCR_NOT_ALLOWED    2047 /* bei Verwendung von TsmBuffers keine Verschlüsselung
                                           zulässig */
#define DSM_RC_OBJ_COMPRESSED      2048 /* Keine Zurückschreibung möglich bei Verwendung von
                                           TsmBuffers für komprimiertes Objekt */
#define DSM_RC_OBJ_ENCRYPTED       2049 /* Keine Zurückschreibung möglich bei Verwendung von
                                           TsmBuffers für verschlüsseltes Objekt */
#define DSM_RC_WILDCHAR_NOTALLOWED 2050 /* Platzhalter für über-/untergeordneten Namen
                                           nicht zulässig */
#define DSM_RC_POR_NOT_ALLOWED     2051 /* Keine Zurückschreibung von Teilobjekten
                                           mit TsmBuffers möglich */
#define DSM_RC_NO_ENCRYPTION_KEY   2052 /* Verschlüsselungsschlüssel nicht gefunden */
#define DSM_RC_ENCR_CONFLICT       2053 /* sich gegenseitig ausschließende Optionen */

#define DSM_RC_FSNAME_NOTFOUND     2060 /* Dateibereichsname nicht gefunden      */
#define DSM_RC_FS_NOT_REGISTERED   2061 /* Dateibereichsname nicht registriert   */
#define DSM_RC_FS_ALREADY_REGED    2062 /* Dateibereich bereits registriert      */
#define DSM_RC_OBJID_NOTFOUND      2063 /* Keine Objekt-ID zum Zurückschreiben   */
#define DSM_RC_WRONG_VERSION       2064 /* Falsche Codeversion                   */
#define DSM_RC_WRONG_VERSION_PARM  2065 /* Falsche Parameterstrukturversion      */

#define DSM_RC_NEEDTO_ENDTXN       2070 /* Aufruf von dsmEndTxn erforderlich     */

#define DSM_RC_OBJ_EXCLUDED        2080 /* Objekt wird von Verwaltungsklasse
                                           ausgeschlossen                   */
#define DSM_RC_OBJ_NOBCG           2081 /* Objekt hat keine Sich.-Kopiengruppe   */
#define DSM_RC_OBJ_NOACG           2082 /* Objekt hat keine Arch.-Kopiengruppe   */

#define DSM_RC_APISYSTEM_ERROR     2090 /* interner API-Fehler                   */

#define DSM_RC_DESC_TOOLONG             2100 /* Beschreibung ist zu lang         */
#define DSM_RC_OBJINFO_TOOLONG          2101 /* Objektattribut objInfo zu lang   */
#define DSM_RC_HL_TOOLONG               2102 /* übergeordnetes Qualifikations-
                                                merkmal zu lang                 */
#define DSM_RC_PASSWD_TOOLONG           2103 /* Kennwort ist zu lang             */
#define DSM_RC_FILESPACE_TOOLONG        2104 /* Dateibereichsname ist zu lang    */
#define DSM_RC_LL_TOOLONG               2105 /* untergeordnetes Qualifikations-
                                                merkmal zu lang                 */
#define DSM_RC_FSINFO_TOOLONG           2106 /* Dateibereichslänge ist zu groß   */
#define DSM_RC_SENDDATA_WITH_ZERO_SIZE  2107 /* Daten mit Größenschätzung 'null'
                                                senden                          */

/*=== neue Rückkehrcodes für dsmaccess ===*/
#define DSM_RC_INVALID_ACCESS_TYPE 2110 /* ungültiger Zugriffstyp               */
```

```
#define DSM_RC_QUERY_COMM_FAILURE  2111 /* Kommunikationsfehler während Abfrage  */
#define DSM_RC_NO_FILES_BACKUP     2112 /* Keine gesicherten Dateien für diesen
                                           Dateibereich                          */
#define DSM_RC_NO_FILES_ARCHIVE    2113 /* Keine archivierten Dateien für diesen
                                           Dateibereich                          */
#define DSM_RC_INVALID_SETACCESS   2114 /* ungültiges SetAccess-Format           */

/*=== neue Rückkehrcodes für dsmaccess ===*/
#define DSM_RC_STRING_TOO_LONG     2120 /* Zeichenfolgeparameter zu lang         */

#define DSM_RC_MORE_DATA           2200 /* Weitere Daten zum Zurückschreiben vorh.*/

#define DSM_RC_BUFF_TOO_SMALL      2210 /* DataBlk-Puffer für Abfrage zu klein    */

#define DSM_RC_NO_API_CONFIGFILE   2228 /* angegebene API-Konfigurationsdatei
                                           nicht gefunden                        */
#define DSM_RC_NO_INCLEXCL_FILE    2229 /* angegebene Ein-/Ausschlussdatei nicht
                                           gefunden                              */
#define DSM_RC_NO_SYS_OR_INCLEXCL  2230 /* entweder wurde dsm.sys oder die in
                                           dsm.sys angegebene Ein-/Ausschlussdatei
                                           nicht gefunden                        */
#define DSM_RC_REJECT_NO_POR_SUPPORT 2231 /* keine POR-Unterstützung für Server   */

#define DSM_RC_NEED_ROOT           2300 /* Aufrufender der API muss Root sein     */
#define DSM_RC_NEEDTO_CALL_BINDMC  2301 /* dsmBindMC muss zuerst aufgerufen werden*/
#define DSM_RC_CHECK_REASON_CODE   2302 /* Ursachencode aus dsmEndTxn prüfen      */
#define DSM_RC_NEEDTO_ENDTXN_DEDUP_SIZE_EXCEEDED 2303 /* maximale Anzahl Byte bei
                                                         Dedupl. überschritten    */

/*=== Rückkehrcodes 2400-2410 von Lizenzdatei verwendet; siehe agentrc.h ===*/

/*=== Rückkehrcodes 2410-2430 von Orcale-Agent verwendet; siehe agentrc.h ===*/

#define DSM_RC_ENC_WRONG_KEY       4580 /* angegebener Schlüssel ist falsch */
#define DSM_RC_ENC_NOT_AUTHORIZED  4582 /* Benutzer darf nicht entschlüsseln*/
#define DSM_RC_ENC_TYPE_UNKNOWN    4584 /* unbekannter Verschlüsselungstyp  */

/*============================================================================
   Rückkehrcodes 4600-4624 sind für Clustering reserviert
============================================================================*/
#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


/*============================================================================
   Rückkehrcodes 5701-5749 sind für Proxy reserviert
============================================================================*/
#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

/*============================================================================
   Rückkehrcodes 5801-5849 sind für Verschlüsselung/Sicherheit reserviert
============================================================================*/

#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

/*============================================================================
   Rückkehrcodes 6300-6399 sind für clientseitige Deduplizierung reserviert
============================================================================*/
#define DSM_RC_DIGEST_VALIDATION_ERROR                6300  /* Fehler bei Überprüfung der
                                                               End-to-End-Digest-Verarbeitung   */
```

```
#define DSM_RC_DATA_FINGERPRINT_ERROR               6301 /* Fehler bei Rabin-Fingerabdruck    */
#define DSM_RC_DATA_DEDUP_ERROR                     6302 /* Fehler beim Konvertieren von
                                                            Daten in Chunks                   */


#endif /* _H_DSMRC */
```

Zugehörige Verweise:

API-Rückkehrcodes

Anhang B. Quellendateien für API-Typdefinitionen

Dieser Anhang enthält Strukturdefinitionen, Typdefinitionen und Konstanten für die API. Die ersten Headerdateien, `dsmapitd.h` und `tsmapitd.h`, enthalten die Definitionen, die für alle Betriebssysteme einheitlich sind.

Die zweite Headerdatei, `dsmapips.h`, stellt ein Beispiel mit Definitionen bereit, die speziell für ein bestimmtes Betriebssystem gelten, im vorliegenden Beispiel für die Windows-Plattform.

Die dritte Headerdatei, `release.h`, enthält die Versions- und Releaseinformationen.

Die hier bereitgestellten Informationen enthalten eine Zeitpunktkopie der Dateien, die mit der API verteilt werden. Die neueste Version können Sie den Dateien im API-Verteilerpaket entnehmen.

```
/*****************************************************************
* Tivoli Storage Manager                                         *
* API-Clientkomponente                                           *
*                                                                *
* (C) Copyright IBM Corporation 1993, 2010                       *
*****************************************************************/

/*****************************************************************
* Headerdateiname: dsmapitd.h
*
* Umgebung:        ***********************************************
*                  ** Dies ist eine plattformunabhängige         **
*                  ** Quellendatei                               **
*
*                  ***********************************************
*
* Anm. zum Entwurf: Diese Datei enthält Basisdatentypen und Konstanten, die in
*                   allen Clientquellendateien verwendet werden können. Die Konstanten
*                   in dieser Datei müssen für die spezielle Maschine bzw. für das
*                   spezielle Betriebssystem, auf dem die Clientsoftware ausgeführt
*                   werden soll, ordnungsgemäß gesetzt werden.
*
*                   dsmapips.h enthält plattformspezifische Definitionen
*
* Beschreib. Name:  Definitionen für Konstanten der Tivoli Storage
*                   Manager-API
*----------------------------------------------------------------------*/

#ifndef _H_DSMAPITD
#define _H_DSMAPITD


#include "dsmapips.h"      /* Plattformspezif. Definitionen*/
#include "release.h"

/*=== Strukturausrichtung zum Packen der Strukturen definieren ===*/
#if (_OPSYS_TYPE == DS_WINNT) && !defined(_WIN64)
#pragma pack(1)
#endif

#ifdef _MAC
/*=====================================================================
  Auswahlmöglichkeiten:
http://developer.apple.com/documentation/DeveloperTools/Conceptual/PowerPCRuntime/Data/chapter_2_section_3.html

#pragma option align=<mode>
Dabei ist <mode> power, mac68k, natural oder packed.
```

```
=============================================================================*/
#pragma options align=packed
#endif

typedef char osChar_t;

/*<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>*/
/*                     D E F I N I T I O N E N                            */
/*<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>*/
/*-------------------------------------------------------------------------+
|  API-Version, -Release und -Stufe für Verwendung in dsmApiVersion        |
|  in 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

/*-------------------------------------------------------------------------+
| Maximale Feldlängen                                                      |
+-------------------------------------------------------------------------*/
#define DSM_MAX_CG_DEST_LENGTH        30        /* Ziel für Kopiengruppe  */
#define DSM_MAX_CG_NAME_LENGTH        30        /* Kopiengruppenname      */
#define DSM_MAX_DESCR_LENGTH          255       /* Archivierungsbeschr.   */
#define DSM_MAX_DOMAIN_LENGTH         30        /* Richtliniendomänenname */
#define DSM_MAX_FSINFO_LENGTH         500       /* Dateibereichsinformat. */
#define DSM_MAX_USER_FSINFO_LENGTH    480       /* Dateibereichsinformat. zu
                                                   max. Anz. Benutzer     */
#define DSM_MAX_FSNAME_LENGTH         1024      /* Dateibereichsname      */
#define DSM_MAX_FSTYPE_LENGTH         32        /* Dateibereichstyp       */
#define DSM_MAX_HL_LENGTH             1024      /* übergeord. Objektname  */
#define DSM_MAX_ID_LENGTH             64        /* Sitzungsknotenname     */
#define DSM_MAX_LL_LENGTH             256       /* untergeord. Objektname */
#define DSM_MAX_MC_NAME_LENGTH        30        /* Verwaltungsklassenname */
#define DSM_MAX_OBJINFO_LENGTH        255       /* Objektinformationen    */
#define DSM_MAX_EXT_OBJINFO_LENGTH    1500      /* Erweiterte Objektinfo. */
#define DSM_MAX_OWNER_LENGTH          64        /* Objekteignername       */
#define DSM_MAX_PLATFORM_LENGTH       16        /* Anwendungstyp          */
#define DSM_MAX_PS_NAME_LENGTH        30        /* Richtliniengruppename  */
#define DSM_MAX_SERVERTYPE_LENGTH     32        /* Serverplattformtyp     */
#define DSM_MAX_VERIFIER_LENGTH       64        /* Kennwort               */
#define DSM_PATH_MAX                  1024      /* API-Konfig.-Datei: Pfad*/
#define DSM_NAME_MAX                  255       /* API-Konfig.-Datei: Name*/
#define DSM_MAX_NODE_LENGTH           64        /* Knoten-/Maschinenname  */
#define DSM_MAX_RC_MSG_LENGTH         1024      /* Nachr.-PRM f. dsmRCMsg */
#define DSM_MAX_SERVER_ADDRESS        1024      /* Serveradresse          */

#define DSM_MAX_MC_DESCR_LENGTH       DSM_MAX_DESCR_LENGTH /* Verw.-Klasse*/
#define DSM_MAX_SERVERNAME_LENGTH     DSM_MAX_ID_LENGTH /* Servername     */
#define DSM_MAX_GET_OBJ               4080    /* max. Obj. in BeginGetData*/
#define DSM_MAX_PARTIAL_GET_OBJ       1300 /*max. Teilobj. in BeginGetData*/
#define DSM_MAX_COMPRESSTYPE_LENGTH   32       /* max. Komprimierungsalgorithmusname */

/*-------------------------------------------------------------------------+
| Mindestfeldlängen                                                        |
+-------------------------------------------------------------------------*/
#define DSM_MIN_COMPRESS_SIZE  2048 /* Mindestanzahl Byte, die ein Objekt */
                                    /* benötigt, bevor Komprimierung      */
                                    /* zulässig ist                       */

/*-------------------------------------------------------------------------+
|  Werte für mtFlag im Aufruf dsmSetup                                     |
+-------------------------------------------------------------------------*/
#define DSM_MULTITHREAD      bTrue
#define DSM_SINGLETHREAD     bFalse

/*-------------------------------------------------------------------------+
|  Werte für Objekttyp in Struktur dsmObjName                              |
|  Anm.: Diese Werte müssen mit dsmcomm.h synchronisiert werden            |
+-------------------------------------------------------------------------*/
```

```
#define DSM_OBJ_FILE                0x01 /*Obj. hat Attr.-Inf. & -Daten */
#define DSM_OBJ_DIRECTORY           0x02 /*Objekt hat nur Attributinf.  */
#define DSM_OBJ_RESERVED1           0x04 /* für zukünftige Verwendung   */
#define DSM_OBJ_RESERVED2           0x05 /* für zukünftige Verwendung   */
#define DSM_OBJ_RESERVED3           0x06 /* für zukünftige Verwendung   */
#define DSM_OBJ_WILDCARD            0xFE /* beliebiger Objekttyp        */
#define DSM_OBJ_ANY_TYPE            0xFF /* für zukünftige Verwendung   */

/*------------------------------------------------------------------------+
| Typdefinition für compressedState in QryResp                            |
+------------------------------------------------------------------------*/
#define DSM_OBJ_COMPRESSED_UNKNOWN    0
#define DSM_OBJ_COMPRESSED_YES        1
#define DSM_OBJ_COMPRESSED_NO         2

/*------------------------------------------------------------------------+
| Definitionen für Feld "Gruppentyp" in tsmGrouphandlerIn_t               |
+------------------------------------------------------------------------*/

#define  DSM_GROUPTYPE_NONE         0x00   /* Kein Gruppenmitglied    */
#define  DSM_GROUPTYPE_RESERVED1    0x01   /* für zukünftige Verwend. */
#define  DSM_GROUPTYPE_PEER         0x02   /* Peergruppe              */
#define  DSM_GROUPTYPE_RESERVED2    0x03   /* für zukünftige Verwend. */

/*------------------------------------------------------------------------+
| Definitionen für Feld "Mitgliedstyp" in tsmGrouphandlerIn_t             |
+------------------------------------------------------------------------*/

#define  DSM_MEMBERTYPE_LEADER      0x01   /* Gruppenleiter           */
#define  DSM_MEMBERTYPE_MEMBER      0x02   /* Gruppenmitglied         */

/*------------------------------------------------------------------------+
| Definitionen für Feld "Operationstyp" in tsmGrouphandlerIn_t            |
+------------------------------------------------------------------------*/
#define DSM_GROUP_ACTION_BEGIN         0x01
#define DSM_GROUP_ACTION_OPEN          0x02 /* neue Gruppe erstellen  */
#define DSM_GROUP_ACTION_CLOSE         0x03 /* offene Gruppe fest-    */
                                            /* schreiben und speichern*/
#define DSM_GROUP_ACTION_ADD           0x04 /* an Gruppe anhängen     */
#define DSM_GROUP_ACTION_ASSIGNTO      0x05 /* anderer Gruppe zuordnen*/
#define DSM_GROUP_ACTION_REMOVE        0x06 /* Mitgl. aus Gruppe entf.*/

/*------------------------------------------------------------------------+
| Werte für copySer in Strukturen DetailCG für Antwort der                |
| Abfrageverwerwaltungsklasse                                             |
+------------------------------------------------------------------------*/
#define Copy_Serial_Static          1   /*Kopiennummer.:Statisch          */
#define Copy_Serial_Shared_Static   2   /*Kopiennummer.:Gemeinsam statisch*/
#define Copy_Serial_Shared_Dynamic  3   /*Kopiennummer.:Gem. dynamisch    */
#define Copy_Serial_Dynamic         4   /*Kopiennummer.:Dynamisch         */

/*------------------------------------------------------------------------+
| Werte für copyMode in Strukturen DetailCG für Antwort der               |
| Abfrageverwerwaltungsklasse                                             |
+------------------------------------------------------------------------*/
#define Copy_Mode_Modified          1   /* Kopiermodus: Geändert          */
#define Copy_Mode_Absolute          2   /* Kopiermodus: Absolut           */

/*------------------------------------------------------------------------+
|  Werte für objState in Struktur qryBackupData                           |
+------------------------------------------------------------------------*/
#define DSM_ACTIVE           0x01      /* nur aktive Objekte abfragen    */
#define DSM_INACTIVE         0x02      /* nur inaktive Objekte abfragen  */
#define DSM_ANY_MATCH        0xFF      /* alle Sicherungsobjekte abfragen*/

/*------------------------------------------------------------------------+
| Grenzwerte für Feld dsmDate.year in Struktur qryArchiveData             |
+------------------------------------------------------------------------*/
#define DATE_MINUS_INFINITE         0x0000   /* niedrigster Grenzwert     */
#define DATE_PLUS_INFINITE          0xFFFF   /* höchster oberer Grenzwert */
```

```
/*-------------------------------------------------------------------------+
| Bitmasken für Parameter für Aktualisierungsaktion in 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)

/*-------------------------------------------------------------------------+
| Bitmasken für Parameter für Sicherungsaktual.-Aktion in 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)

/*-------------------------------------------------------------------------+
|  Werte für Repositoryparameter in dsmDeleteFS()                          |
+-------------------------------------------------------------------------*/
#define DSM_ARCHIVE_REP       0x0A      /* Archivrepository               */
#define DSM_BACKUP_REP        0x0B      /* Sicherungsrepository           */
#define DSM_REPOS_ALL         0x01      /* alle Respositorytypen          */

/*-------------------------------------------------------------------------+
|  Werte für Votumparameter in dsmEndTxn()                                 |
+-------------------------------------------------------------------------*/
#define DSM_VOTE_COMMIT  1             /* akt. Transaktion festschreiben  */
#define DSM_VOTE_ABORT   2             /* akt. Transakt. rückgängig machen*/

/*-------------------------------------------------------------------------+
|  Werte für verschiedene in Struktur ApiSessInfo zurückgegebene Flags     |
+-------------------------------------------------------------------------*/
/* Codes für Feld für Clientkomprimierung */
#define COMPRESS_YES    1    /* Client muss Daten komprimieren            */
#define COMPRESS_NO     2    /* Client darf Daten NICHT komprimieren      */
#define COMPRESS_CD     3    /* vom Client bestimmt                       */

/* Berechtigungscodes zum Löschen der Archivierung */
#define ARCHDEL_YES     1    /* Löschen der Archivierung zulässig         */
#define ARCHDEL_NO      2    /* Löschen der Archivierung NICHT zulässig   */

/* Berechtigungscode zum Löschen der Sicherung */
#define BACKDEL_YES     1    /* Löschen der Sicherung zulässig            */
#define BACKDEL_NO      2    /* Löschen der Sicherung NICHT zulässig      */


/*-------------------------------------------------------------------------+
  Werte für verschiedene in Struktur optStruct zurückgegebene Flags        |
-------------------------------------------------------------------------*/
#define DSM_PASSWD_GENERATE  1
#define DSM_PASSWD_PROMPT    0

#define DSM_COMM_TCP        1    /* TCP/IP         */
#define DSM_COMM_NAMEDPIPE  2    /* Benannte Pipes */
#define DSM_COMM_SHM        3    /* Shared Memory  */

/* veraltete Übertragungsmethoden */
#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    /* Für IPX/SPX-Unterstützung */
```

```
#define DSM_COMM_NETBIOS    12    /* NETBIOS */
#define DSM_COMM_400COMM    12
#define DSM_COMM_CLIO       12    /* CLIO/S */
/*-------------------------------------------------------------------------+
|  Werte für userNameAuthorities in dsmInitEx für zukünftige Verwendung    |
+-------------------------------------------------------------------------*/
#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)

/*-------------------------------------------------------------------------+
|  Werte für encryptionType in 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)

/*-------------------------------------------------------------------------+
| Definitionen für Feld "Datenträgerklasse" (mediaClass)                   |
+-------------------------------------------------------------------------*/
/*
 *  Die folgenden Konstanten definieren eine Hierarchie von
 *  Datenträgerzugriffsklassen.
 *  Je niedriger die Zahl, desto schneller kann der Zugriff auf Daten
 *  von dem Datenträger bereitgestellt werden.
 */

/* Fixed: Gibt die Klasse der online verfügbaren fest installierten
          Datenträger (wie z. B. Festplatten) an. */
#define  MEDIA_FIXED          0x10

/* Library: Gibt die Klasse mountfähiger Datenträger an, auf die
            über eine mechanische Ladeeinheit zugegriffen werden
            kann. */
#define  MEDIA_LIBRARY        0x20

/* zukünftige Verwendung */
#define  MEDIA_NETWORK        0x30

/* zukünftige Verwendung */
#define  MEDIA_SHELF          0x40

/* zukünftige Verwendung */
#define  MEDIA_OFFSITE        0x50

/* zukünftige Verwendung */
#define  MEDIA_UNAVAILABLE    0xF0


/*-------------------------------------------------------------------------+
| Typdefinition für partielle Objektdaten für dsmBeginGetData()            |
+-------------------------------------------------------------------------*/
typedef struct
{
    dsUint16_t   stVersion;               /* Strukturversion                  */
    dsStruct64_t   partialObjOffset;   /* Abweichung in Objekt für Lesebeginn   */
    dsStruct64_t   partialObjLength;   /* Zu lesende Objektmenge                */
} PartialObjData ;                     /* partielle Objektdaten                 */

#define PartialObjDataVersion 1  /*                                          */

/*-------------------------------------------------------------------------+
|  Typdefinition für Datumsstruktur                                        |
+-------------------------------------------------------------------------*/
typedef struct
```

```
{
   dsUint16_t    year;                 /* Jahr, 16-Bit-Integer (z. B. 1990) */
   dsUint8_t     month;                /* Monat, 8-Bit-Integer (1 - 12)     */
   dsUint8_t     day;                  /* Tag, 8-Bit-Integer (1 - 31)       */
   dsUint8_t     hour;                 /* Stunde, 8-Bit-Integer (0 - 23)    */
   dsUint8_t     minute;               /* Minute, 8-Bit-Integer (0 - 59)    */
   dsUint8_t     second;               /* Sekunde, 8-Bit-Integer (0 - 59)   */
}dsmDate ;

/*------------------------------------------------------------------------+
|  Typdefinition für Objekt-ID in dsmGetObj() und in Struktur dsmGetList   |
+------------------------------------------------------------------------*/
typedef dsStruct64_t  ObjID ;

/*------------------------------------------------------------------------+
| Typdefinition für dsmQueryBuff in dsmBeginQuery()                       |
+------------------------------------------------------------------------*/
typedef void dsmQueryBuff ;

/*------------------------------------------------------------------------+
| Typdefinition für Parameter dsmGetType in dsmBeginGetData()             |
+------------------------------------------------------------------------*/
typedef enum
{
        gtBackup = 0x00,                    /* Sicherungsverarbeitungstyp*/
        gtArchive                           /* Archivier.verarbeitungstyp*/
} dsmGetType ;

/*------------------------------------------------------------------------+
|  Typdefinition für Parameter dsmQueryType in dsmBeginQuery()            |
+------------------------------------------------------------------------*/
typedef enum
{
   qtArchive = 0x00,                /* Archivabfragetyp                  */
   qtBackup,                        /* Sicherungsabfragetyp              */
   qtBackupActive,                  /* Schnellabfr. f. aktive Sich.dateien*/
   qtFilespace,                     /* Dateibereichsabfragetyp           */
   qtMC,                            /* Verwaltungsklassenabfragetyp      */
   qtReserved1,                     /* zukünftige Verwendung             */
   qtReserved2,                     /* zukünftige Verwendung             */
   qtReserved3,                     /* zukünftige Verwendung             */
   qtReserved4,                     /* zukünftige Verwendung             */
   qtBackupGroups,                  /* Gruppenleiter in speziell. Dateisy.*/
   qtOpenGroups,                    /* Offene Gruppen i.speziell.Dateisys.*/
   qtReserved5,                     /* zukünftige Verwendung             */
   qtProxyNodeAuth,                 /* Knoten, an die Knoten weiterl. kann*/
   qtProxyNodePeer,                 /* Peerknoten mit demselben Ziel     */
   qtReserved6,                     /* zukünftige Verwendung             */
   qtReserved7,                     /* zukünftige Verwendung             */
   qtReserved8                      /* zukünftige Verwendung             */
}dsmQueryType ;

/*------------------------------------------------------------------------+
|  Typdefinition für Parameter sendType in dsmBindMC() und dsmSendObj()   |
+------------------------------------------------------------------------*/
typedef enum
{
   stBackup = 0x00,                    /* Sicherungsverarbeitungstyp    */
   stArchive,                          /* Archivierungsverarbeitungstyp */
   stBackupMountWait,           /* Sicherungsverarbeitung; 'Auf Ladevorgang warten' ist aktiviert*/
   stArchiveMountWait           /* Archivierungsverarbeitung; 'Auf Ladevorgang warten' ist aktiviert*/
}dsmSendType ;

/*------------------------------------------------------------------------+
|  Typdefinition für Parameter delType in dsmDeleteObj()                  |
+------------------------------------------------------------------------*/
typedef enum
{
   dtArchive = 0x00,                 /* Typ 'Archivierung löschen' */
   dtBackup,                         /* Typ 'Sicherung löschen (inaktivieren)' */
```

```
   dtBackupID                          /* Typ 'Sicherung löschen (entfernen)' */
}dsmDelType ;

/*-------------------------------------------------------------------------+
|  Typdefinition für Parameter sendType in dsmSetAccess()                  |
+-------------------------------------------------------------------------*/
typedef enum
{
   atBackup = 0x00,                       /* Sicherungsverarbeitungstyp   */
   atArchive                              /* Archivierungsverarbeitungstyp*/
}dsmAccessType;

/*-------------------------------------------------------------------------+
|  Typdefinition für API-Version in dsmInit() und dsmQueryApiVersion()     |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t version;       /* API-Version                     */
   dsUint16_t release;       /* API-Release                     */
   dsUint16_t level;         /* API-Stand                       */
}dsmApiVersion;

/*-------------------------------------------------------------------------+
|  Typdefinition für API-Version in dsmInit() und dsmQueryApiVersion()     |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t     stVersion;             /* Strukturversion             */
   dsUint16_t version;       /* API-Version                     */
   dsUint16_t release;       /* API-Release                     */
   dsUint16_t level;         /* API-Stand                       */
   dsUint16_t subLevel;      /* API-Unterstufe                  */
   dsmBool_t  unicode;       /* API-Unicode? */
}dsmApiVersionEx;


#define apiVersionExVer       2

/*-------------------------------------------------------------------------+
|  Typdefinition für Anwendungsversion in dsmInit()                        |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t   stVersion;               /* Strukturversion                  */
   dsUint16_t   applicationVersion;      /* Anwendungsversionsnummer         */
   dsUint16_t   applicationRelease;      /* Anwendungsreleasenummer          */
   dsUint16_t   applicationLevel;        /* Anwendungsebenennummer           */
   dsUint16_t   applicationSubLevel;     /* Anwendungsunterstufennummer      */
} dsmAppVersion;

#define appVersionVer     1

/*-------------------------------------------------------------------------+
|  Typdefinition für in BindMC, Send, Delete, Query verwendeten Objektnamen|
+-------------------------------------------------------------------------*/
typedef struct S_dsmObjName
{
   char       fs[DSM_MAX_FSNAME_LENGTH + 1] ;             /* Dateibereichsname */
   char       hl[DSM_MAX_HL_LENGTH + 1] ;                 /* Übergeordneter Name */
   char       ll[DSM_MAX_LL_LENGTH + 1] ;                 /* Untergeordneter Name */
   dsUint8_t  objType;              /* Definitionen für Objekttypwerte siehe oben */
}dsmObjName;


/*-------------------------------------------------------------------------+
|  Typdefinition f. Informationen zu 'Sicherung löschen' in dsmDeleteObj() |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t         stVersion;              /* Strukturversion               */
```

```
   dsmObjName       *objNameP ;                    /* Objektname            */
   dsUint32_t       copyGroup ;                    /* Kopiengruppe          */
}delBack ;

#define  delBackVersion    1

/*--------------------------------------------------------------------------+
|  Typdefinition für Infos zu 'Archivierung löschen' in dsmDeleteObj()      |
+--------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsStruct64_t     objId ;                       /* Objekt-ID             */
}delArch ;

#define  delArchVersion  1

/*--------------------------------------------------------------------------+
|  Typdefinition für Infos zu 'Sicherungs-ID löschen' in dsmDeleteObj()     |
+--------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsStruct64_t     objId ;                       /* Objekt-ID             */
}delBackID;

#define  delBackIDVersion  1

/*--------------------------------------------------------------------------+
|  Typdefinition für Löschinformationen in dsmDeleteObj()                   |
+--------------------------------------------------------------------------*/
typedef union
{
   delBack   backInfo ;
   delArch   archInfo ;
   delBackID backIDInfo ;
}dsmDelInfo ;

/*--------------------------------------------------------------------------+
|  Typdefinition für Parameter Object Attribute in dsmSendObj()             |
+--------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t   stVersion;                      /* Strukturversion   */
   char         owner[DSM_MAX_OWNER_LENGTH + 1]; /* Objekteigner      */
   dsStruct64_t sizeEstimate;                   /* Größenschätzung des Objekts in Byte */
   dsmBool_t    objCompressed;                  /* Ist Objekt bereits komprimiert? */
   dsUint16_t   objInfoLength;                  /* Länge der objektabhängigen Information */
   char         *objInfo;                       /* objektabhängige Information */
   char         *mcNameP;                       /* Verwaltungsklassenname für Überschreibung */
   dsmBool_t    disableDeduplication;     /* 'Keine Deduplizierung' für dieses Objekt erzwingen */
   dsmBool_t    useExtObjInfo;                  /* Erw. Objektinfo. bis zu 1536 verwenden */
}ObjAttr;

#define ObjAttrVersion  4



/*--------------------------------------------------------------------------+
| Typdefinition für zurückgegebenen mcBindKey in dsmBindMC()                |
+--------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   char        mcName[DSM_MAX_MC_NAME_LENGTH + 1];
                                     /* Name der an das Objekt gebundenen Verwaltungsklasse. */
   dsmBool_t   backup_cg_exists;                    /* Wahr/Falsch */
   dsmBool_t   archive_cg_exists;                   /* Wahr/Falsch */
   char        backup_copy_dest[DSM_MAX_CG_DEST_LENGTH + 1];
                                      /* Zielname für Sicherungskopie */
```

```
   char        archive_copy_dest[DSM_MAX_CG_DEST_LENGTH + 1];
                                                /* Zielname für Archivierungskopie */
}mcBindKey;

#define mcBindKeyVersion  1


/*--------------------------------------------------------------------------+
| Typdefinition für Objektliste in dsmBeginGetData()                        |
+--------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsUint32_t       numObjId ;          /* Anzahl Objekt-IDs in Liste   */
   ObjID           *objId ;             /* Liste der zurückzuschreibenden Objekt-IDs */
   PartialObjData  *partialObjData;     /* Liste partieller Objektdateninformationen */
}dsmGetList ;

#define dsmGetListVersion     2  /* Standard, bei Nichtverwendung partieller Objektdaten */
#define dsmGetListPORVersion  3  /* Version, bei Verwendung partieller Objektdaten */

/*--------------------------------------------------------------------------+
|  Typdefinition für DataBlk zum Abrufen oder Senden von Daten              |
+--------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t      stVersion;           /* Strukturversion     */
   dsUint32_t bufferLen;                /* Länge des unten übergebenen Puffers */
   dsUint32_t numBytes;                 /* Anzahl der aktuell aus dem Puffer gelesenen */
                                        /* oder in den Puffer geschriebenen Byte */
   char       *bufferPtr;               /* Datenpuffer */
   dsUint32_t numBytesCompressed;       /* beim Senden tatsächlich komprimierte Byte */
   dsUint16_t reserved;                 /* für zukünftige Verwendung      */
}DataBlk;

#define DataBlkVersion  3


/*--------------------------------------------------------------------------+
|  Typdefinition für queryBuffer für Verwaltungsklasse in dsmBeginQuery()   |
+--------------------------------------------------------------------------*/
typedef struct S_qryMCData
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   char         *mcName;                        /* Verwaltungsklassenname */
                     /* einzelner Name für 1 Klasse oder leere Zeichenfolge, um alle abzurufen */
   dsmBool_t    mcDetail;                           /* Details erwünscht oder nicht? */
}qryMCData;

#define qryMCDataVersion  1

/*=== Werte für RETINIT ===*/
#define ARCH_RETINIT_CREATE 0
#define ARCH_RETINIT_EVENT  1

/*--------------------------------------------------------------------------+
|  Typdefinition f. Archivierungskopiengruppendetails in Antwort auf Verwaltungsklassenabfrage|
+--------------------------------------------------------------------------*/
typedef struct S_archDetailCG
{
   char         cgName[DSM_MAX_CG_NAME_LENGTH + 1];       /* Kopiengruppenname */
   dsUint16_t   frequency;                        /* Häufigkeit von Kopien (Archivierungen) */
   dsUint16_t   retainVers;                              /* Version aufbewahren */
   dsUint8_t    copySer;       /* Hinweise zur Kopiennummerierung siehe Definitionen */
   dsUint8_t    copyMode;         /* Hinweise zu Kopiermoduswerten siehe Definitionen oben */
   char         destName[DSM_MAX_CG_DEST_LENGTH + 1];      /* Zielname für Kopie */
   dsmBool_t    bLanFreeDest;           /* Ziel hat LAN-unabhängigen Pfad?    */
   dsmBool_t    reserved;               /* Derzeit nicht verwendet          */
   dsUint8_t    retainInit;             /* Mögliche Werte siehe oben        */
   dsUint16_t   retainMin;              /* wenn retInit auf EVENT gesetzt: Anzahl Tage */
```

```
   dsmBool_t    bDeduplicate;            /* Am Ziel ist Deduplizierung aktiviert    */
}archDetailCG;

/*-------------------------------------------------------------------------+
|  Typdefinition f. Sicherungskopiengruppendetails in Antwort auf Verwaltungsklassenabfrage   |
+-------------------------------------------------------------------------*/
typedef struct S_backupDetailCG
{
   char         cgName[DSM_MAX_CG_NAME_LENGTH + 1];       /* Kopiengruppenname */
   dsUint16_t   frequency;                              /* Häufigkeit der Sicherung */
   dsUint16_t   verDataExst;                           /* Versionen bestehender Daten */
   dsUint16_t   verDataDltd;                          /* Versionen gelöschter Daten */
   dsUint16_t   retXtraVers;                          /* Extraversionen aufbewahren */
   dsUint16_t   retOnlyVers;                          /* Einzige Version aufbewahren */
   dsUint8_t    copySer;       /* Hinweise zur Kopiennummerierung siehe Definitionen */
   dsUint8_t    copyMode;         /* Hinweise zu Kopiermoduswerten siehe Definitionen oben */
   char         destName[DSM_MAX_CG_DEST_LENGTH + 1];      /* Zielname für Kopie */
   dsmBool_t    bLanFreeDest;            /* Ziel hat LAN-unabhängigen Pfad?    */
   dsmBool_t    reserved;                /* Derzeit nicht verwendet          */
   dsmBool_t    bDeduplicate;            /* Am Ziel ist Deduplizierung aktiviert    */
}backupDetailCG;


/*-------------------------------------------------------------------------+
|  Typdefinition für Detailantwort zu Verwaltungsklassenabfrage in dsmGetNextQObj()           |
+-------------------------------------------------------------------------*/
typedef struct S_qryRespMCDetailData
{
   dsUint16_t         stVersion;               /* Strukturversion                */
   char            mcName[DSM_MAX_MC_NAME_LENGTH + 1];        /* Verwaltungsklassenname */
   char            mcDesc[DSM_MAX_MC_DESCR_LENGTH + 1]; /* Verwaltungsklassenbeschreibung */
   archDetailCG    archDet;                /* Archivierungskopiengruppendetail */
   backupDetailCG  backupDet;               /* Sicherungskopiengruppendetail */
}qryRespMCDetailData;

#define qryRespMCDetailDataVersion  4

/*-------------------------------------------------------------------------+
| Typdefinition f. Antwort mit Zusammenfass. zur Verwaltungsklassenabfrage in dsmGetNextQObj()|
+-------------------------------------------------------------------------*/
typedef struct S_qryRespMCData
{
   dsUint16_t         stVersion;               /* Strukturversion                */
   char            mcName[DSM_MAX_MC_NAME_LENGTH + 1];        /* Verwaltungsklassenname */
   char            mcDesc[DSM_MAX_MC_DESCR_LENGTH + 1]; /* Verwaltungsklassenbeschreibung */
}qryRespMCData;

#define qryRespMCDataVersion  1



/*-------------------------------------------------------------------------+
|  Typdefinition für queryBuffer für Archivierung in dsmBeginQuery()       |
+-------------------------------------------------------------------------*/
typedef struct S_qryArchiveData
{
   dsUint16_t         stVersion;               /* Strukturversion                */
   dsmObjName   *objName;                     /* Vollständiger DSM-Name des Objekts */
   char         *owner;                                  /* Eignername */
                 /* Hinweise zu maximalen Datumsgrenzen siehe Definitionen oben */
   dsmDate      insDateLowerBound;      /* unterer Grenzwert für Archivierungseinfügedatum */
   dsmDate      insDateUpperBound;      /* oberer Grenzwert für Archivierungseinfügedatum */
   dsmDate      expDateLowerBound;      /* unterer Grenzwert für Verfallsdatum */
   dsmDate      expDateUpperBound;      /* oberer Grenzwert für Verfallsdatum */
   char         *descr;                /* Archivierungsbeschreibung */
} qryArchiveData;

#define qryArchiveDataVersion  1
```

```
/*=== Werte für Feld retentionInitiated ===*/
#define DSM_ARCH_RETINIT_UNKNOWN 0 /* Aufbewahrungseinleitung ist unbekannt (Server mit älterer Version) */
#define DSM_ARCH_RETINIT_STARTED 1 /* Uhr für Aufbewahrungsdauer wird gestartet */
#define DSM_ARCH_RETINIT_PENDING 2 /* Uhr für Aufbewahrungsdauer wird nicht gestartet */


/*=== Werte für objHeld ===*/
#define DSM_ARCH_HELD_UNKNOWN 0    /* unbekannter Haltestatus (Server mit älterer Version) */
#define DSM_ARCH_HELD_FALSE   1    /* Objekt ist NICHT in einem Rückstellungsstatus für 'Löschen' */
#define DSM_ARCH_HELD_TRUE    2    /* Objekt ist in einem Rückstellungsstatus für 'Löschen' */

/*--------------------------------------------------------------------------+
| Typdefinition für Antwort auf Archivierungsabfrage in dsmGetNextQObj()   |
+--------------------------------------------------------------------------*/
typedef struct S_qryRespArchiveData
{
   dsUint16_t         stVersion;               /* Strukturversion               */
   dsmObjName      objName;                            /* Qualifikationsmerkmal für Dateibereichsname */
   dsUint32_t      copyGroup;                          /* Kopiengruppenzahl */
   char            mcName[DSM_MAX_MC_NAME_LENGTH + 1];        /* Verwaltungsklassenname */
   char        owner[DSM_MAX_OWNER_LENGTH + 1];            /* Eignername */
   dsStruct64_t    objId;                              /* Eindeutige Kopien-ID */
   dsStruct64_t    reserved;                        /* Abwärtskompatibilität */
   dsUint8_t       mediaClass;                         /* Datenträgerzugriffsklasse */
   dsmDate         insDate;                            /* Archivierungseinfügedatum */
   dsmDate         expDate;                         /* Verfallsdatum für Objekt */
   char            descr[DSM_MAX_DESCR_LENGTH + 1];     /* Archivierungsbeschreibung */
   dsUint16_t  objInfolen;                          /* Länge der objektabhängigen Informationen */
   char            reservedObjInfo[DSM_MAX_OBJINFO_LENGTH];  /* objektabhängige Informationen */
   dsUint160_t     restoreOrderExt;                    /* Zurückschreibungsreihenfolge */
   dsStruct64_t    sizeEstimate;              /* vom Benutzer gespeicherte Größenschätzung */
   dsUint8_t       compressType;                         /* Komprimierungsflag */
   dsUint8_t       retentionInitiated;                 /* Objekt wartet bei Aufbewahrungsereignis */
   dsUint8_t       objHeld;  /* Objekt befindet sich im Aufbewahrungshaltestatus, siehe Werte oben */
   dsUint8_t       encryptionType;          /* Verschlüsselungstyp */
   dsmBool_t       clientDeduplicated;      /* Objekt wird von API dedupliziert */
   char            objInfo[DSM_MAX_EXT_OBJINFO_LENGTH];  /* objektabhängige Informationen */
   char            compressAlg[DSM_MAX_COMPRESSTYPE_LENGTH + 1];  /* Komprimierungsalgorithmusname */
}qryRespArchiveData;

#define qryRespArchiveDataVersion  7

/*--------------------------------------------------------------------------+
|  Typdefinition für Archivierungsparameter sendBuff in dsmSendObj()       |
+--------------------------------------------------------------------------*/
typedef struct S_sndArchiveData
{
   dsUint16_t         stVersion;               /* Strukturversion               */
   char         *descr;                       /* Archivierungsbeschreibung */
}sndArchiveData;

#define sndArchiveDataVersion  1

/*--------------------------------------------------------------------------+
|  Typdefinition für queryBuffer für Sicherung in dsmBeginQuery()          |
+--------------------------------------------------------------------------*/
typedef struct S_qryBackupData
{
   dsUint16_t         stVersion;               /* Strukturversion               */
   dsmObjName  *objName;          /* Vollständiger DSM-Name des Objekts */
   char        *owner;            /* Eignername */
   dsUint8_t   objState;          /* Objektstatusselektor */
   dsmDate     pitDate;           /* Datumswert für zeitpunktgesteuerte Zurückschreibung */
                                  /* Hinweise zu möglichen Werten siehe Definitionen oben */
}qryBackupData;

#define qryBackupDataVersion  2

typedef struct
```

```
{
  dsUint8_t    reserved1;
  dsStruct64_t reserved2;
} reservedInfo_t;                /* für zukünftige Verwendung */

/*--------------------------------------------------------------------------+
| Typdefinition für Antwort auf Sicherungsabfrage in dsmGetNextQObj()       |
+--------------------------------------------------------------------------*/
typedef struct S_qryRespBackupData
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsmObjName      objName;                      /* Vollständiger DSM-Name des Objekts */
   dsUint32_t      copyGroup;                         /* Kopiengruppenzahl */
   char            mcName[DSM_MAX_MC_NAME_LENGTH + 1];        /* Verwaltungsklassenname */
   char        owner[DSM_MAX_OWNER_LENGTH + 1];            /* Eignername */
   dsStruct64_t    objId;                             /* Eindeutige Objekt-ID */
   dsStruct64_t    reserved;                       /* Abwärtskompatibilität */
   dsUint8_t       mediaClass;                        /* Datenträgerzugriffsklasse */
   dsUint8_t       objState;                           /* Objektstatus, aktiv usw. */
   dsmDate     insDate;                           /* Sicherungseinfügedatum */
   dsmDate         expDate;                     /* Verfallsdatum für Objekt */
   dsUint16_t  objInfolen;                          /* Länge der objektabhängigen Informationen */
   char            reservedObjInfo[DSM_MAX_OBJINFO_LENGTH];  /* objektabhängige Informationen */
   dsUint160_t     restoreOrderExt;                   /* Zurückschreibungsreihenfolge */
   dsStruct64_t    sizeEstimate;                      /* vom Benutzer gespeicherte Größenschätzung */
   dsStruct64_t    baseObjId;
   dsUint16_t      baseObjInfolen;                    /* Länge der basisobjektabhängigen Informationen */
   dsUint8_t       baseObjInfo[DSM_MAX_OBJINFO_LENGTH]; /* basisobjektabhängige Informationen */
   dsUint160_t     baseRestoreOrder;                  /* Zurückschreibungsreihenfolge */
   dsUint32_t      fsID;
   dsUint8_t       compressType;
   dsmBool_t       isGroupLeader;
   dsmBool_t       isOpenGroup;
   dsUint8_t       reserved1;               /* für zukünftige Verwendung */
   dsmBool_t       reserved2;               /* für zukünftige Verwendung */
   dsUint16_t      reserved3;               /* für zukünftige Verwendung */
   reservedInfo_t  *reserved4;              /* für zukünftige Verwendung */
   dsUint8_t       encryptionType;          /* Verschlüsselungstyp */
   dsmBool_t       clientDeduplicated;      /* Objekt wird von API dedupliziert */
   char            objInfo[DSM_MAX_EXT_OBJINFO_LENGTH];  /* objektabhängige Informationen */
   char            compressAlg[DSM_MAX_COMPRESSTYPE_LENGTH + 1];  /* Komprimierungsalgorithmusname */
}qryRespBackupData;

#define qryRespBackupDataVersion  8

/*--------------------------------------------------------------------------+
|  Typdefinition für queryBuffer für aktive Sicherung in dsmBeginQuery()   |
|                                                                          |
|  Hinweise: Für die Abfrage der aktiven Sicherung müssen nur die Felder   |
|            fs (Dateibereich) und objType gesetzt werden. objType kann nur|
|            auf DSM_OBJ_FILE oder DSM_OBJ_DIRECTORY gesetzt werden.       |
|            Für DSM_OBJ_ANY_TYPE wird keine Übereinstimmung in der Abfrage|
|            gefunden.                                                     |
+--------------------------------------------------------------------------*/
typedef struct S_qryABackupData
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsmObjName      *objName;                  /* Nur fs und objType werden verwendet */
}qryABackupData;

#define qryABackupDataVersion  1

/*--------------------------------------------------------------------------+
| Typdefinition für Antwort auf Abfrage der aktiven Sicherung in dsmGetNextQObj()            |
+--------------------------------------------------------------------------*/
typedef struct S_qryARespBackupData
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsmObjName  objName;                      /* Vollständiger DSM-Name des Objekts */
   dsUint32_t  copyGroup;                         /* Kopiengruppenzahl */
```

```
   char        mcName[DSM_MAX_MC_NAME_LENGTH + 1];/* Verwaltungsklassenname */
   char        owner[DSM_MAX_OWNER_LENGTH + 1];            /* Eignername */
   dsmDate     insDate;                            /* Sicherungseinfügedatum */
   dsUint16_t  objInfolen;                         /* Länge der objektabhängigen Informationen */
   char            reservedObjInfo[DSM_MAX_OBJINFO_LENGTH];  /* objektabhängige Informationen */
   char            objInfo[DSM_MAX_EXT_OBJINFO_LENGTH];  /* objektabhängige Informationen */
}qryARespBackupData;

#define qryARespBackupDataVersion  2

/*--------------------------------------------------------------------------+
|  Typdefinition für queryBuffer für Sicherung in dsmBeginQuery()           |
+--------------------------------------------------------------------------*/
typedef struct qryBackupGroups
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   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


/*--------------------------------------------------------------------------+
|  Typdefinition für queryBuffer für Proxy-Knoten in dsmBeginQuery()        |
+--------------------------------------------------------------------------*/
typedef struct qryProxyNodeData
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   char        *targetNodeName;            /* Zielknotenname  */
}qryProxyNodeData;

#define qryProxyNodeDataVersion  1

/*--------------------------------------------------------------------------+
| Typdefinition für den in dsmGetNextQObj() verwendeten Parameter qryRespProxyNodeData        |
+--------------------------------------------------------------------------*/

typedef struct
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   char            targetNodeName[DSM_MAX_ID_LENGTH+1]; /* Zielknotenname  */
   char            peerNodeName[DSM_MAX_ID_LENGTH+1];   /* Peerknotenname  */
   char            hlAddress[DSM_MAX_ID_LENGTH+1];      /* übergeordnete Peeradresse */
   char            llAddress[DSM_MAX_ID_LENGTH+1];      /* untergeordnete Peeradresse */
}qryRespProxyNodeData;

#define qryRespProxyNodeDataVersion  1


/*--------------------------------------------------------------------------+
|  Typdefinition für WINNT- und OS/2-Dateibereichsattribute                 |
+--------------------------------------------------------------------------*/
typedef struct
{
   char         driveLetter ;          /* Laufwerkbuchstabe für Dateibereich */
   dsUint16_t   fsInfoLength;          /* für Dateibereichsinformationen verwendete Länge */
   char         fsInfo[DSM_MAX_FSINFO_LENGTH];/* vom Aufrufenden bestimmte Daten */
}dsmDosFSAttrib ;

/*--------------------------------------------------------------------------+
|  Typdefinition für UNIX-Dateibereichsattribute                            |
+--------------------------------------------------------------------------*/
typedef struct
```

```
{
   dsUint16_t   fsInfoLength;          /* für Dateibereichsinformationen verwendete Länge */
   char         fsInfo[DSM_MAX_FSINFO_LENGTH];/* vom Aufrufenden bestimmte Daten */
}dsmUnixFSAttrib ;

/*--------------------------------------------------------------------------+
|  Typdefinition für NetWare-Dateibereichsattribute                         |
+--------------------------------------------------------------------------*/
typedef dsmUnixFSAttrib dsmNetwareFSAttrib;

/*--------------------------------------------------------------------------+
|  Typdefinition für Dateibereichsattribute in allen Dateibereichsaufrufen |
+--------------------------------------------------------------------------*/
typedef union
{
   dsmNetwareFSAttrib  netwareFSAttr;
   dsmUnixFSAttrib     unixFSAttr ;
   dsmDosFSAttrib      dosFSAttr ;
}dsmFSAttr ;

/*--------------------------------------------------------------------------+
|  Typdefinition für Parameter fsUpd in dsmUpdateFS()                       |
+--------------------------------------------------------------------------*/
typedef struct S_dsmFSUpd
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   char           *fsType ;              /* Dateibereichstyp              */
   dsStruct64_t    occupancy ;            /* Belegungsschätzung            */
   dsStruct64_t    capacity  ;            /* Kapazitätsschätzung           */
   dsmFSAttr       fsAttr ;               /* plattformspezifische Attribute */
}dsmFSUpd ;

#define dsmFSUpdVersion  1

/*--------------------------------------------------------------------------+
|  Typdefinition für queryBuffer für Dateibereich in dsmBeginQuery()        |
+--------------------------------------------------------------------------*/
typedef struct S_qryFSData
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   char        *fsName;                  /* Dateibereichsname */
}qryFSData;

#define qryFSDataVersion  1

/*--------------------------------------------------------------------------+
| Typdefinition für Antwort auf Dateibereichsabfrage in dsmGetNextQObj()    |
+--------------------------------------------------------------------------*/
typedef struct S_qryRespFSData
{
   dsUint16_t         stVersion;             /* Strukturversion               */
   char           fsName[DSM_MAX_FSNAME_LENGTH + 1];         /* Dateibereichsname */
   char           fsType[DSM_MAX_FSTYPE_LENGTH + 1] ;        /* Dateibereichstyp  */
   dsStruct64_t   occupancy;                      /* Belegungsschätzung in Byte */
   dsStruct64_t   capacity;                       /* Kapazitätsschätzung in Byte */
   dsmFSAttr      fsAttr ;                       /* plattformspezifische Attribute */
   dsmDate        backStartDate;                 /* Datum für Sicherungsbeginn */
   dsmDate        backCompleteDate;              /* Datum für Sicherungsende   */
   dsmDate        reserved1;                     /* Für zukünftige Verwendung  */
   dsmDate        lastReplStartDate;        /* Startzeitpunkt der letzten Replikation     */
   dsmDate        lastReplCmpltDate;        /* Endzeitpunkt der letzten Replikation       */
                                            /*   (eventuell trat ein Fehler auf,          */
                                            /*    aber sie wurde dennoch beendet)         */
   dsmDate        lastBackOpDateFromServer; /* Die letzte Speicherungszeitmarke, die der  */
                                            /* Client auf dem Server gespeichert hat      */
   dsmDate        lastArchOpDateFromServer; /* Die letzte Speicherungszeitmarke, die der  */
                                            /* Client auf dem Server gespeichert hat      */
   dsmDate        lastSpMgOpDateFromServer; /* Die letzte Speicherungszeitmarke, die der  */
                                            /* Client auf dem Server gespeichert hat      */
   dsmDate        lastBackOpDateFromLocal;  /* Die letzte Speicherungszeitmarke, die der  */
```

```
                                        /*    Client lokal gespeichert hat           */
   dsmDate        lastArchOpDateFromLocal;  /* Die letzte Speicherungszeitmarke, die der  */
                                        /*    Client lokal gespeichert hat           */
   dsmDate        lastSpMgOpDateFromLocal;  /* Die letzte Speicherungszeitmarke, die der  */
                                        /*    Client lokal gespeichert hat           */
   dsInt32_t      failOverWriteDelay;       /* Minuten, die der Client vor dem Speichern  */
                                        /* auf diesem Rpl.server warten muss; Sonder- */
                                        /* codes: NO_ACCESS(-1), ACCESS_RDONLY (-2)   */
}qryRespFSData;

#define qryRespFSDataVersion 4

/*--------------------------------------------------------------------------+
|  Typdefinition für Parameter regFilespace in dsmRegisterFS()              |
+--------------------------------------------------------------------------*/
typedef struct S_regFSData
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   char            *fsName;                    /* Dateibereichsname */
   char            *fsType;                    /* Dateibereichstype */
   dsStruct64_t    occupancy;                  /* Belegungsschätzung in Byte */
   dsStruct64_t   capacity;                     /* Kapazitätsschätzung in Byte */
   dsmFSAttr       fsAttr ;                    /* plattformspezifische Attribute */
}regFSData;

#define regFSDataVersion  1

/*--------------------------------------------------------------------------+
| Typdefinition für den in apisessInfo verwendeten dedupType                |
+--------------------------------------------------------------------------*/
typedef enum
{
   dedupServerOnly= 0x00,             /* Deduplizierung erfolgt nur auf dem Server */
   dedupClientOrServer                /* Deduplizierung kann auf dem Client oder Server erfolgen */
}dsmDedupType ;

/*--------------------------------------------------------------------------+
 | Typdefinition für Übernahmekonfiguration und -status
 --------------------------------------------------------------------------*/
typedef enum
{
   failOvrNotConfigured = 0x00,
   failOvrConfigured,
   failOvrConnectedToReplServer
}dsmFailOvrCfgType ;

/*--------------------------------------------------------------------------+
|  Typdefinition für Antwort mit Sitzungsdaten in dsmQuerySessionInfo()     |
+--------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t   stVersion;               /* Strukturversion                   */
      /*-----------------------------------------------------------------*/
      /*           Serverinformationen                                   */
      /*-----------------------------------------------------------------*/
   char           serverHost[DSM_MAX_SERVERNAME_LENGTH+1];
                                         /* Netzhostname des DSM-Servers */
   dsUint16_t     serverPort;            /* Serverkommunikationsport auf Host */
   dsmDate        serverDate;            /* Datum/Zeit des Servers */
   char           serverType[DSM_MAX_SERVERTYPE_LENGTH+1];
                                         /* Ausführungsplattform des Servers */
   dsUint16_t     serverVer;             /* Versionsnummer des Servers       */
   dsUint16_t     serverRel;             /* Releasenummer des Servers        */
   dsUint16_t     serverLev;             /* Stufennummer des Servers         */
   dsUint16_t         serverSubLev;  /* Unterstufennummer des Servers    */
      /*-----------------------------------------------------------------*/
      /*           Clientstandardwerte                                    */
      /*-----------------------------------------------------------------*/
   char           nodeType[DSM_MAX_PLATFORM_LENGTH+1]; /* Knoten-/Anwendungstyp */
   char           fsdelim;               /* Dateibereichsbegrenzer             */
```

```
   char           hldelim;               /* Begrenzer zwischen oberer u. unterer Ebene */
   dsUint8_t      compression;           /* Komprimierungsflag */
   dsUint8_t      archDel;               /* Berechtigung zum Löschen des Archivs */
   dsUint8_t      backDel;               /* Berechtigung zum Löschen der Sicherung */
   dsUint32_t     maxBytesPerTxn;        /* für zukünftige Verwendung */
   dsUint16_t     maxObjPerTxn;          /* Max. Anzahl Objekte in einer Transaktion */
      /*--------------------------------------------------------------------*/
      /*          Sitzungsdaten                                             */
      /*--------------------------------------------------------------------*/
   char       id[DSM_MAX_ID_LENGTH+1];    /* Knotenname der Anmelde-ID    */
   char       owner[DSM_MAX_OWNER_LENGTH+1]; /* Anmeldeeigner             */
                                 /*   (für Mehrbenutzerplattformen   */
   char       confFile[DSM_PATH_MAX + DSM_NAME_MAX +1];
                                /* Länge ist plattformabhängig       */
                                /* dsInit-Name der Anwend.-Konfig.-Datei */
   dsUint8_t  opNoTrace;             /* Option dsInit - NoTrace = 1      */
      /*--------------------------------------------------------------------*/
      /*          Maßnahmendaten                                            */
      /*--------------------------------------------------------------------*/
   char           domainName[DSM_MAX_DOMAIN_LENGTH+1]; /* Domänenname         */
   char           policySetName[DSM_MAX_PS_NAME_LENGTH+1];
                                 /* Name der aktiven Maßnahmengruppe */
   dsmDate        polActDate;         /* Aktivierungsdatum der Maßnahmengruppe */
   char           dfltMCName[DSM_MAX_MC_NAME_LENGTH+1];/* Standardverwaltungsklasse */
   dsUint16_t     gpBackRetn;             /* Karenzzeit f. Aufbewahrungszeitraum f. Sicherung */
   dsUint16_t     gpArchRetn;             /* Karenzzeit f. Aufbewahrungszeitraum f. Archivierung */
   char           adsmServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* ADSM-Servername */
   dsmBool_t      archiveRetentionProtection; /* Aufbewahrungsschutz durch Server ist aktiviert*/
   dsStruct64_t   maxBytesPerTxn_64;         /* für zukünftige Verwendung      */
   dsmBool_t      lanFreeEnabled;            /* Option 'LAN-unabhängig' ist gesetzt */
   dsmDedupType   dedupType;                 /* server oder clientOrServer       */
   char           accessNode[DSM_MAX_ID_LENGTH+1];    /* als Knotenname          */

   /*--------------------------------------------------------------------*/
   /*          Replikations- und Übernahmeinformationen                  */
   /*--------------------------------------------------------------------*/
   dsmFailOvrCfgType failOverCfgType; /* Status der Übernahme */
   char           replServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* Replikationsservername            */
   char           homeServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* Home-Servername                   */
   char           replServerHost[DSM_MAX_SERVERNAME_LENGTH+1]; /* Netzhostname des DSM-Servers      */
   dsInt32_t      replServerPort;                              /* Serverkommunikationsport auf Host */

}ApiSessInfo;

#define ApiSessInfoVersion  6



/*--------------------------------------------------------------------------+
| Typdefinition für Antwort auf Abfrageoptionen in dsmQueryCliOptions()     |
|      und 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 ;

/*--------------------------------------------------------------------------+
| Typdefinition für in logInfo verwendeten LogType                          |
+--------------------------------------------------------------------------*/
typedef enum
```

```
{
   logServer = 0x00,            /* Protokollnachricht nur an Server */
   logLocal,                    /* Protokollnachricht nur an lokales Fehlerprotokoll */
   logBoth,                     /* Protokollnachricht an Server und lokales Fehlerprotokoll */
   logNone
}dsmLogType ;

/*------------------------------------------------------------------------+
| Typdefinition für in dsmLogEvent() verwendeten Parameter logInfo         |
+------------------------------------------------------------------------*/

typedef struct
{
   char        *message;   /* Text der zu protokollierenden Nachricht */
   dsmLogType  logType;    /* Protokolltyp: lokal, Server oder beides */
}logInfo;

/*------------------------------------------------------------------------+
| Typdefinition für den in dsmQueryAccess() verwendeten Parameter qryRespAccessData           |
+------------------------------------------------------------------------*/

typedef struct
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   char              node[DSM_MAX_ID_LENGTH+1];     /* Knotenname             */
   char              owner[DSM_MAX_OWNER_LENGTH+1]; /* Eigner                 */
   dsmObjName        objName ;                      /* Objektname             */
   dsmAccessType     accessType;                    /* Archiv. oder Sicherung */
   dsUint32_t        ruleNumber ;                   /* Zugriffsregel-ID       */
}qryRespAccessData;

#define qryRespAccessDataVersion  1

/*------------------------------------------------------------------------+
|  Typdefinition für Parameter envSetUp in dsmSetUp()                     |
+------------------------------------------------------------------------*/
typedef struct S_envSetUp
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   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;  /* für Name von ausführbaren Dateien (argv[0]) */
   char            logName[DSM_NAME_MAX +1];
   dsmBool_t       reserved1;         /* für zukünftige Verwendung */
   dsmBool_t       reserved2;               /* für zukünftige Verwendung */
}envSetUp;

#define envSetUpVersion  4

/*------------------------------------------------------------------------+
|  Typdefinition für dsmInitExIn_t                                        |
+------------------------------------------------------------------------*/
typedef struct dsmInitExIn_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   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

/*-------------------------------------------------------------------------+
|  Typdefinition für dsmInitExOut_t                                        |
+-------------------------------------------------------------------------*/
typedef struct dsmInitExOut_t
{
   dsUint16_t        stVersion;              /* Strukturversion               */
   dsInt16_t         userNameAuthorities;
   dsInt16_t         infoRC;        /* Fehlercode, falls festgestellt */
   char              adsmServerName[DSM_MAX_SERVERNAME_LENGTH+1];
   dsUint16_t     serverVer;              /* Versionsnummer des Servers       */
   dsUint16_t        serverRel;     /* Releasenummer des Servers        */
   dsUint16_t     serverLev;              /* Stufennummer des Servers         */
   dsUint16_t        serverSubLev;  /* Unterstufennummer des Servers    */

   dsmBool_t         bIsFailOverMode; /* wahr im Fall einer Übernahme   */
   char              replServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* Replikationsservername */
   char              homeServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* Home-Servername        */

}dsmInitExOut_t;

#define dsmInitExOutVersion 3

/*-------------------------------------------------------------------------+
| Typdefinition für in logInfo verwendeten LogType                         |
+-------------------------------------------------------------------------*/
typedef enum
{
   logSevInfo = 0x00,       /* Information ANE4991 */
   logSevWarning,           /* Warnung     ANE4992 */
   logSevError,             /* Fehler      ANE4993 */
   logSevSevere,            /* schwerwiegend  ANE4994 */
   logSevLicense,           /* Lizenz      ANE4995 */
   logSevTryBuy             /* Probelizenz ANE4996 */
}dsmLogSeverity ;


/*-------------------------------------------------------------------------+
|  Typdefinition für dsmLogExIn_t                                          |
+-------------------------------------------------------------------------*/
typedef struct dsmLogExIn_t
{
   dsUint16_t        stVersion;              /* Strukturversion               */
   dsmLogSeverity    severity;
   char              appMsgID[8];
   dsmLogType        logType;    /* Protokolltyp: lokal, Server oder beides */
   char              *message;   /* Text der zu protokollierenden Nachricht */
   char              appName[DSM_MAX_PLATFORM_LENGTH];
   char              osPlatform[DSM_MAX_PLATFORM_LENGTH];
   char              appVersion[DSM_MAX_PLATFORM_LENGTH];
}dsmLogExIn_t;

#define dsmLogExInVersion 2

/*-------------------------------------------------------------------------+
|  Typdefinition für dsmlogExOut_t                                         |
+-------------------------------------------------------------------------*/
typedef struct dsmLogExOut_t
{
   dsUint16_t        stVersion;              /* Strukturversion               */
}dsmLogExOut_t;

#define dsmLogExOutVersion 1
```

```
/*--------------------------------------------------------------------------+
|  Typdefinition für dsmRenameIn_t                                          |
+--------------------------------------------------------------------------*/
typedef struct dsmRenameIn_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsUint32_t      dsmHandle;        /* Sitzungskennung */
   dsUint8_t        repository;                  /* Sicherung oder Archivierung */
   dsmObjName       *objNameP ;                  /* Objektname             */
   char             newHl[DSM_MAX_HL_LENGTH + 1];  /* neuer übergeordneter Name */
   char             newLl[DSM_MAX_LL_LENGTH + 1];  /* neuer untergeordneter Name */
   dsmBool_t        merge;                      /* mit vorhandenem Namen zusammenfassen */
   ObjID            objId;                      /* Objekt-ID für Archivierung */
}dsmRenameIn_t;

#define dsmRenameInVersion 1

/*--------------------------------------------------------------------------+
|  Typdefinition für dsmRenameOut_t                                         |
+--------------------------------------------------------------------------*/
typedef struct dsmRenameOut_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
}dsmRenameOut_t;

#define dsmRenameOutVersion 1

/*--------------------------------------------------------------------------+
|  Typdefinition für dsmEndSendObjExIn_t                                    |
+--------------------------------------------------------------------------*/
typedef struct dsmEndSendObjExIn_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsUint32_t      dsmHandle;        /* Sitzungskennung */
}dsmEndSendObjExIn_t;

#define dsmEndSendObjExInVersion 1

/*--------------------------------------------------------------------------+
|  Typdefinition für dsmEndSendObjExOut_t                                   |
+--------------------------------------------------------------------------*/
typedef struct dsmEndSendObjExOut_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsStruct64_t       totalBytesSent;      /* Gesamtsumme aus Anwendung gelesener Byte */
   dsmBool_t          objCompressed;       /* mit Objektkomprimierung */
   dsStruct64_t       totalCompressSize;   /* Gesamtgröße nach Komprimierung */
   dsStruct64_t       totalLFBytesSent;    /* Gesamtanzahl LAN-unabhängig gesendeter Byte */
   dsUint8_t          encryptionType;      /* verwendeter Verschlüsselungstyp */
   dsmBool_t          objDeduplicated;     /* Objektverarbeitung für verteilte Deduplizierung */
   dsStruct64_t       totalDedupSize;      /* Gesamtgröße nach Deduplizierung */
}dsmEndSendObjExOut_t;

#define dsmEndSendObjExOutVersion 3
/*--------------------------------------------------------------------------+
|  Typdefinition für dsmGroupHandlerIn_t                                    |
+--------------------------------------------------------------------------*/
typedef struct dsmGroupHandlerIn_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsUint32_t      dsmHandle;        /* Sitzungskennung */
   dsUint8_t       groupType;        /* Gruppentyp         */
   dsUint8_t       actionType;       /* Gruppenoperationstyp */
   dsUint8_t       memberType;       /* Mitgliedstyp: Leiter oder Mitglied */
   dsStruct64_t    leaderObjId;      /* OBJID des Gruppenleiters beim Bearbeiten eines Mitglieds */
   char            *uniqueGroupTagP; /* Eindeutige Gruppen-ID */
   dsmObjName      *objNameP ;       /* Objektname des Gruppenleiters */
   dsmGetList      memberObjList;    /* Liste der zu enfernenden bzw. zuzuordnenden Objekte */
}dsmGroupHandlerIn_t;
```

```
#define dsmGroupHandlerInVersion 1

/*------------------------------------------------------------------------+
|  Typdefinition für dsmGroupHandlerExOut_t                               |
+------------------------------------------------------------------------*/
typedef struct dsmGroupHandlerOut_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
}dsmGroupHandlerOut_t;

#define dsmGroupHandlerOutVersion 1

/*------------------------------------------------------------------------+
|  Typdefinition für dsmEndTxnExIn_t                                      |
+------------------------------------------------------------------------*/
typedef struct dsmEndTxnExIn_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsUint32_t      dsmHandle;        /* Sitzungskennung */
   dsUint8_t        vote;
}dsmEndTxnExIn_t;

#define dsmEndTxnExInVersion 1

/*------------------------------------------------------------------------+
|  Typdefinition für dsmEndTxnExOut_t                                     |
+------------------------------------------------------------------------*/
typedef struct dsmEndTxnExOut_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsUint16_t         reason;                 /* Ursachencode                  */
   dsStruct64_t       groupLeaderObjId;       /* Gruppenleiterobjekt-ID (für    */
                                              /* DSM_ACTION_OPEN zurückgegeben) */
   dsUint8_t          reserved1;              /* für zukünftige Verwendung      */
   dsUint16_t         reserved2;              /* für zukünftige Verwendung      */
}dsmEndTxnExOut_t;

#define dsmEndTxnExOutVersion 1

/*------------------------------------------------------------------------+
|  Typdefinition für dsmEndGetDataExIn_t                                  |
+------------------------------------------------------------------------*/
typedef struct dsmEndGetDataExIn_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsUint32_t     dsmHandle;     /* Sitzungskennung */
}dsmEndGetDataExIn_t;

#define dsmEndGetDataExInVersion 1

/*------------------------------------------------------------------------+
|  Typdefinition für dsmEndGetDataExOut_t                                 |
+------------------------------------------------------------------------*/
typedef struct dsmEndGetDataExOut_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsUint16_t         reason;                 /* Ursachencode                  */
   dsStruct64_t   totalLFBytesRecv;  /* Gesamtzahl LAN-unabhängig empfangener Byte */
}dsmEndGetDataExOut_t;

#define dsmEndGetDataExOutVersion 1

/*------------------------------------------------------------------------+
| Typdefinition für Objektliste in dsmRetentionEvent()                    |
+------------------------------------------------------------------------*/
typedef struct dsmObjList
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsUint32_t       numObjId ;         /* Anzahl Objekt-IDs in Liste       */
```

```
   ObjID           *objId;           /* Liste der als Signal zu sendenden Objekt-IDs */
}dsmObjList_t ;

#define dsmObjlistVersion 1

/*-------------------------------------------------------------------------+
| Typdefinition für den in dsmRetentionEvent verwendeten eventType         |
+-------------------------------------------------------------------------*/
typedef enum
{
   eventRetentionActivate = 0x00,   /* dem Server signalisieren, dass das Ereignis stattgefunden hat */
   eventHoldObj,                    /* Löschen/Verfall des Objekts aussetzen */
   eventReleaseObj                  /* normale Lösch-/Verfallsverarbeitung wiederaufnehmen */
}dsmEventType_t;

/*-------------------------------------------------------------------------+
|  Typdefinition für dsmRetentionEvent()                                   |
+-------------------------------------------------------------------------*/
typedef struct dsmRetentionEventIn_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsUint32_t         dsmHandle;                   /* Sitzungskennung    */
   dsmEventType_t     eventType;                   /* Ereignistyp      */
   dsmObjList_t       objList;                     /* Objekt-ID        */
}dsmRetentionEventIn_t;

#define dsmRetentionEventInVersion 1

/*-------------------------------------------------------------------------+
|  Typdefinition für dsmRetentionEvent()                                   |
+-------------------------------------------------------------------------*/
typedef struct dsmRetentionEventOut_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
}dsmRetentionEventOut_t;

#define dsmRetentionEventOutVersion 1


/*-------------------------------------------------------------------------+
|  Typdefinition für dsmRequestBuffer()                                    |
+-------------------------------------------------------------------------*/
typedef struct requestBufferIn_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsUint32_t         dsmHandle;                   /* Sitzungskennung    */
}requestBufferIn_t;

#define requestBufferInVersion 1

/*-------------------------------------------------------------------------+
|  Typdefinition für dsmRequestBuffer()                                    |
+-------------------------------------------------------------------------*/
typedef struct requestBufferOut_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsUint8_t        tsmBufferHandle;          /* Kennung zu TSM-Datenpuffer */
   char             *dataPtr;                 /* Adresse, in die Daten geschrieben werden */
   dsUint32_t       bufferLen;                /* Maximale Länge der zu schreibenden Daten */
}requestBufferOut_t;

#define requestBufferOutVersion 1

/*-------------------------------------------------------------------------+
|  Typdefinition für dsmReleaseBuffer()                                    |
+-------------------------------------------------------------------------*/
typedef struct releaseBufferIn_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsUint32_t         dsmHandle;                   /* Sitzungskennung    */
```

```
   dsUint8_t        tsmBufferHandle;       /* Kennung zu TSM-Datenpuffer */
   char               *dataPtr;                 /* Adresse, in die Daten geschrieben werden */
}releaseBufferIn_t;

#define releaseBufferInVersion 1

/*------------------------------------------------------------------------+
|  Typdefinition für dsmReleaseBuffer()                                   |
+------------------------------------------------------------------------*/
typedef struct releaseBufferOut_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
}releaseBufferOut_t;

#define releaseBufferOutVersion 1

/*------------------------------------------------------------------------+
|  Typdefinition für dsmGetBufferData()                                   |
+------------------------------------------------------------------------*/
typedef struct getBufferDataIn_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsUint32_t         dsmHandle;                   /* Sitzungskennung    */
}getBufferDataIn_t;

#define getBufferDataInVersion 1

/*------------------------------------------------------------------------+
|  Typdefinition für dsmGetBufferData()                                   |
+------------------------------------------------------------------------*/
typedef struct getBufferDataOut_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsUint8_t        tsmBufferHandle;       /* Kennung zu TSM-Datenpuffer */
   char             *dataPtr;              /* Adresse der aktuell zu lesenden Daten */
   dsUint32_t       numBytes;              /* Aktuell aus dataPtr zu lesende Anzahl Byte */
}getBufferDataOut_t;

#define getBufferDataOutVersion 1

/*------------------------------------------------------------------------+
|  Typdefinition für dsmSendBufferData()                                  |
+------------------------------------------------------------------------*/
typedef struct sendBufferDataIn_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsUint32_t         dsmHandle;             /* Sitzungskennung    */
   dsUint8_t          tsmBufferHandle;       /* Kennung zu TSM-Datenpuffer */
   char               *dataPtr;              /* Adresse der aktuell zu sendenden Daten */
   dsUint32_t         numBytes;              /* Anzahl der aktuell aus dataPtr zu sendenden Byte */
}sendBufferDataIn_t;

#define sendBufferDataInVersion 1

/*------------------------------------------------------------------------+
|  Typdefinition für dsmSendBufferData()                                  |
+------------------------------------------------------------------------*/
typedef struct sendBufferDataOut_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
}sendBufferDataOut_t;

#define sendBufferDataOutVersion 1

/*------------------------------------------------------------------------+
|  Typdefinition für dsmUpdateObjExIn_t                                   |
+------------------------------------------------------------------------*/
typedef struct dsmUpdateObjExIn_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
```

```
   dsUint32_t      dsmHandle;           /* Sitzungskennung     */
   dsmSendType     sendType;            /* Sendetyp (Sich./Archiv.) */
   char            *descrP;             /* Archivierungsbeschreibung */
   dsmObjName      *objNameP;           /* Objektname          */
   ObjAttr         *objAttrPtr;         /* Attribut            */
   dsUint32_t      objUpdAct;           /* Aktualisierungsaktion */
   ObjID           archObjId;           /* Objekt-ID für Archivierung */
}dsmUpdateObjExIn_t;

#define dsmUpdateObjExInVersion 1

/*--------------------------------------------------------------------------+
|  Typdefinition für dsmUpdateObjExOut_t                                    |
+--------------------------------------------------------------------------*/
typedef struct dsmUpdateObjExOut_t
{
    dsUint16_t         stVersion;              /* Strukturversion               */
}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-Clientkomponente                                                      *
 *                                                                           *
 * (C) Copyright IBM Corporation 1993, 2010                                  *
 *****************************************************************************/

/*****************************************************************************
 * Headerdateiname: tsmapitd.h
 *
 * Umgebung:        ***********************************************
 *                  ** Dies ist eine plattformunabhängige         **
*                  ** Quellendatei                               **
 *
 *                  ***********************************************
 *
 * Anm. zum Entwurf: Diese Datei enthält Basisdatentypen und Konstanten, die in
 *                    allen Clientquellendateien verwendet werden können. Die Konstanten
 *                    in dieser Datei müssen für die spezielle Maschine bzw. für das
 *                    spezielle Betriebssystem, auf dem die Clientsoftware ausgeführt
 *                    werden soll, ordnungsgemäß gesetzt werden.
 *
 *                    dsmapips.h enthält plattformspezifische Definitionen
 *
 * Beschreib. Name:  Definitionen für Konstanten der Tivoli Storage
*                   Manager-API
 *---------------------------------------------------------------------------*/

#ifndef _H_TSMAPITD
#define _H_TSMAPITD


/*=== Strukturausrichtung zum Packen der Strukturen definieren ===*/
#if _OPSYS_TYPE == DS_WINNT
#ifdef _WIN64
#pragma pack(8)
#else
#pragma pack(1)
```

```
#endif
#endif

#ifdef _MAC
#pragma options align = packed
#endif

/*============================================================
 Win32-Anwendungen, die die TSM-Schnittstelle verwenden,
 müssen während der Kompilierung das Flag -DUNICODE verwenden.
 ============================================================*/
#if _OPSYS_TYPE == DS_WINNT && !defined(DSMAPILIB)
#ifndef UNICODE
#error "Win32-Anwendungen, die die TSM-Schnittstelle verwenden, MÜSSEN mit dem Flag -DUNICODE kompiliert werden"
#endif
#endif

/*============================================================
 Mac OS X-Anwendungen, die die TSM-Schnittstelle verwenden,
 müssen während der Kompilierung das Flag -DUNICODE verwenden.
 ============================================================*/
#if _OPSYS_TYPE == DS_MACOS && !defined(DSMAPILIB)
#ifndef UNICODE
#error "Mac OS X-Anwendungen, die die TSM-Schnittstelle verwenden, MÜSSEN mit dem Flag -DUNICODE kompiliert werden"
#endif
#endif

/*-------------------------------------------------------------------------+
 | Typdefinition für Parameter dsmGetType in tsmBeginGetData()             |
 +-------------------------------------------------------------------------*/
typedef enum
{
   gtTsmBackup = 0x00,                    /* Sicherungsverarbeitungstyp    */
   gtTsmArchive                           /* Archivierungsverarbeitungstyp */
} tsmGetType ;

/*-------------------------------------------------------------------------+
 |  Typdefinition für Parameter dsmQueryType in tsmBeginQuery()            |
 +-------------------------------------------------------------------------*/
typedef enum
{
   qtTsmArchive = 0x00,                     /* Archivabfragetyp          */
   qtTsmBackup,                             /* Sicherungsabfragetyp      */
   qtTsmBackupActive,                     /* Schnellabfr. f. aktive Sich.dateien*/
   qtTsmFilespace,                          /* Dateibereichsabfragetyp   */
   qtTsmMC,                                 /* Verwaltungsklassenabfragetyp */
    qtTsmReserved1,                          /* zukünftige Verwendung     */
    qtTsmReserved2,                          /* zukünftige Verwendung     */
    qtTsmReserved3,                          /* zukünftige Verwendung     */
    qtTsmReserved4,                          /* zukünftige Verwendung     */
   qtTsmBackupGroups,                       /* Alle Gruppenleiter in einem bestimmten Dateibereich */
   qtTsmOpenGroups,                         /* Alle einem Leiter zugeordneten Gruppenmitglieder */
    qtTsmReserved5,                          /* zukünftige Verwendung     */
   qtTsmProxyNodeAuth,                      /* Knoten, an die dieser Knoten weiterleiten kann*/
    qtTsmProxyNodePeer,                      /* Peerknoten unter diesem Zielknoten */
    qtTsmReserved6,                         /* zukünftige Verwendung     */
    qtTsmReserved7,                         /* zukünftige Verwendung     */
    qtTsmReserved8                          /*  zukünftige Verwendung    */
} tsmQueryType ;

/*-------------------------------------------------------------------------+
 |  Typdefinition für Parameter sendType in tsmBindMC() und tsmSendObj()   |
 +-------------------------------------------------------------------------*/
typedef enum
{
   stTsmBackup = 0x00,                       /* Sicherungsverarbeitungstyp */
   stTsmArchive,                             /* Archivierungsverarbeitungstyp */
```

```
   stTsmBackupMountWait,           /* Sicherungsverarbeitung; 'Auf Ladevorgang warten' ist aktiviert*/
   stTsmArchiveMountWait           /* Archivierungsverarbeitung; 'Auf Ladevorgang warten' ist aktiviert*/
} tsmSendType ;

/*-------------------------------------------------------------------------+
 |  Typdefinition für Parameter delType in tsmDeleteObj()                  |
 +-------------------------------------------------------------------------*/
typedef enum
{
   dtTsmArchive = 0x00,                 /* Typ 'Archivierung löschen' */
   dtTsmBackup,                         /* Typ 'Sicherung löschen (inaktivieren)' */
   dtTsmBackupID                        /* Typ 'Sicherung löschen' (entfernen)' */
} tsmDelType ;

/*-------------------------------------------------------------------------+
 |  Typdefinition für Parameter sendType in tsmSetAccess()                 |
 +-------------------------------------------------------------------------*/
typedef enum
{
   atTsmBackup = 0x00,                   /* Sicherungsverarbeitungstyp */
   atTsmArchive                          /* Archivierungsverarbeitungstyp */
}tsmAccessType;

/*-------------------------------------------------------------------------+
 |  Typdefinition für Parameter Overwrite in tsmSendObj()                  |
 +-------------------------------------------------------------------------*/
typedef enum
{
   owIGNORE = 0x00,
   owYES,
   owNO
}tsmOwType;


/*-------------------------------------------------------------------------+
 |  Typdefinition für API-Version in tsmInit() und tsmQueryApiVersion()    |
 +-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t     stVersion;             /* Strukturversion             */
   dsUint16_t version;       /* API-Version                       */
   dsUint16_t release;       /* API-Release                       */
   dsUint16_t level;         /* API-Stand                         */
   dsUint16_t subLevel;      /* API-Unterstufe                    */
   dsmBool_t  unicode;       /* API-Unicode? */
} tsmApiVersionEx;

#define tsmApiVersionExVer       2

/*-------------------------------------------------------------------------+
|  Typdefinition für Anwendungsversion in tsmInit()                        |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t   stVersion;               /* Strukturversion                  */
   dsUint16_t   applicationVersion;      /* Anwendungsversionsnummer         */
   dsUint16_t   applicationRelease;      /* Anwendungsreleasenummer          */
   dsUint16_t   applicationLevel;        /* Anwendungsebenennummer           */
   dsUint16_t   applicationSubLevel;     /* Anwendungsunterstufennummer      */
} tsmAppVersion;

#define tsmAppVersionVer      1


/*-------------------------------------------------------------------------+
 |  Typdefinition für in BindMC, Send, Delete, Query verwendeten Objektnamen|
 +-------------------------------------------------------------------------*/
```

```
typedef struct tsmObjName
{
   dsChar_t   fs[DSM_MAX_FSNAME_LENGTH + 1] ;             /* Dateibereichsname */
   dsChar_t   hl[DSM_MAX_HL_LENGTH + 1] ;                 /* Übergeordneter Name */
   dsChar_t   ll[DSM_MAX_LL_LENGTH + 1] ;                 /* Untergeordneter Name */
   dsUint8_t  objType;              /* Definitionen für Objekttypwerte siehe oben */
   dsChar_t   dirDelimiter;
} tsmObjName;

/*------------------------------------------------------------------------+
 |  Typdefinition f. Informationen zu 'Sicherung löschen' in dsmDeleteObj() |
 +------------------------------------------------------------------------*/
typedef struct tsmDelBack
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   tsmObjName       *objNameP ;                   /* Objektname             */
   dsUint32_t       copyGroup ;                   /* Kopiengruppe           */
} tsmDelBack ;

#define  tsmDelBackVersion    1

/*------------------------------------------------------------------------+
 |  Typdefinition für Infos zu 'Archivierung löschen' in dsmDeleteObj()     |
 +------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsStruct64_t     objId ;                       /* Objekt-ID              */
} tsmDelArch ;

#define  tsmDelArchVersion  1

/*------------------------------------------------------------------------+
 |  Typdefinition für Infos zu 'Sicherungs-ID löschen' in dsmDeleteObj()    |
 +------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsStruct64_t     objId ;                       /* Objekt-ID              */
} tsmDelBackID;

#define  tsmDelBackIDVersion  1


/*------------------------------------------------------------------------+
 |  Typdefinition für Löschinformationen in dsmDeleteObj()                  |
 +------------------------------------------------------------------------*/
typedef union
{
   tsmDelBack   backInfo ;
   tsmDelArch   archInfo ;
   tsmDelBackID backIDInfo;
} tsmDelInfo ;

/*------------------------------------------------------------------------+
 |  Typdefinition für Parameter Object Attribute in dsmSendObj()            |
 +------------------------------------------------------------------------*/
typedef struct tsmObjAttr
{
   dsUint16_t   stVersion;               /* Strukturversion                  */
   dsChar_t     owner[DSM_MAX_OWNER_LENGTH + 1];      /* Objekteigner          */
   dsStruct64_t sizeEstimate;            /* Größenschätzung des Objekts in Byte */
   dsmBool_t    objCompressed;           /* Ist Objekt bereits komprimiert? */
   dsUint16_t   objInfoLength;           /* Länge der objektabhängigen Information */
   char         *objInfo;                /* Bytepuffer für objektabhängige Informationen */
   dsChar_t     *mcNameP;                /* Verwaltungsklassenname für Überschreibung */
```

```
   tsmOwType    reserved1;               /* für zukünftige Verwendung         */
   tsmOwType    reserved2;               /* für zukünftige Verwendung         */
   dsmBool_t    disableDeduplication;    /* 'Keine Deduplizierung' für dieses Objekt erzwingen */
   dsmBool_t    useExtObjInfo;                 /* Erw. Objektinfo. bis zu 1536 verwenden */
} tsmObjAttr;

#define tsmObjAttrVersion 5

/*------------------------------------------------------------------------+
 | Typdefinition für zurückgegebenen mcBindKey in dsmBindMC()             |
 +------------------------------------------------------------------------*/
typedef struct tsmMcBindKey
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsChar_t    mcName[DSM_MAX_MC_NAME_LENGTH + 1];
   /* Name der an das Objekt gebundenen Verwaltungsklasse. */
   dsmBool_t   backup_cg_exists;                          /* Wahr/Falsch */
   dsmBool_t   archive_cg_exists;                         /* Wahr/Falsch */
   dsChar_t    backup_copy_dest[DSM_MAX_CG_DEST_LENGTH + 1];
   /* Zielname für Sicherungskopie */
   dsChar_t    archive_copy_dest[DSM_MAX_CG_DEST_LENGTH + 1];
   /* Zielname für Archivierungskopie */
} tsmMcBindKey;

#define tsmMcBindKeyVersion  1


/*------------------------------------------------------------------------+
 |  Typdefinition für queryBuffer für Verwaltungsklasse in dsmBeginQuery()  |
 +------------------------------------------------------------------------*/
typedef struct tsmQryMCData
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsChar_t     *mcName;                          /* Verwaltungsklassenname */
   /* einzelner Name für 1 Klasse oder leere Zeichenfolge, um alle abzurufen */
   dsmBool_t    mcDetail;                          /* Details erwünscht oder nicht? */
} tsmQryMCData;

#define tsmQryMCDataVersion  1


/*------------------------------------------------------------------------+
 |  Typdefinition f. Archivierungskopiengruppendetails in Antwort auf Verwaltungsklassenabfrage|
 +------------------------------------------------------------------------*/
typedef struct tsmArchDetailCG
{
   dsChar_t     cgName[DSM_MAX_CG_NAME_LENGTH + 1];     /* Kopiengruppenname */
   dsUint16_t   frequency;                      /* Häufigkeit von Kopien (Archivierungen) */
   dsUint16_t   retainVers;                           /* Version aufbewahren */
   dsUint8_t    copySer;       /* Hinweise zur Kopiennummerierung siehe Definitionen */
   dsUint8_t    copyMode;         /* Hinweise zu Kopiermoduswerten siehe Definitionen oben */
   dsChar_t     destName[DSM_MAX_CG_DEST_LENGTH + 1];    /* Zielname für Kopie */
   dsmBool_t    bLanFreeDest;            /* Ziel hat LAN-unabhängigen Pfad?    */
   dsmBool_t    reserved;                /* Derzeit nicht verwendet          */
   dsUint8_t    retainInit;              /* Mögliche Werte siehe dsmapitd.h  */
   dsUint16_t   retainMin;               /* wenn retInit auf EVENT gesetzt: Anzahl Tage */
   dsmBool_t    bDeduplicate;            /* Am Ziel ist Deduplizierung aktiviert    */
}tsmArchDetailCG;

/*------------------------------------------------------------------------+
 |  Typdefinition f. Sicherungskopiengruppendetails in Antwort auf Verwaltungsklassenabfrage   |
 +------------------------------------------------------------------------*/
typedef struct tsmBackupDetailCG
{
   dsChar_t     cgName[DSM_MAX_CG_NAME_LENGTH + 1];     /* Kopiengruppenname */
   dsUint16_t   frequency;                              /* Häufigkeit der Sicherung */
   dsUint16_t   verDataExst;                          /* Versionen bestehender Daten */
```

```
   dsUint16_t   verDataDltd;                        /* Versionen gelöschter Daten */
   dsUint16_t   retXtraVers;                        /* Extraversionen aufbewahren */
   dsUint16_t   retOnlyVers;                        /* Einzige Version aufbewahren */
   dsUint8_t    copySer;       /* Hinweise zur Kopiennummerierung siehe Definitionen */
   dsUint8_t    copyMode;         /* Hinweise zu Kopiermoduswerten siehe Definitionen oben */
   dsChar_t     destName[DSM_MAX_CG_DEST_LENGTH + 1];    /* Zielname für Kopie */
   dsmBool_t    bLanFreeDest;            /* Ziel hat LAN-unabhängigen Pfad?    */
   dsmBool_t    reserved;                /* Derzeit nicht verwendet          */
   dsmBool_t    bDeduplicate;            /* Am Ziel ist Deduplizierung aktiviert    */
}tsmBackupDetailCG;


/*-------------------------------------------------------------------------+
 |  Typdefinition für Detailantwort zu Verwaltungsklassenabfrage in dsmGetNextQObj()           |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryRespMCDetailData
{
   dsUint16_t         stVersion;               /* Strukturversion               */
   dsChar_t        mcName[DSM_MAX_MC_NAME_LENGTH + 1];        /* Verwaltungsklassenname */
   dsChar_t        mcDesc[DSM_MAX_MC_DESCR_LENGTH + 1]; /* Verwaltungsklassenbeschreibung */
   archDetailCG    archDet;                 /* Archivierungskopiengruppendetail */
   backupDetailCG  backupDet;                /* Sicherungskopiengruppendetail */
} tsmQryRespMCDetailData;

#define tsmQryRespMCDetailDataVersion  4

/*-------------------------------------------------------------------------+
 | Typdefinition f. Antwort mit Zusammenfass. zur Verwaltungsklassenabfrage in dsmGetNextQObj()|
 +-------------------------------------------------------------------------*/
typedef struct tsmQryRespMCData
{
   dsUint16_t         stVersion;               /* Strukturversion               */
   dsChar_t        mcName[DSM_MAX_MC_NAME_LENGTH + 1];        /* Verwaltungsklassenname */
   dsChar_t     mcDesc[DSM_MAX_MC_DESCR_LENGTH + 1];    /* Verwaltungsklassenbeschreibung */
}tsmQryRespMCData;

#define tsmQryRespMCDataVersion  1

/*-------------------------------------------------------------------------+
 |  Typdefinition für queryBuffer für Archivierung in tsmBeginQuery()       |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryArchiveData
{
   dsUint16_t         stVersion;               /* Strukturversion               */
   tsmObjName   *objName;                     /* Vollständiger DSM-Name des Objekts */
   dsChar_t     *owner;                                  /* Eignername */
   /* Hinweise zu maximalen Datumsgrenzen siehe Definitionen oben */
   dsmDate      insDateLowerBound;      /* unterer Grenzwert für Archivierungseinfügedatum */
   dsmDate      insDateUpperBound;      /* oberer Grenzwert für Archivierungseinfügedatum */
   dsmDate      expDateLowerBound;      /* unterer Grenzwert für Verfallsdatum */
   dsmDate      expDateUpperBound;      /* oberer Grenzwert für Verfallsdatum */
   dsChar_t     *descr;                 /* Archivierungsbeschreibung */
} tsmQryArchiveData;

#define tsmQryArchiveDataVersion  1

/*-------------------------------------------------------------------------+
| Typdefinition für Antwort auf Archivierungsabfrage in dsmGetNextQObj()   |
+-------------------------------------------------------------------------*/
typedef struct tsmQryRespArchiveData
{
    dsUint16_t         stVersion;               /* Strukturversion               */
    tsmObjName      objName;                           /* Qualifikationsmerkmal für Dateibereichsname */
    dsUint32_t      copyGroup;                         /* Kopiengruppenzahl */
    dsChar_t        mcName[DSM_MAX_MC_NAME_LENGTH + 1];  /* Verwaltungsklassenname */
    dsChar_t        owner[DSM_MAX_OWNER_LENGTH + 1];     /* Eignername */
    dsStruct64_t    objId;                             /* Eindeutige Kopien-ID */
```

```
    dsStruct64_t    reserved;                       /* Abwärtskompatibilität */
    dsUint8_t       mediaClass;                         /* Datenträgerzugriffsklasse */
    dsmDate         insDate;                             /* Archivierungseinfügedatum */
    dsmDate         expDate;                     /* Verfallsdatum für Objekt */
    dsChar_t        descr[DSM_MAX_DESCR_LENGTH + 1];     /* Archivierungsbeschreibung */
    dsUint16_t  objInfolen;                           /* Länge der objektabhängigen Informationen */
    dsUint8_t       reservedObjInfo[DSM_MAX_OBJINFO_LENGTH];  /* objektabhängige Informationen */
    dsUint160_t     restoreOrderExt;                    /* Zurückschreibungsreihenfolge */
    dsStruct64_t    sizeEstimate;              /* vom Benutzer gespeicherte Größenschätzung */
    dsUint8_t       compressType;                       /* Komprimierungsflag */
    dsUint8_t       retentionInitiated;                 /* Objekt wartet bei Aufbewahrungsereignis */
    dsUint8_t       objHeld;                            /* Objekt angehalten; Werte siehe dsmapitd.h */
    dsUint8_t       encryptionType;          /* Verschlüsselungstyp */
    dsmBool_t       clientDeduplicated;      /* Objekt wird von API dedupliziert */
    dsUint8_t       objInfo[DSM_MAX_EXT_OBJINFO_LENGTH];  /* objektabhängige Informationen */
    dsChar_t        compressAlg[DSM_MAX_COMPRESSTYPE_LENGTH + 1];  /* Komprimierungsalgorithmusname */
} tsmQryRespArchiveData;

#define tsmQryRespArchiveDataVersion  7

/*-------------------------------------------------------------------------+
 |  Typdefinition für Archivierungsparameter sendBuff in dsmSendObj()      |
 +-------------------------------------------------------------------------*/
typedef struct tsmSndArchiveData
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsChar_t     *descr;                  /* Archivierungsbeschreibung */
} tsmSndArchiveData;

#define tsmSndArchiveDataVersion  1

/*-------------------------------------------------------------------------+
 |  Typdefinition für queryBuffer für Sicherung in dsmBeginQuery()         |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryBackupData
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   tsmObjName  *objName;          /* Vollständiger DSM-Name des Objekts */
   dsChar_t    *owner;            /* Eignername */
   dsUint8_t   objState;          /* Objektstatusselektor */
   dsmDate     pitDate;           /* Datumswert für zeitpunktgesteuerte Zurückschreibung */
   /* Hinweise zu möglichen Werten siehe Definitionen oben */
    dsUint32_t  reserved1;
    dsUint32_t  reserved2;
} tsmQryBackupData;

#define tsmQryBackupDataVersion  3

/*-------------------------------------------------------------------------+
 | Typdefinition für Antwort auf Sicherungsabfrage in dsmGetNextQObj()     |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryRespBackupData
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   tsmObjName      objName;                      /* Vollständiger DSM-Name des Objekts */
   dsUint32_t      copyGroup;                          /* Kopiengruppenzahl */
   dsChar_t        mcName[DSM_MAX_MC_NAME_LENGTH + 1];  /* Verwaltungsklassenname */
   dsChar_t        owner[DSM_MAX_OWNER_LENGTH + 1];     /* Eignername */
   dsStruct64_t    objId;                              /* Eindeutige Objekt-ID */
   dsStruct64_t    reserved;                        /* Abwärtskompatibilität */
   dsUint8_t       mediaClass;                         /* Datenträgerzugriffsklasse */
   dsUint8_t       objState;                           /* Objektstatus, aktiv usw. */
   dsmDate     insDate;                           /* Sicherungseinfügedatum */
   dsmDate         expDate;                    /* Verfallsdatum für Objekt */
   dsUint16_t  objInfolen;                         /* Länge der objektabhängigen Informationen */
   dsUint8_t       reservedObjInfo[DSM_MAX_OBJINFO_LENGTH];  /* objektabhängige Informationen */
   dsUint160_t     restoreOrderExt;                    /* Zurückschreibungsreihenfolge */
```

```
   dsStruct64_t    sizeEstimate;                      /* vom Benutzer gespeicherte Größenschätzung */
   dsStruct64_t    baseObjId;
   dsUint16_t      baseObjInfolen;                    /* Länge der basisobjektabhängigen Informationen */
   dsUint8_t       baseObjInfo[DSM_MAX_OBJINFO_LENGTH]; /* basisobjektabhängige Informationen */
   dsUint160_t     baseRestoreOrder;                  /* Zurückschreibungsreihenfolge */
   dsUint32_t      fsID;
   dsUint8_t       compressType;
   dsmBool_t       isGroupLeader;
   dsmBool_t       isOpenGroup;
   dsUint8_t       reserved1;              /* für zukünftige Verwendung */
   dsmBool_t       reserved2;              /* für zukünftige Verwendung */
   dsUint16_t      reserved3;              /* für zukünftige Verwendung */
   reservedInfo_t  *reserved4;             /* für zukünftige Verwendung */
   dsUint8_t       encryptionType;         /* Verschlüsselungstyp */
   dsmBool_t       clientDeduplicated;     /* Objekt wird von API dedupliziert */
   dsUint8_t       objInfo[DSM_MAX_EXT_OBJINFO_LENGTH];  /* objektabhängige Informationen */
   dsChar_t        compressAlg[DSM_MAX_COMPRESSTYPE_LENGTH + 1];  /* Komprimierungsalgorithmusname */
} tsmQryRespBackupData;

#define tsmQryRespBackupDataVersion  8
/*------------------------------------------------------------------------+
 |  Typdefinition für queryBuffer für aktive Sicherung in dsmBeginQuery()   |
 |                                                                          |
 |  Hinweise: Für die Abfrage der aktiven Sicherung müssen nur die Felder   |
 |            fs (Dateibereich) und objType gesetzt werden. objType kann nur|
 |            auf DSM_OBJ_FILE oder DSM_OBJ_DIRECTORY gesetzt werden.       |
 |            Für DSM_OBJ_ANY_TYPE wird keine Übereinstimmung in der Abfrage|
 |            gefunden.                                                     |
 +------------------------------------------------------------------------*/
typedef struct tsmQryABackupData
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   tsmObjName      *objName;                    /* Nur fs und objType werden verwendet */
} tsmQryABackupData;

#define tsmQryABackupDataVersion  1

/*------------------------------------------------------------------------+
 | Typdefinition für Antwort auf Abfrage der aktiven Sicherung in dsmGetNextQObj()            |
 +------------------------------------------------------------------------*/
typedef struct tsmQryARespBackupData
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   tsmObjName  objName;                      /* Vollständiger DSM-Name des Objekts */
   dsUint32_t  copyGroup;                         /* Kopiengruppenzahl */
   dsChar_t    mcName[DSM_MAX_MC_NAME_LENGTH + 1];  /* Verwaltungsklassenname */
   dsChar_t        owner[DSM_MAX_OWNER_LENGTH + 1];     /* Eignername */
   dsmDate     insDate;                           /* Sicherungseinfügedatum */
   dsUint16_t  objInfolen;                        /* Länge der objektabhängigen Informationen */
   dsUint8_t       reservedObjInfo[DSM_MAX_OBJINFO_LENGTH];  /* objektabhängige Informationen */
   dsUint8_t       objInfo[DSM_MAX_EXT_OBJINFO_LENGTH];  /* objektabhängige Informationen */
} tsmQryARespBackupData;

#define tsmQryARespBackupDataVersion  2

/*------------------------------------------------------------------------+
 |  Typdefinition für queryBuffer für Sicherung in dsmBeginQuery()          |
 +------------------------------------------------------------------------*/
typedef struct tsmQryBackupGroups
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   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

/*-------------------------------------------------------------------------+
 |  Typdefinition für queryBuffer für Proxy-Knoten in tsmBeginQuery()      |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryProxyNodeData
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsChar_t    *targetNodeName;            /* Zielknotenname  */
}tsmQryProxyNodeData;

#define tsmQryProxyNodeDataVersion  1

/*-------------------------------------------------------------------------+
 | Typdefinition für den in tsmGetNextQObj() verwendeten Parameter qryRespProxyNodeData       |
 +-------------------------------------------------------------------------*/

typedef struct tsmQryRespProxyNodeData
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsChar_t          targetNodeName[DSM_MAX_ID_LENGTH+1];  /* Zielknotenname  */
   dsChar_t          peerNodeName[DSM_MAX_ID_LENGTH+1];    /* Peerknotenname  */
   dsChar_t          hlAddress[DSM_MAX_ID_LENGTH+1];       /* übergeordnete Peeradresse */
   dsChar_t          llAddress[DSM_MAX_ID_LENGTH+1];       /* untergeordnete Peeradresse */
}tsmQryRespProxyNodeData;

#define tsmQryRespProxyNodeDataVersion  1

/*-------------------------------------------------------------------------+
 |  Typdefinition für WINNT- und OS/2-Dateibereichsattribute               |
 +-------------------------------------------------------------------------*/
typedef struct tsmDosFSAttrib
{
   osChar_t     driveLetter ;          /* Laufwerkbuchstabe für Dateibereich */
   dsUint16_t   fsInfoLength;          /* für Dateibereichsinformationen verwendete Länge */
   osChar_t     fsInfo[DSM_MAX_FSINFO_LENGTH]; /* vom Aufrufenden bestimmte Daten */
} tsmDosFSAttrib ;

/*-------------------------------------------------------------------------+
 |  Typdefinition für UNIX-Dateibereichsattribute                          |
 +-------------------------------------------------------------------------*/
typedef struct tsmUnixFSAttrib
{
   dsUint16_t   fsInfoLength;          /* für Dateibereichsinformationen verwendete Länge */
   osChar_t     fsInfo[DSM_MAX_FSINFO_LENGTH]; /* vom Aufrufenden bestimmte Daten */
} tsmUnixFSAttrib ;

/*-------------------------------------------------------------------------+
 |  Typdefinition für NetWare-Dateibereichsattribute                       |
 +-------------------------------------------------------------------------*/
typedef tsmUnixFSAttrib tsmNetwareFSAttrib;

/*-------------------------------------------------------------------------+
 |  Typdefinition für Dateibereichsattribute in allen Dateibereichsaufrufen |
 +-------------------------------------------------------------------------*/
typedef union
{
   tsmNetwareFSAttrib  netwareFSAttr;
   tsmUnixFSAttrib     unixFSAttr ;
   tsmDosFSAttrib      dosFSAttr ;
} tsmFSAttr ;
```

```
/*------------------------------------------------------------------------+
 |  Typdefinition für Parameter fsUpd in dsmUpdateFS()                     |
 +------------------------------------------------------------------------*/
typedef struct   tsmFSUpd
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsChar_t        *fsType ;              /* Dateibereichstyp              */
   dsStruct64_t    occupancy ;            /* Belegungsschätzung            */
   dsStruct64_t    capacity  ;            /* Kapazitätsschätzung           */
   tsmFSAttr       fsAttr ;                      /* plattformspezifische Attribute */
} tsmFSUpd ;

#define tsmFSUpdVersion  1

/*------------------------------------------------------------------------+
 |  Typdefinition für queryBuffer für Dateibereich in dsmBeginQuery()      |
 +------------------------------------------------------------------------*/
typedef struct tsmQryFSData
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsChar_t    *fsName;                  /* Dateibereichsname */
} tsmQryFSData;

#define tsmQryFSDataVersion  1

/*------------------------------------------------------------------------+
 | Typdefinition für Antwort auf Dateibereichsabfrage in dsmGetNextQObj()  |
 +------------------------------------------------------------------------*/
typedef struct tsmQryRespFSData
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsChar_t       fsName[DSM_MAX_FSNAME_LENGTH + 1];     /* Dateibereichsname */
   dsChar_t       fsType[DSM_MAX_FSTYPE_LENGTH + 1] ;    /* Dateibereichstyp  */
   dsStruct64_t    occupancy;                     /* Belegungsschätzung in Byte */
   dsStruct64_t   capacity;                       /* Kapazitätsschätzung in Byte */
   tsmFSAttr       fsAttr ;                      /* plattformspezifische Attribute */
   dsmDate        backStartDate;                 /* Datum für Sicherungsbeginn */
   dsmDate        backCompleteDate;        /* Datum für Sicherungsende   */
    dsmDate        reserved1;                     /* Für zukünftige Verwendung  */
   dsmBool_t      bIsUnicode;
   dsUint32_t     fsID;
   dsmDate        lastReplStartDate;        /* Startzeitpunkt der letzten Replikation     */
   dsmDate        lastReplCmpltDate;        /* Endzeitpunkt der letzten Replikation       */
                                            /*   (eventuell trat ein Fehler auf,          */
                                            /*    aber sie wurde dennoch beendet)         */
   dsmDate        lastBackOpDateFromServer; /* Die letzte Speicherungszeitmarke, die der  */
                                            /* Client auf dem Server gespeichert hat      */
   dsmDate        lastArchOpDateFromServer; /* Die letzte Speicherungszeitmarke, die der  */
                                            /* Client auf dem Server gespeichert hat      */
   dsmDate        lastSpMgOpDateFromServer; /* Die letzte Speicherungszeitmarke, die der  */
                                            /* Client auf dem Server gespeichert hat      */
   dsmDate        lastBackOpDateFromLocal;  /* Die letzte Speicherungszeitmarke, die der  */
                                            /*    Client lokal gespeichert hat            */
   dsmDate        lastArchOpDateFromLocal;  /* Die letzte Speicherungszeitmarke, die der  */
                                            /*    Client lokal gespeichert hat            */
   dsmDate        lastSpMgOpDateFromLocal;  /* Die letzte Speicherungszeitmarke, die der  */
                                            /*    Client lokal gespeichert hat            */
   dsInt32_t      failOverWriteDelay;       /* Minuten, die der Client vor dem Speichern  */
                                            /* auf diesem Rpl.server warten muss; Sonder- */
                                            /* codes: NO_ACCESS(-1), ACCESS_RDONLY (-2)   */
} tsmQryRespFSData;

#define tsmQryRespFSDataVersion 5

/*------------------------------------------------------------------------+
 |  Typdefinition für Parameter regFilespace in dsmRegisterFS()            |
```

```
 +------------------------------------------------------------------------*/
typedef struct tsmRegFSData
{
   dsUint16_t         stVersion;               /* Strukturversion               */
   dsChar_t        *fsName;                       /* Dateibereichsname */
   dsChar_t        *fsType;                       /* Dateibereichstyp */
   dsStruct64_t    occupancy;                     /* Belegungsschätzung in Byte */
   dsStruct64_t    capacity;                      /* Kapazitätsschätzung in Byte */
   tsmFSAttr       fsAttr ;                       /* plattformspezifische Attribute */
} tsmRegFSData;

#define tsmRegFSDataVersion  1

/*------------------------------------------------------------------------+
 |  Typdefinition für Antwort mit Sitzungsdaten in dsmQuerySessionInfo()    |
 +------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t   stVersion;               /* Strukturversion                  */
   /*---------------------------------------------------------------------*/
   /*           Serverinformationen                                       */
   /*---------------------------------------------------------------------*/
   dsChar_t       serverHost[DSM_MAX_SERVERNAME_LENGTH+1];
       /* Netzhostname des DSM-Servers */
   dsUint16_t     serverPort;             /* Serverkommunikationsport auf Host */
   dsmDate        serverDate;             /* Datum/Zeit des Servers */
   dsChar_t       serverType[DSM_MAX_SERVERTYPE_LENGTH+1];
        /* Ausführungsplattform des Servers */
   dsUint16_t     serverVer;              /* Versionsnummer des Servers       */
   dsUint16_t     serverRel;              /* Releasenummer des Servers        */
   dsUint16_t     serverLev;              /* Stufennummer des Servers         */
   dsUint16_t         serverSubLev;  /* Unterstufennummer des Servers    */
   /*---------------------------------------------------------------------*/
   /*           Clientstandardwerte                                       */
   /*---------------------------------------------------------------------*/
   dsChar_t       nodeType[DSM_MAX_PLATFORM_LENGTH+1]; /* Knoten-/Anwendungstyp */
   dsChar_t       fsdelim;                /* Dateibereichsbegrenzer             */
   dsChar_t       hldelim;                /* Begrenzer zwischen oberer u. unterere Ebene */
   dsUint8_t      compression;            /* Komprimierungsflag */
   dsUint8_t      archDel;                /* Berechtigung zum Löschen des Archivs */
   dsUint8_t      backDel;                /* Berechtigung zum Löschen der Sicherung */
   dsUint32_t     maxBytesPerTxn;         /* für zukünftige Verwendung */
   dsUint16_t     maxObjPerTxn;           /* Max. Anzahl Objekte in einer Transaktion */
   /*---------------------------------------------------------------------*/
   /*           Sitzungsdaten                                             */
   /*---------------------------------------------------------------------*/
   dsChar_t       id[DSM_MAX_ID_LENGTH+1];    /* Knotenname der Anmelde-ID */
   dsChar_t       owner[DSM_MAX_OWNER_LENGTH+1]; /* Anmeldeeigner */
   /*   (für Mehrbenutzerplattformen   */
   dsChar_t       confFile[DSM_PATH_MAX + DSM_NAME_MAX +1];
   /* Länge ist plattformabhängig       */
   /* dsInit-Name der Anwend.-Konfig.-Datei */
   dsUint8_t  opNoTrace;              /* Option dsInit - NoTrace = 1      */
   /*---------------------------------------------------------------------*/
   /*           Maßnahmendaten                                            */
   /*---------------------------------------------------------------------*/
   dsChar_t       domainName[DSM_MAX_DOMAIN_LENGTH+1]; /* Domänenname */
   dsChar_t       policySetName[DSM_MAX_PS_NAME_LENGTH+1];
   /* Name der aktiven Maßnahmengruppe */
   dsmDate        polActDate;         /* Aktivierungsdatum der Maßnahmengruppe */
   dsChar_t       dfltMCName[DSM_MAX_MC_NAME_LENGTH+1]; /* Standardverwaltungsklasse */
   dsUint16_t     gpBackRetn;             /* Karenzzeit f. Aufbewahrungszeitraum f. Sicherung */
   dsUint16_t     gpArchRetn;             /* Karenzzeit f. Aufbewahrungszeitraum f. Archivierung */
   dsChar_t       adsmServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* ADSM-Servername */
  dsmBool_t      archiveRetentionProtection; /* Aufbewahrungsschutz durch Server ist aktiviert*/
   dsUint64_t     maxBytesPerTxn_64;          /* für zukünftige Verwendung     */
   dsmBool_t      lanFreeEnabled;            /* Option 'LAN-unabhängig' ist gesetzt */
```

```
   dsmDedupType   dedupType;                /* server oder clientOrServer       */
   dsChar_t       accessNode[DSM_MAX_ID_LENGTH+1];    /* als Knotenname    */

   /*---------------------------------------------------------------*/
   /*           Replikations- und Übernahmeinformationen             */
   /*---------------------------------------------------------------*/
   dsmFailOvrCfgType failOverCfgType; /* Status der Übernahme */
   dsChar_t           replServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* Replikationsservername           */
   dsChar_t           homeServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* Home-Servername                  */
   dsChar_t        replServerHost[DSM_MAX_SERVERNAME_LENGTH+1]; /* Netzhostname des DSM-Servers      */
   dsInt32_t      replServerPort;                              /* Serverkommunikationsport auf Host */

} tsmApiSessInfo;

#define tsmApiSessInfoVersion  6

/*-------------------------------------------------------------------------+
 | Typdefinition für Antwort auf Abfrageoptionen in dsmQueryCliOptions()    |
 |      und 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

/*-------------------------------------------------------------------------+
 | Typdefinition für den in dsmQueryAccess() verwendeten Parameter qryRespAccessData          |
 +-------------------------------------------------------------------------*/

typedef struct
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsChar_t          node[DSM_MAX_ID_LENGTH+1];     /* Knotenname             */
   dsChar_t          owner[DSM_MAX_OWNER_LENGTH+1]; /* Eigner                 */
   tsmObjName        objName ;                      /* Objektname             */
   dsmAccessType     accessType;                    /* Archiv. oder Sicherung */
   dsUint32_t        ruleNumber ;                   /* Zugriffsregel-ID       */

}tsmQryRespAccessData;

#define tsmQryRespAccessDataVersion  1

/*-------------------------------------------------------------------------+
 |  Typdefinition für Parameter envSetUp in dsmSetUp()                      |
 +-------------------------------------------------------------------------*/
typedef struct tsmEnvSetUp
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   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;  /* für Name von ausführbaren Dateien (argv[0]) */
   dsChar_t        logName[DSM_NAME_MAX +1];
   dsmBool_t       reserved1;         /* für zukünftige Verwendung */
   dsmBool_t       reserved2;              /* für zukünftige Verwendung */
```

```
} tsmEnvSetUp;

#define tsmEnvSetUpVersion  4

/*------------------------------------------------------------------------+
 |  Typdefinition für dsmInitExIn_t                                       |
 +------------------------------------------------------------------------*/
typedef struct tsmInitExIn_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   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

/*------------------------------------------------------------------------+
 |  Typdefinition für dsmInitExOut_t                                      |
 +------------------------------------------------------------------------*/
typedef struct tsmInitExOut_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsInt16_t          userNameAuthorities;
   dsInt16_t          infoRC;        /* Fehlercode, falls festgestellt */
   /* ADSM-Servername               */
   dsChar_t           adsmServerName[DSM_MAX_SERVERNAME_LENGTH+1];
   dsUint16_t     serverVer;              /* Versionsnummer des Servers       */
   dsUint16_t         serverRel;     /* Releasenummer des Servers        */
   dsUint16_t     serverLev;              /* Stufennummer des Servers         */
   dsUint16_t         serverSubLev;  /* Unterstufennummer des Servers    */
   dsmBool_t          bIsFailOverMode; /* wahr im Fall einer Übernahme   */
   dsChar_t           replServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* Replikationsservername           */
   dsChar_t           homeServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* Home-Servername                  */
} tsmInitExOut_t;

#define tsmInitExOutVersion 3

/*------------------------------------------------------------------------+
 |  Typdefinition für dsmLogExIn_t                                        |
 +------------------------------------------------------------------------*/
typedef struct tsmLogExIn_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsmLogSeverity     severity;
   dsChar_t           appMsgID[8];
   dsmLogType         logType;    /* Protokolltyp: lokal, Server oder beides */
   dsChar_t           *message;   /* Text der zu protokollierenden Nachricht */
   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

/*-------------------------------------------------------------------------+
 |  Typdefinition für dsmlogExOut_t                                        |
 +-------------------------------------------------------------------------*/
typedef struct tsmLogExOut_t
{
   dsUint16_t         stVersion;              /* Strukturversion              */
} tsmLogExOut_t;

#define tsmLogExOutVersion 1

/*-------------------------------------------------------------------------+
 |  Typdefinition für dsmRenameIn_t                                        |
 +-------------------------------------------------------------------------*/
typedef struct tsmRenameIn_t
{
   dsUint16_t         stVersion;              /* Strukturversion              */
   dsUint32_t       tsmHandle;                  /* Sitzungskennung */
   dsUint8_t        repository;                 /* Sicherung oder Archivierung */
   tsmObjName       *objNameP ;                 /* Objektname            */
   dsChar_t         newHl[DSM_MAX_HL_LENGTH + 1];  /* neuer übergeordneter Name */
   dsChar_t         newLl[DSM_MAX_LL_LENGTH + 1];  /* neuer untergeordneter Name */
   dsmBool_t        merge;                     /* mit vorhandenem Namen zusammenfassen */
   ObjID            objId;                     /* Objekt-ID für Archivierung */
} tsmRenameIn_t;

#define tsmRenameInVersion 1

/*-------------------------------------------------------------------------+
 |  Typdefinition für dsmRenameOut_t                                       |
 +-------------------------------------------------------------------------*/
typedef struct tsmRenameOut_t
{
   dsUint16_t         stVersion;              /* Strukturversion              */
} tsmRenameOut_t;

#define tsmRenameOutVersion 1

/*-------------------------------------------------------------------------+
 |  Typdefinition für tsmEndSendObjExIn_t                                  |
 +-------------------------------------------------------------------------*/
typedef struct tsmEndSendObjExIn_t
{
   dsUint16_t         stVersion;              /* Strukturversion              */
   dsUint32_t       tsmHandle;                  /* Sitzungskennung */
} tsmEndSendObjExIn_t;

#define tsmEndSendObjExInVersion 1

/*-------------------------------------------------------------------------+
 |  Typdefinition für dsmEndSendObjExOut_t                                 |
 +-------------------------------------------------------------------------*/
typedef struct tsmEndSendObjExOut_t
{
   dsUint16_t         stVersion;              /* Strukturversion              */
   dsStruct64_t       totalBytesSent;      /* Gesamtsumme aus Anwendung gelesener Byte */
   dsmBool_t          objCompressed;       /* mit Objektkomprimierung */
   dsStruct64_t       totalCompressSize;   /* Gesamtgröße nach Komprimierung */
   dsStruct64_t       totalLFBytesSent;    /* Gesamtanzahl LAN-unabhängig gesendeter Byte */
   dsUint8_t          encryptionType;      /* verwendeter Verschlüsselungstyp */
   dsmBool_t          objDeduplicated;     /* Objektverarbeitung für verteilte Deduplizierung */
   dsStruct64_t       totalDedupSize;      /* Gesamtgröße nach Deduplizierung */
}tsmEndSendObjExOut_t;
```

```
#define tsmEndSendObjExOutVersion 3

/*-------------------------------------------------------------------------+
 |  Typdefinition für tsmGroupHandlerIn_t                                  |
 +-------------------------------------------------------------------------*/
typedef struct tsmGroupHandlerIn_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsUint32_t       tsmHandle;                    /* Sitzungskennung */
   dsUint8_t       groupType;        /* Gruppentyp         */
   dsUint8_t        actionType;       /* Gruppenoperationstyp                   */
   dsUint8_t       memberType;       /* Mitgliedstyp: Leiter oder Mitglied */
   dsStruct64_t     leaderObjId;      /* OBJID des Gruppenleiters               */
   dsChar_t         *uniqueGroupTagP; /* Eindeutige Gruppen-ID                  */
   tsmObjName       *objNameP ;       /* Objektname des Gruppenleiters          */
   dsmGetList      memberObjList;    /* Liste der zu enfernenden bzw. zuzuordnenden Objekte */
} tsmGroupHandlerIn_t;

#define tsmGroupHandlerInVersion 1

/*-------------------------------------------------------------------------+
 |  Typdefinition für tsmGroupHandlerExOut_t                               |
 +-------------------------------------------------------------------------*/
typedef struct tsmGroupHandlerOut_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
} tsmGroupHandlerOut_t;

#define tsmGroupHandlerOutVersion 1

/*-------------------------------------------------------------------------+
 |  Typdefinition für tsmEndTxnExIn_t                                      |
 +-------------------------------------------------------------------------*/
typedef struct tsmEndTxnExIn_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsUint32_t       tsmHandle;                    /* Sitzungskennung */
   dsUint8_t        vote;
} tsmEndTxnExIn_t;

#define tsmEndTxnExInVersion 1

/*-------------------------------------------------------------------------+
 |  Typdefinition für tsmEndTxnExOut_t                                     |
 +-------------------------------------------------------------------------*/
typedef struct tsmEndTxnExOut_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsUint16_t         reason;                 /* Ursachencode                  */
   dsStruct64_t       groupLeaderObjId;       /* Gruppenleiterobjekt-ID (für    */
   /* DSM_ACTION_OPEN zurückgegeben) */
    dsUint8_t          reserved1;             /* für zukünftige Verwendung      */
    dsUint16_t         reserved2;             /* für zukünftige Verwendung      */
} tsmEndTxnExOut_t;

#define tsmEndTxnExOutVersion 1

/*-------------------------------------------------------------------------+
 |  Typdefinition für tsmEndGetDataExIn_t                                  |
 +-------------------------------------------------------------------------*/
typedef struct tsmEndGetDataExIn_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsUint32_t       tsmHandle;                    /* Sitzungskennung */
}tsmEndGetDataExIn_t;

#define tsmEndGetDataExInVersion 1
```

```
/*-------------------------------------------------------------------------+
 |  Typdefinition für tsmEndGetDataExOut_t                                  |
 +-------------------------------------------------------------------------*/
typedef struct tsmEndGetDataExOut_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsUint16_t         reason;                 /* Ursachencode                  */
   dsStruct64_t   totalLFBytesRecv;  /* Gesamtzahl LAN-unabhängig empfangener Byte */
}tsmEndGetDataExOut_t;

#define tsmEndGetDataExOutVersion 1

/*-------------------------------------------------------------------------+
 |  Typdefinition für tsmRetentionEvent()                                   |
 +-------------------------------------------------------------------------*/
typedef struct tsmRetentionEventIn_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsUint32_t         tsmHandle;                   /* Sitzungskennung  */
   dsmEventType_t     eventType;                   /* Ereignistyp      */
   dsmObjList_t       objList;                     /* Objekt-ID        */
}tsmRetentionEventIn_t;

#define tsmRetentionEventInVersion 1

/*-------------------------------------------------------------------------+
 |  Typdefinition für tsmRetentionEvent()                                   |
 +-------------------------------------------------------------------------*/
typedef struct tsmRetentionEventOut_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
}tsmRetentionEventOut_t;

#define tsmRetentionEventOutVersion 1

/*-------------------------------------------------------------------------+
 |  Typdefinition für tsmUpdateObjExIn_t                                    |
 +-------------------------------------------------------------------------*/
typedef struct tsmUpdateObjExIn_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
   dsUint32_t         tsmHandle;                   /* Sitzungskennung  */
   tsmSendType     sendType;            /* Sendetyp(Sich./Archiv.) */
   dsChar_t        *descrP;             /* Archivierungsbeschreibung */
   tsmObjName      *objNameP;           /* Objektname          */
   tsmObjAttr      *objAttrPtr;         /* Attribut            */
   dsUint32_t      objUpdAct;           /* Aktualisierungsaktion */
   ObjID           archObjId;           /* Objekt-ID für Archivierung */
}tsmUpdateObjExIn_t;

#define tsmUpdateObjExInVersion 1

/*-------------------------------------------------------------------------+
 |  Typdefinition für tsmUpdateObjExOut_t                                   |
 +-------------------------------------------------------------------------*/
typedef struct tsmUpdateObjExOut_t
{
   dsUint16_t         stVersion;              /* Strukturversion               */
}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-Clientkomponente                                                 *
*                                                                      *
* (C) Copyright IBM Corporation 1993, 2010                             *
***********************************************************************/

/**************************************************************************
* Header File Name: dsmapips.h
*
* Umgebung:         *********************************************
*                   ** Dies ist eine plattformspezifische      **
*                   ** Quellendatei für Windows NT             **
*
*                   *********************************************
*
* Anm. zum Entwurf: Diese Datei enthält plattformabhängige Definitionen
*
* Beschreib. Name:  Definitionen für Tivoli Storage Manager-Typdefinitionen
*                   und LINKAGE
*-------------------------------------------------------------------------*/

#ifndef _H_DSMAPIPS
#define _H_DSMAPIPS


#ifndef _WIN64
#pragma pack(1)
#endif

/*<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>*/
/*                     T Y P D E F I N I T I O N E N                      */
/*<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>*/

/* Neue Datei typedef für Version 3   */


#if !defined(DSMAPILIB) || defined (XOPEN_BUILD)

/* Unterstützung für 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;

/*=== Zeichen- und Zeichenfolgedatentypen ===*/
#ifdef UNICODE
  typedef wchar_t dsChar_t;
  #define dsTEXT(x)       L##x
#else
  typedef char dsChar_t;
  #define dsTEXT(x)       x
#endif  /* !UNICODE */
```

```
/*=== Aus dsChar_t abgeleitete einheitliche Typdefinitionen und Definitionen ===*/
typedef dsChar_t            *dsString_t;

  /* für erweiterte Zurückschreibungsreihenfolge hinzugefügt */
   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;
   /*=== Eine "true" 64-Bit-Ganzzahl ohne Vorzeichen ===*/
   typedef __int64              dsLongLong_t;
#else
typedef struct tagUINT64_t
   {
      dsUint32_t hi;        /* Höchstwertige 32 Bit. */
      dsUint32_t lo;        /* Niedrigstwertige 32 Bit. */
   } dsUint64_t;
#endif

/*-------------------------------------------------------------------------+
| Typdefinition für bool_t                                                 |
+-------------------------------------------------------------------------*/
/*
 *  Es musste ein boolescher Typ erstellt werden, der mit keiner anderen vordefinierten
 *  Version in einem Betriebssystem oder einer Fenstertechnik kollidierte.
 */
typedef enum
{
   dsmFalse = 0x00,
   dsmTrue  = 0x01
}dsmBool_t ;


/*===  für Abwärtskompatibilität ===*/
#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;        /* Höchstwertige 32 Bit. */
   dsUint32_t lo;        /* Niedrigstwertige 32 Bit. */
}dsStruct64_t ;

#endif /* DSMAPILIB */
```

```
#ifndef _WIN64
#pragma pack()
#endif
#endif  /* _H_DSMAPIPS */
/*****************************************************************************
* Tivoli Storage Manager                                                     *
* Allgemeine Quellenkomponente                                               *
*                                                                            *
* (C) Copyright IBM Corporation 1993,2016                                    *
*****************************************************************************/

/*****************************************************************************
* Headerdateiname: release.h
*
* Umgebung:        ***********************************************
*                  ** Dies ist eine plattformunabhängige         **
*                  ** Quellendatei                               **
*                  ***********************************************
*
* Anm. zum Entwurf: Diese Datei enthält allgemeine Informationen zur
*                   aktuellen Version samt Release.Stufe.Unterstufe
*
Beschreib. Name:    Definitionen für Tivoli Storage Manager-Version
*
* Hinweis: Diese Datei sollte keine LOG- oder CMVC-Informationen
*          enthalten. Sie wird mit dem API-Code geliefert.
*
*----------------------------------------------------------------------------*/

#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"

/*===========================================================================
   Die folgenden Zeichenfolgendefinitionen werden für die Versions-
   information verwendet und sollten nicht in dsTEXT oder osTEXT konver-
   tiert werden. Sie werden nur zum Zeitpunkt der Verbindung verwendet.

   Sie werden auch beim Erstellen der JAR-Datei unter UNIX verwendet.
   Siehe 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     ""

/*=== Produktnamendefinitionen ===*/
#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

/*===========================================================================
   Interne Version mit Release und Stufenversion (Build). Sie sollte für
```

```
   Version+Release+PTF eines Produkts immer eindeutig sein.
   Diese Informationen werden in den Dateiattributen und im Datenstrom
   zu Diagnosezwecken aufgezeichnet.
   HINWEIS: MODIFIZIEREN SIE DIESE WERTE NICHT. SIE KÖNNEN LEDIGLICH
            NEUE EINTRÄGE HINZUFÜGEN!
*****************************************************************************/
#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 */
```

Anhang C. Quellendatei mit den API-Funktionsdefinitionen

Dieser Anhang enthält die Headerdatei `dsmapifp.h` mit den Funktionsdefinitionen für die API.

Anmerkung: **`DSMLINKAGE`** ist für jedes Betriebssystem unterschiedlich definiert. Lesen Sie die Definitionen in der Datei `dsmapips.h` für Ihr Betriebssystem.

Die hier bereitgestellten Informationen enthalten eine Zeitpunktkopie der Dateien, die mit der API verteilt werden. Die neueste Version können Sie den Dateien im API-Verteilerpaket entnehmen.

```
/*************************************************************************
* Tivoli Storage Manager                                                 *
* API-Clientkomponente                                                   *
*                                                                        *
* (C) Copyright IBM Corporation 1993,2002                                *
*************************************************************************/

/*************************************************************************/
/* Headerdateiname:     dsmapifp.h                                       */
/*                                                                       */
/* Beschreibender Name: Tivoli Storage Manager API-Funktionsprototypen   */
/*************************************************************************/
#ifndef _H_DSMAPIFP
#define _H_DSMAPIFP

#if defined(__cplusplus)
extern "C" {
#endif

#ifdef DYNALOAD_DSMAPI

/* Funktion wird dynamisch geladen     */
#include "dsmapidl.h"

#else

/* Funktionen werden implizit aus Bibliothek geladen*/

/*=======================================================================*/
/*              A L L G E M E I N E   F U N K T I O N E N                */
/*=======================================================================*/


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 */
```

Dieser Abschnitt enthält die Funktionsdefinition für die API. Es handelt sich um eine Kopie der Headerdatei `tsmapifp.h`.

Anmerkung: **`DSMLINKAGE`** ist für jedes Betriebssystem unterschiedlich definiert. Lesen Sie die Definitionen in der Datei `tsmapips.h` für Ihr Betriebssystem.

```
/*****************************************************************
* Tivoli Storage Manager                                         *
* API-Clientkomponente                                           *
*                                                                *
* (C) Copyright IBM Corporation 1993,2002                        *
*****************************************************************/

/*****************************************************************/
/* Headerdateiname:     tsmapifp.h                               */
```

```
/*                                                                  */
/* Beschreibender Name: Tivoli Storage Manager API-Funktionsprototypen    */
/********************************************************************/
#ifndef _H_TSMAPIFP
#define _H_TSMAPIFP

#if defined(__cplusplus)
extern "C" {
#endif

#ifdef DYNALOAD_DSMAPI

/* Funktion wird dynamisch geladen     */
#include "dsmapidl.h"

#else

/* Funktionen werden implizit aus Bibliothek geladen*/

/*==================================================================*/
/*              A L L G E M E I N E   F U N K T I O N E N           */
/*==================================================================*/


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 */
```

Anhang D. Funktionen zur behindertengerechten Bedienung für die IBM Spectrum Protect-Produktfamilie

Funktionen zur behindertengerechten Bedienung helfen Benutzern mit Behinderungen, wie eingeschränkter Beweglichkeit oder Sehfähigkeit, damit sie informationstechnologische Inhalte erfolgreich verwenden können.

Übersicht

Die IBM Spectrum Protect-Produktfamilie umfasst die folgenden bedeutenden Funktionen zur behindertengerechten Bedienung:

- Bedienung ausschließlich über die Tastatur
- Operationen, die ein Sprachausgabeprogramm verwenden

Die IBM Spectrum Protect-Produktfamilie verwendet den neuesten W3C-Standard WAI-ARIA 1.0(www.w3.org/TR/wai-aria/), um die Einhaltung von US Section 508(www.access-board.gov/guidelines-and-standards/communications-and-it/about-the-section-508-standards/section-508-standards) und der Web Content Accessibility Guidelines (WCAG) 2.0(www.w3.org/TR/WCAG20/) sicherzustellen. Um die Funktionen zur behindertengerechten Bedienung zu nutzen, verwenden Sie das neueste Release Ihres Sprachausgabeprogramms in Verbindung mit dem neuesten Web-Browser, der von diesem Produkt unterstützt wird.

Die Produktdokumentation im IBM Knowledge Center ist für die behindertengerechte Bedienung aktiviert. Eine Beschreibung der Funktionen zur behindertengerechten Bedienung im IBM Knowledge Center finden Sie im Abschnitt 'Accessibility' der IBM Knowledge Center-Hilfe (www.ibm.com/support/knowledgecenter/about/releasenotes.html?view=kc#accessibility).

Navigation mithilfe der Tastatur

Dieses Produkt verwendet Standardnavigationstasten.

Schnittstelleninformationen

In den Benutzerschnittstellen gibt es keine Inhalte, die 2 - 55 Mal in der Sekunde blinken.

Die Webbenutzerschnittstellen basieren auf Cascading Style Sheets, um Inhalte ordnungsgemäß wiederzugeben und um positive Erfahrungen zu ermöglichen. Die Anwendung bietet eine funktional entsprechende Möglichkeit für Benutzer mit eingeschränktem Sehvermögen, um die Systemanzeigeeinstellungen des Benutzers einschließlich des Modus für kontraststarke Anzeige zu verwenden. Sie können die Schriftgröße über die Einstellungen für die Einheit oder für den Web-Browser steuern.

Die Webbenutzerschnittstellen beinhalten WAI-ARIA-Navigationsmarkierungen, mit deren Hilfe Sie schnell zu Funktionsbereichen in der Anwendung navigieren können.

Software anderer Anbieter

Die IBM Spectrum Protect-Produktfamilie enthält bestimmte Software anderer Anbieter, die nicht der IBM Lizenzvereinbarung unterliegt. IBM gibt keine Erklärung zu den Funktionen zur behindertengerechten Bedienung dieser Produkte ab. Wenden Sie sich an den Softwareanbieter, um Informationen zur behindertengerechten Bedienung der Produkte zu erhalten.

Zugehörige Informationen zur behindertengerechten Bedienung

Neben dem standardmäßigen IBM Help-Desk und den Support-Websites bietet IBM einen TTY-Telefonservice für gehörlose oder hörgeschädigte Kunden für den Zugriff auf Vertriebs- und Support-Services:

TTY-Service
800-IBM-3383 (800-426-3383)
(innerhalb von Nordamerika)

Weitere Informationen zum Engagement von IBM im Bereich der behindertengerechten Bedienung finden Sie in IBM Accessibility (www.ibm.com/able).

Bemerkungen

Die vorliegenden Informationen wurden für Produkte und Services entwickelt, die auf dem deutschen Markt angeboten werden. IBM stellt dieses Material möglicherweise auch in anderen Sprachen zur Verfügung. Für den Zugriff auf das Material in einer anderen Sprache kann eine Kopie des Produkts oder der Produktversion in der jeweiligen Sprache erforderlich sein.

Möglicherweise bietet IBM die in dieser Dokumentation beschriebenen Produkte, Services oder Funktionen in anderen Ländern nicht an. Informationen über die gegenwärtig im jeweiligen Land verfügbaren Produkte und Services sind beim zuständigen IBM Ansprechpartner erhältlich. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Services von IBM verwendet werden können. Anstelle der IBM Produkte, Programme oder Services können auch andere, ihnen äquivalente Produkte, Programme oder Services verwendet werden, solange diese keine gewerblichen oder anderen Schutzrechte von IBM verletzen. Die Verantwortung für den Betrieb von Produkten, Programmen und Services anderer Anbieter liegt beim Kunden.

Für in diesem Handbuch beschriebene Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieses Handbuchs ist keine Lizenzierung dieser Patente verbunden. Lizenzanforderungen sind schriftlich an folgende Adresse zu richten (Anfragen an diese Adresse müssen auf Englisch formuliert werden):

IBM Director of Licensing
IBM Europe, Middle East & Africa
Tour Descartes
2, avenue Gambetta
92066 Paris La Defense
France

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden. Die hier enthaltenen Informationen werden in regelmäßigen Zeitabständen aktualisiert und als Neuausgabe veröffentlicht. IBM kann ohne weitere Mitteilung jederzeit Verbesserungen und/oder Änderungen an den in dieser Veröffentlichung beschriebenen Produkten und/oder Programmen vornehmen.

Verweise in diesen Informationen auf Websites anderer Anbieter werden lediglich als Service für den Kunden bereitgestellt und stellen keinerlei Billigung des Inhalts dieser Websites dar. Das über diese Websites verfügbare Material ist nicht Bestandteil des Materials für dieses IBM Produkt. Die Verwendung dieser Websites geschieht auf eigene Verantwortung.

Werden an IBM Informationen eingesandt, können diese beliebig verwendet werden, ohne dass eine Verpflichtung gegenüber dem Einsender entsteht.

Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängig voneinander erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Die Lieferung des in diesem Dokument beschriebenen Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt auf der Basis der IBM Rahmenvereinbarung bzw. der Allgemeinen Geschäftsbedingungen von IBM, der IBM Internationalen Nutzungsbedingungen für Programmpakete oder einer äquivalenten Vereinbarung.

Die in diesem Dokument enthaltenen Leistungsdaten wurden von bestimmten Betriebsbedingungen abgeleitet. Die tatsächlichen Ergebnisse können davon abweichen.

Alle Informationen zu Produkten anderer Anbieter stammen von den Anbietern der aufgeführten Produkte, deren veröffentlichten Ankündigungen oder anderen allgemein verfügbaren Quellen. IBM hat diese Produkte nicht getestet und kann daher keine Aussagen zu Leistung, Kompatibilität oder anderen Merkmalen machen. Fragen zu den Leistungsmerkmalen von Produkten anderer Anbieter sind an den jeweiligen Anbieter zu richten.

Diese Veröffentlichung enthält Beispiele für Daten und Berichte des alltäglichen Geschäftsablaufes. Sie sollen nur die Funktionen des Lizenzprogramms illustrieren; sie können Namen von Personen, Firmen, Marken oder Produkten enthalten. Alle diese Namen sind frei erfunden; Ähnlichkeiten mit tatsächlichen Namen und Adressen sind rein zufällig.

COPYRIGHTLIZENZ:

Diese Veröffentlichung enthält Beispielanwendungsprogramme, die in Quellensprache geschrieben sind und Programmiertechniken in verschiedenen Betriebsumgebungen veranschaulichen. Sie dürfen diese Beispielprogramme kostenlos kopieren, ändern und verteilen, wenn dies zu dem Zweck geschieht, Anwendungsprogramme zu entwickeln, zu verwenden, zu vermarkten oder zu verteilen, die mit der Anwendungsprogrammierschnittstelle für die Betriebsumgebung konform sind, für die diese Beispielprogramme geschrieben werden. Diese Beispiele wurden nicht unter allen denkbaren Bedingungen getestet. Daher kann IBM die Zuverlässigkeit, Wartungsfreundlichkeit oder Funktion dieser Programme weder zusagen noch gewährleisten. Die Beispielprogramme werden ohne Wartung (auf "as-is"-Basis) und ohne jegliche Gewährleistung zur Verfügung gestellt. IBM übernimmt keine Haftung für Schäden, die durch die Verwendung der Beispielprogramme entstehen.

Kopien oder Teile der Beispielprogramme bzw. daraus abgeleiteter Code müssen folgenden Copyrightvermerk beinhalten: © (Name Ihrer Firma) (Jahr). Teile des vorliegenden Codes wurden aus Beispielprogrammen der IBM Corp. abgeleitet. © Copyright IBM Corp. _Jahr/Jahre angeben_.

Marken

IBM, das IBM Logo und ibm.com sind Marken oder eingetragene Marken der IBM Corporation in den USA und/oder anderen Ländern. Weitere Produkt- und Servicenamen können Marken von IBM oder anderen Unternehmen sein. Eine aktuelle Liste der IBM Marken finden Sie auf der Website "Copyright and trademark information" unter www.ibm.com/legal/copytrade.shtml.

Adobe ist eine eingetragene Marke der Adobe Systems Incorporated in den USA und/oder anderen Ländern.

Linear Tape-Open, LTO und Ultrium sind Marken von HP, der IBM Corporation und von Quantum in den USA und/oder anderen Ländern.

Intel und Itanium sind Marken oder eingetragene Marken der Intel Corporation oder ihrer Tochtergesellschaften in den USA und anderen Ländern.

Linux ist eine eingetragene Marke von Linus Torvalds in den USA und/oder anderen Ländern.

Microsoft, Windows und Windows NT sind Marken der Microsoft Corporation in den USA und/oder anderen Ländern.

Java™ und alle auf Java basierenden Marken und Logos sind Marken oder eingetragene Marken der Oracle Corporation und/oder ihrer verbundenen Unternehmen.

SoftLayer ist eine eingetragene Marke von SoftLayer Inc., einem IBM Unternehmen.

UNIX ist eine eingetragene Marke von The Open Group in den USA und anderen Ländern.

Bedingungen für die Produktdokumentation

Die Berechtigungen zur Nutzung dieser Veröffentlichungen werden Ihnen auf der Basis der folgenden Bedingungen gewährt.

Anwendbarkeit

Diese Bedingungen sind eine Ergänzung der Nutzungsbedingungen auf der IBM Website.

Persönliche Nutzung

Sie dürfen diese Veröffentlichungen für Ihre persönliche, nicht kommerzielle Nutzung unter der Voraussetzung vervielfältigen, dass alle Eigentumsvermerke erhalten bleiben. Sie dürfen diese Veröffentlichungen oder Teile der Veröffentlichungen ohne ausdrückliche Genehmigung von IBM nicht weitergeben, anzeigen oder abgeleitete Werke davon erstellen.

Kommerzielle Nutzung

Sie dürfen diese Veröffentlichungen nur innerhalb Ihres Unternehmens und unter der Voraussetzung, dass alle Eigentumsvermerke erhalten bleiben, vervielfältigen, weitergeben und anzeigen. Sie dürfen diese Veröffentlichungen oder Teile der Veröffentlichungen ohne ausdrückliche Genehmigung von IBM außerhalb Ihres Unternehmens nicht vervielfältigen, weitergeben, anzeigen oder abgeleitete Werke davon erstellen.

Rechte

Abgesehen von den hier gewährten Berechtigungen werden keine weiteren Berechtigungen, Lizenzen oder Rechte (veröffentlicht oder stillschweigend) in Bezug auf die Veröffentlichungen oder darin enthaltene Informationen, Daten, Software oder geistiges Eigentum gewährt.

IBM behält sich das Recht vor, die hierin gewährten Berechtigungen nach eigenem Ermessen zurückzuziehen, wenn sich die Nutzung der Veröffentlichungen für IBM als nachteilig erweist oder wenn die obigen Nutzungsbestimmungen nicht genau befolgt werden.

Sie dürfen diese Informationen nur in Übereinstimmung mit allen anwendbaren Gesetzen und Verordnungen, einschließlich aller US-amerikanischen Exportgesetze und Verordnungen, herunterladen und exportieren.

IBM IBM übernimmt keine Gewährleistung für den Inhalt dieser Veröffentlichungen. Diese Veröffentlichungen werden auf der Grundlage des gegenwärtigen Zustands (auf "as-is"-Basis) und ohne eine ausdrückliche oder stillschweigende Gewährleistung für die Handelsüblichkeit, die Verwendungsfähigkeit für einen bestimmten Zweck oder die Freiheit von Rechten Dritter zur Verfügung gestellt.

Hinweise zur Datenschutzrichtlinie

IBM Softwareprodukte, einschließlich Software as a Service-Lösungen ("Softwareangebote"), können Cookies oder andere Technologien verwenden, um Informationen zur Produktnutzung zu erfassen, die Endbenutzererfahrung zu verbessern und Interaktionen mit dem Endbenutzer anzupassen oder zu anderen Zwecken. In vielen Fällen werden von den Softwareangeboten keine personenbezogenen Daten erfasst. Einige der IBM Softwareangebote können Sie jedoch bei der Erfassung personenbezogener Daten unterstützen. Wenn dieses Softwareangebot Cookies zur Erfassung personenbezogener Daten verwendet, sind nachfolgend nähere Informationen über die Verwendung von Cookies durch dieses Angebot zu finden.

Dieses Softwareangebot verwendet keine Cookies oder andere Technologien zur Erfassung personenbezogener Daten.

Wenn die für dieses Softwareangebot genutzten Konfigurationen Sie als Kunde in die Lage versetzen, personenbezogene Daten von Endbenutzern über Cookies und andere Technologien zu erfassen, müssen Sie sich zu allen gesetzlichen Bestimmungen in Bezug auf eine solche Datenerfassung, einschließlich aller Mitteilungspflichten und Zustimmungsanforderungen, rechtlich beraten lassen.

Weitere Informationen zur Nutzung verschiedener Technologien, einschließlich Cookies, für diese Zwecke finden Sie in den Schwerpunkten der IBM Online-Datenschutzerklärung unter http://www.ibm.com/privacy, in der IBM Online-Datenschutzerklärung unter http://www.ibm.com/privacy/details im Abschnitt "Cookies, Web-Beacons und sonstige Technologien" und auf der Seite "IBM Software Products and Software-as-a-Service Privacy Statement" unter http://www.ibm-.com/software/info/product-privacy.

Glossar

Für die IBM Spectrum Protect-Produktfamilie ist ein Glossar mit Begriffen und Definitionen verfügbar.

Siehe das Glossar für IBM Spectrum Protect.

Informationen zum Anzeigen von Glossaren für andere IBM Produkte finden Sie unter IBM Terminologie.

Index

Numerische Stichwörter

64-Bit
  kompilieren 1
  Voraussetzungen 1

A

Abfrage
  actlog 131
  Knoten mit Client-Proxy-Knotenberechtigung 87
Abfragen, System 33
Ablaufdiagramm
  Beispiel für Sicherung und Archivierung 62
  Zurückschreibung und Abruf 75
Abrufen
  Objekte von einem Server 70
Administratoroptionen 2
Aktive Kopien von Objekten 43
Aktive Version
  löschen 79
Anmeldeprozess 17
Anwendungstyp 16, 125, 128
Anwendungsversion vii
API
  dsmInitEx
    Konfigurationsdatei, verwendet von 3
  Musteranwendungen 5
  Übersicht 1
  Umgebungskonfiguration 4
  Unicode verwenden 89
  von dsmInitEx verwendete Optionszeichenfolge 3
API-Konfigurationsdatei
  von dsmInitEx verwendet 16
API-Musteranwendungen
  callbuff 5
  callbuff - Datenpuffer 5
  callevnt 5
  callevnt - ereignisgesteuerte Aufbewahrungsdauer 5
  callhold 5
  callhold - Status 'Löschen unzulässig' 5
  callmt* 5
  callmt* - Multithread-API-Musteranwendungen 5
  callmtu1.c 89
  callmtu2.c 89
  callret 5
  callret - API-Musteranwendungen für Aufbewahrungsschutz für Daten 5
  dapi* 5
  dapi* - interaktive Einzelthread-API-Musteranwendungen 5
  dsmgrp 5
API-Musteranwendungen *(Forts.)*
  dsmgrp* - Muster für Objektgruppierung 5
  UNIX oder Linux 5
  Windows 64-Bit 7
API-Optionsliste
  von dsmInitEx verwendet 16
archiveretentionprotection 32
Archivieren von Objekten 43
Archivierungsdateien
  Aufbewahrungsdauer 28
Archivierungskopiengruppe 28
Archivierungsobjekte
  aussetzen 30
  freigeben 30
  Verfall 30
asnodename 87
Aufbewahrungsschutz 31
Ausschließen von Dateien von der Datendeduplizierung 58
Ausschließen von Objekten 23
Auswählen von Objekten
  für die Zurückschreibung 72
Automatisierte Clientübernahme 59

B

Beenden einer Sitzung 15
  mit dsmTerminate 16
Befehle
  makemtu 89
Behinderung 225
Benutzer
  Eingriff 15
Benutzer mit Verwaltungsaufgaben
  erstellen 21
Berechtigter Benutzer 20, 24
Berechtigungsregel
  dsmDeleteAccess, Funktion 106
Betriebssystem, Interoperabilität 86

C

callbuff
  API-Musteranwendungen für IBM Spectrum Protect-Datenpuffer 5
callevnt
  ereignisgesteuerte Aufbewahrungsdauer 5
callhold
  API-Musteranwendungen für Status 'Löschen unzulässig' 5
callmt*
  Multithread-API-Musteranwendungen 5
callmt1.c
  Muster 14
callret
  API-Musteranwendungen für Aufbewahrungsschutz für Daten 5
capacity
  Dateibereich 25
Client für Sichern/Archivieren
  Interoperabilität 83
Clienteignerberechtigung 21
Clientknoten-Proxy-Unterstützung 87
Clientleistungsüberwachung 40
Clientleistungsüberwachungsoptionen
  PERFCOMMTIMEOUT 42
  PERFMONTCPPORT 41
  PERFMONTCPSERVERADDRESS 41
Clientseitige Datendeduplizierung 55
Codepages 89
compressalways 3
  Option 6

D

dapi*
  interaktive Einzelthread-API-Musteranwendungen 5
Dateibereich
  capacity 25
  Löschen 25
  registrieren 25
  verwalten 25
Dateibereiche
  Nicht-Unicode 90
Dateibereichsname
  Dateizusammenfassung 38
  Übersicht 22
Dateibereichsverwaltung
  dsmUpdateFS 26
Dateien
  Konfigurationsdateien 1
  Objekttyp 23
  Option 1
Dateigruppierung 67
Dateisystemverwaltung
  dsmDeleteFS 26
Dateizusammenfassung 38
Daten senden
  an Nicht-Unicode-Dateibereiche 90
Datenaufbewahrung 32
Datendeduplizierung 52
Datendeduplizierung, Dateien
  ausschließen 58
  einschließen 58
Datenschutz 32
Datenstrukturen
  Größenbeschränkungen 12, 36
  Versionssteuerung 13
Datenübertragung
  LAN-unabhängig 38
DB Chg, Operation 9
DBCS 89
delete archive 85
delete filespace 85
Direktaufruf 34
Direktaufrufabfragen 98
Doppelbytezeichensatz 89

dscenu.txt 4
dsierror.log 4
DSM_MAX_PLATFORM_LENGTH 16
dsm.opt 1
  asnodename, Option 87
  enablearchiveretentionprotection 32
  encryptkey 49
dsm.sys 1, 4, 18
  asnodename, Option 87
  enablearchiveretentionprotection 32
  encryptkey 49
dsmapifp.h
  Headerdatei 93, 215
dsmapips.h, Headerdatei 171
dsmapitd.h 12, 13, 124, 127
  Headerdatei 134
dsmapitd.h, Headerdatei 171
dsmApiVersion, Funktion
  Sitzung 15
dsmBeginGetData, Funktion 70, 71, 74
  Codebeispiel 77
  dsmEndGetData, Funktion 109
  dsmTerminate, Funktion 109
  in Ablaufdiagramm 76
  Pufferverwaltung 47
  Rückkehrcodes 97
  Syntax 96
  Übersicht 96
  Zustandsdiagramm 75, 80
dsmBeginQuery, Funktion
  Abfragebeispiel 35
  abfragen 33
  Ablaufdiagramm 33
  Beispiel für das Senden von Daten 37
  Daten empfangen 71
  dsmEndQuery, Funktion 111
  dsmGetNextQObj, Funktion 118
  Rückkehrcodes 102
  Syntax 97
  Übersicht 97
  Verwaltungsklasse 29
  Zustandsdiagramm 33, 80
dsmBeginTxn 25
dsmBeginTxn, Funktion
  Codebeispiel 65
  dsmEndTxn, Funktion 113
  dsmRenameObj, Funktion 141
  dsmRetentionEvent, Funktion 143
  Löschen 30
  Maßnahme für Aufbewahrungsdauer 33
  Objekte löschen 79
  Pufferkopieneliminierung 46
  Rückkehrcodes 103
  Syntax 103
  Transaktionsmodell 37
  Übersicht 102
  Verfall 30
  Zustandsdiagramm 80
dsmBindMC
  Beispiel 30
dsmBindMC, Funktion
  allgemeine Beschreibung 64
  Codebeispiel 65
  dsmSendObj, Funktion 146
  Einschluss-/Ausschlussliste 28
dsmBindMC, Funktion *(Forts.)*
  Objektnamen 23
  Pufferkopieneliminierung 46
  Rückkehrcodes 104
  Syntax 103
  Übersicht 103
  Verwaltungsklassen 30
  zurückgegebene Informationen 29
  Zustandsdiagramm 80
dsmChangePW
  allgemeine Beschreibung 80
dsmChangePW, Funktion
  Rückkehrcodes 105
  Sitzungssicherheit 17
  Syntax 105
  Übersicht 105
  Zustandsdiagramm 80
dsmCleanUp, Funktion
  dsmSetUp, Funktion 151
  Multithreading 14
  Signale 15
  Syntax 106
  Übersicht 106
dsmclientV3.cat 4
dsmDeleteAccess, Funktion
  Syntax 106
  Übersicht 106
  Zugriff auf Objekte 25
dsmDeleteFS, Funktion
  Dateibereiche 25
  Dateisystemverwaltung 26
  Mustercode 25
  Rückkehrcodes 108
  Syntax 107
  Übersicht 107
  Zustandsdiagramm 80
dsmDeleteObj, Funktion
  dsmEndTxn, Funktion 113
  dsmSendObj, Funktion
    Verwaltungsklasse 9
  Objektbenennung 9
  Objekte 43
  Objekte löschen 79
  Rückkehrcodes 109
  Syntax 108
  Übersicht 108
  Zustandsdiagramm 80
dsmEndGetData
  Prozess stoppen 75
dsmEndGetData, Funktion 71
  Codebeispiel 77
  in Ablaufdiagramm 76
  LAN-unabhängig 38
  Pufferverwaltung 47
  Syntax 109
  Übersicht 109
  Zustandsdiagramm 75, 80
dsmEndGetDataEx, Funktion
  Syntax 110
  Übersicht 110
dsmEndGetObj, Funktion 71
  Codebeispiel 77
  dsmBeginGetData, Funktion 96
  in Ablaufdiagramm 76
  Pufferverwaltung 47
  Rückkehrcodes 111
  Syntax 110
dsmEndGetObj, Funktion *(Forts.)*
  Übersicht 110
  Zustandsdiagramm 75, 80
dsmEndQuery 33
  allgemeine Beschreibung 33
dsmEndQuery, Funktion 35
  Ablaufdiagramm 33
  dsmGetNextQObj, Funktion 118
  Server abfragen 71
  Syntax 111
  Übersicht 111
  Zustandsdiagramm 33, 80
dsmEndSendObj, Funktion
  Ablaufdiagramm 63
  Codebeispiel 65
  dsmEndTxn, Funktion 113
  dsmSendData, Funktion 145
  dsmSendObj, Funktion 146
  Objekte senden 42
  Rückkehrcodes 112
  Syntax 111
  Übersicht 111
  Zustandsdiagramm 62, 80
dsmEndSendObjEx, Funktion 46
  Komprimierung 44
  LAN-unabhängig 38
  Rückkehrcodes 112
  Syntax 112
  Übersicht 112
  Verschlüsselung 49
dsmEndTxn, Funktion 30, 143
  Ablaufdiagramm 63
  Codebeispiel 65
  Dateigruppierung 67
  dsmEndTxnEx, Funktion 114
  dsmRenameObj, Funktion 141
  dsmRetentionEvent, Funktion 143
  dsmSendObj, Funktion 146
  Gleichzeitiges Schreiben 39
  Objekte löschen 79
  Pufferkopieneliminierung 46
  Rückkehrcodes 114
  Syntax 113
  Transaktionsmodell 37
  Übersicht 113
  Zustandsdiagramm 62, 80
dsmEndTxnEx, Funktion
  Dateigruppierung 67
  Rückkehrcodes 115
  Syntax 114
  Übersicht 114
dsmEventType, Funktion
  Maßnahme für Aufbewahrungsdauer 33
dsmGetBufferData, Funktion 47
  Rückkehrcodes 117
  Syntax 117
  Übersicht 117
dsmGetData 75
dsmGetData, Funktion
  Codebeispiel 77
  in Ablaufdiagramm 76
  in Zustandsdiagramm 75
  Rückkehrcodes 116
  Syntax 116
  Übersicht 116
  Zustandsdiagramm 80

dsmGetDataEx, Funktion
dsmReleaseBuffer, Funktion 140
dsmRequestBuffer, Funktion 142
dsmGetList, Funktion
dsmGetObj, Funktion 121
dsmGetNextObj
dsmDeleteObj, Funktion 108
dsmGetNextQObj 33
dsmEndQuery, Funktion 111
dsmGetNextQObj, Funktion 31, 33, 60
Abfragebeispiel 35
Ablaufdiagramm 33
dsmRetentionEvent, Funktion 143
Rückkehrcodes 120
Syntax 118
Übersicht 118
Zustandsdiagramm 33, 80
dsmGetObj
Objekte empfangen 75
dsmGetObj, Funktion 71
Codebeispiel 77
dsmBeginGetData, Funktion 96
dsmEndGetObj, Funktion 110
dsmGetData, Funktion 116
in Ablaufdiagramm 76
Rückkehrcodes 122
Syntax 122
Übersicht 121
Zustandsdiagramm 75, 80
dsmGroupHandler, Funktion
Dateigruppierung 67
dsmEndTxnEx, Funktion 114
Rückkehrcodes 123
Syntax 122
Übersicht 122
dsmgrp*
API-Musteranwendungen für logische Objektgruppierung 5
dsmgrp.c 69
dsmHandle 136, 137
dsmHandle, Funktion
Sitzung 15
DSMI_CONFIG, Umgebungsvariable 4
DSMI_DIR
Umgebungsvariable 6
DSMI_DIR, Umgebungsvariable 4
DSMI_LOG, Umgebungsvariable 4
dsmInit, Funktion
Aufbewahrungsschutz 31
Rückkehrcodes 126
Syntax 124
Übersicht 123
dsmInitEx, Funktion 25, 46
abgelaufenes Kennwort 17
asnodename, Option 87
Aufbewahrungsschutz 31
Benutzer mit Verwaltungsaufgaben 21
dsmChangePW, Funktion 105
dsmEndGetData, Funktion 109
dsmGetBufferData, Funktion 117
dsmGetNextQObj, Funktion 118
dsmLogEvent, Funktion 131
dsmQueryCliOptions, Funktion 135
dsmQuerySessInfo, Funktion 136
dsmQuerySessOptions 137
dsmReleaseBuffer, Funktion 140
dsmInitEx, Funktion *(Forts.)*
dsmSetUp, Funktion 151
Interoperabilität 86
Multithreading 14
Optionen angeben 2
Optionszeichenfolge 3
Rückkehrcodes 130
Sitzung 15
Sitzung starten 15
Sitzungseigner festlegen 24
Sitzungssicherheit 17
Syntax 127
Übersicht 127
Verschlüsselung 49
Zustandsdiagramm 80
dsmLogEvent, Funktion
Rückkehrcodes 132
Syntax 131
Übersicht 131
dsmLogEventEx, Funktion 79
Rückkehrcodes 133
Syntax 132
Übersicht 132
dsmQuery, Funktion
mehrere Knoten 87
dsmQueryAccess, Funktion 25
dsmDeleteAccess, Funktion 106
Übersicht 133
dsmQueryApiVersion, Funktion
Syntax 134
Übersicht 134
Zustandsdiagramm 80
dsmQueryApiVersionEx, Funktion
Syntax 135
Übersicht 135
Versionssteuerung 12
dsmQueryAPIVersionEx, Funktion
Multithreading 14
dsmQueryCliOptions, Funktion
dsmQuerySessOptions 137
Sitzung 15
Syntax 135
Übersicht 135
dsmQuerySessInfo
dsmDeleteFS, Funktion 107
dsmQuerySessInfo, Funktion
allgemeine Beschreibung 16
dsmRetentionEvent, Funktion 143
Rückkehrcodes 137
Syntax 136
Transaktionsmodell 37
Übersicht 136
Zustandsdiagramm 80
dsmQuerySessOptions, Funktion
Syntax 137
Übersicht 137
dsmrc.h
Headerdatei 159
dsmRCMsg, Funktion
Rückkehrcodes 139
Syntax 138
Übersicht 138
dsmRegisterFS, Funktion
Dateibereiche 25
Mustercode 25
Rückkehrcodes 140
Syntax 139
dsmRegisterFS, Funktion *(Forts.)*
Übersicht 139
Zustandsdiagramm 80
dsmReleaseBuffer, Funktion 46, 47
dsmGetBufferData, Funktion 117
dsmReleaseBuffer, Funktion 140
dsmRequestBuffer, Funktion 142
dsmSendBufferData, Funktion 144
Rückkehrcodes 140
Syntax 140
Übersicht 140
dsmRenameObj, Funktion
Rückkehrcodes 142
Syntax 141
Übersicht 141
dsmRequestBuffer, Funktion
Pufferkopieneliminierung 46
Rückkehrcodes 143
Syntax 142
Übersicht 142
dsmRetentionEvent, Funktion
Löschen 30
Maßnahme für Aufbewahrungsdauer 33
Rückkehrcodes 144
Syntax 144
Übersicht 143
Verfall 30
dsmSendBufferData, Funktion
Pufferkopieneliminierung 46
Rückkehrcodes 145
Syntax 145
Übersicht 144
dsmSendData, Funktion
Ablaufdiagramm 63
Codebeispiel 65
dsmEndSendObj, Funktion 111
dsmEndTxn, Funktion 113
dsmSendObj, Funktion 146
Komprimierung 44
Leistung 39
Multithreading 14
Objekte senden 42
Rückkehrcodes 146
Syntax 146
Übersicht 145
Zustandsdiagramm 62, 80
dsmSendObj
Maßnahme für Aufbewahrungsdauer 33
dsmSendObj, Funktion 33
Ablaufdiagramm 63
Codebeispiel 65
dsmEndTxn, Funktion 113
in Zustandsdiagramm 62
Komprimierung 44
Kopiengruppen 29
Maßnahme für Aufbewahrungsdauer 33
Objektbenennung 9
Objekte löschen 79
Objekte senden 42
Sicherungskopiengruppe 29
Syntax 147
Übersicht 146
Zugriff auf Objekte 24
Zustandsdiagramm 80

dsmSendType, Funktion
Objekte aktualisieren 78
dsmSetAccess, Funktion
Rückkehrcodes 150
Syntax 150
Übersicht 150
Zugriff auf Objekte 25
dsmSetUp, Funktion
LAN-unabhängig 9, 38
Multithread 14
Multithreading 14, 38
passwordaccess 20
Syntax 152
Übersicht 151
dsmtca
Versionssteuerung 12
dsmTerminate 75
dsmTerminate, Funktion
allgemeine Beschreibung 16
dsmInit, Funktion 123
dsmReleaseBuffer, Funktion 140
dsmRequestBuffer, Funktion 142
dsmSetUp, Funktion 151
Puffer 46
Pufferkopieneliminierung 46
Signale 15
Sitzung 15
Syntax 153
Übersicht 153
Zustandsdiagramm 80
dsmUpdateFS, Funktion
Dateibereiche 25
Dateibereichsverwaltung 26
Mustercode 25
Rückkehrcodes 154
Syntax 153
Übersicht 153
Zustandsdiagramm 80
dsmUpdateObj, Funktion
Rückkehrcodes 155
Syntax 155
Übersicht 154
Verwaltungsklasse ändern 28
dsmUpdateObject(Ex), Funktion
Objekte aktualisieren 78
dsmUpdateObjEx, Funktion
Rückkehrcodes 158
Syntax 156
Übersicht 156
Verwaltungsklasse ändern 28

E

Eignerberechtigung 21
Eignername 9, 24
NULL 24
Einschließen von Dateien für die Datendeduplizierung 58
Einschließen von Objekten 23
Einschluss-/Ausschluss
Datei 153
Einschluss-/Ausschlussliste 28, 91
Einschränkungen
Multithreading 14
Verschlüsselung und Komprimierung mit Pufferkopieneliminierung 48
Empfangen von Daten von einem Server
allgemeine Beschreibung 70
Prozedur 71
Zurückschreibung oder Abruf von Teilobjekten 70
Empfehlungen
dsmGetObject
großes Datenvolumen 121
HP-Thread-Stack definieren 15
enablearchiveretentionprotection 32
dsm.opt 32
dsm.sys 32
encryptkey 49
Entwurfsempfehlungen 9
envSetUp 152
Ereignis
eventRetentionActivate 33
Ereignisgesteuert
Maßnahme für Aufbewahrungsdauer 33
Ereignisprotokollierung 79
errorlogretention
Verwendung 79
eventRetentionActivate, Ereignis 33

F

fromowner, Option 25
Funktionen zur behindertengerechten Bedienung 225
Funktionsaufrufe
Kurzbeschreibungen 93
Funktionsdefinitionen, API 215, 219

G

Gleichzeitiges Schreiben
Speicherpools 39
Größe von Objekten 42
Größenbeschränkungen
API-Datenstrukturen 12, 36
Größenschätzungen 42
Gruppenleiter 67

H

Headerdatei dsmapips.h 171
Headerdatei dsmapitd.h 171
Headerdatei release.h 171
Headerdatei tsmapitd.h 171
Headerdateien
dsmapifp.h 215
dsmrc.h 159
tsmapifp.h 219
HP-Thread-Stack 15

I

IBM Knowledge Center v
Inaktive Kopien von Objekten 43
Inaktivieren von Objekten 79
InSession, Zustand 131, 132
Interoperabilität
Befehle 85
Benennung von API-Objekten 83
Interoperabilität *(Forts.)*
Betriebssystem 86
Client für Sichern/Archivieren 83
Konventionen
UNIX oder Linux 83
Windows 83
Zugriff auf API-Objekte 83

K

Knoten
Berechtigung 79
eignerübergreifender Zugriff 25
mit Client-Proxy-Unterstützung 87
Namen 9
Verwaltungsklassen abfragen 30
Knotenreplikation 59
Knowledge Center v
Kompatibilität
zwischen API-Versionen 12
Kompilieren
Unicode 89
Komprimierung 44, 70
Komprimierungstyp
LZ4 45
LZW 45
Konfigurationsdatei
API 3
Konfigurationsquellen
Prioritätsreihenfolge 2
Kopiengruppe 28

L

LAN-unabhängig
Datenübertragung 38
dsmEndGetDataEx, Funktion 110
dsmSetUp, Funktion 9
Leistungsaspekte 39
dsmSendData, Funktion 39
Leistungsüberwachung
Client 40
LZ4-Komprimierung 45
LZW-Komprimierung 45

M

makemtu 89
Maßnahme
Maßnahme für Aufbewahrungsdauer 33
Maßnahmen zum Speichern von Daten 28
Mehrbytezeichensatz 89
Metadaten
Objektbenennung 22
Multithreading
Einschränkungen 14
Flag 9
mtflag, Wert 14
Multithread-Option 14
Übersicht 14
Musteranwendung
callmt1.c 14
Mustercode
dsmgrp.c 69

N

Nachrichten
  dsmRCMsg, Funktion 138
NULL
  Sicherungs- oder Archivierungsgruppe 28

O

objectID-Werte 9
Objekt
  Versionssteuerung 43
Objekt-IDs, Übersicht 22
Objektbenennung
  Beispiele nach Betriebssystem 23
  Dateibereichsname 22
  dsmBindMC 23
  Interoperabilität 83
  Objekttyp 23
  übergeordnet
    Objektname 23
  Übersicht 22
  untergeordnet
    Objektname 23
Objekte
  aktive Kopien 43
  aktualisieren 78
  inaktive Kopien 43
  inaktivieren 79
  löschen 78
  Verfallszyklus 79
  vom Server löschen 79
  Zugriffsregeln 24
Objekttypen 23
Optionen
  compressalways 3
  enablearchiveretentionprotection 32
  errorlogretention 79
  fromnode 25
  fromowner 25
  in API nicht unterstützt 1
  passwordaccess 14, 123
  servername 3
  tcpbuffsize 39
  tcpnodelay 39
  tcpserveraddr 3
  vom Administrator definiert 2
Optionsdateien
  Benutzer 3
Optionsliste
  Format 125, 129
Optionszeichenfolge
  API 3
  fromowner 25

P

passwordaccess
  generate 153
  Option 7, 9, 49
passwordaccess, Option
  dsmInit, Funktion 123
  generate 17
  Multithreading 14
  ohne TCA 20
  userNamePswd, Wert 21
passwordaccess prompt 17
passworddir, Option
  in dsm.sys 20
PERFMONCOMMTIMEOUT 42
PERFMONTCPPORT 41
PERFMONTCPSERVERADDRESS 41
Pfadbeispiele
  nach Betriebssystem 23
Pfadinformationen
  Interoperabilität 83
Protokollierung von Ereignissen 79
proxynode 87
Pufferkopieneliminierung
  Übersicht 46
  Zurückschreibung und Abruf 47

Q

qMCData, Struktur 35
qryRespArchiveData 31
qryRespBackupData
  dsmDeleteObj, Funktion 108
qryRespBackupData, Struktur 33
query
  Befehl 85

R

rcApiOut
  Beispiel, Details 16
rcApiOut, Funktion
  Sitzung 15
Registrieren von Dateibereichen 25
release.h, Headerdatei 171
Replikationsstatus 60
restore 85
retrieve 85
Rückkehrcodes
  Quellenheaderdatei 159
  über dsmRCMsg abrufen 138

S

Senden von Daten an einen Server 37
Server
  Objekte vom Server löschen 79
servername 3
Serverseitige Datendeduplizierung 59
set access 85
Sicherheit 17
Sichern
  mehrere Knoten 87
  mithilfe von Clientknoten-Proxy 87
Sichern von Objekten 43
Sicherungskopiengruppe 28
Signale verwenden 15
Signalhandler 15
Sitzung
  Kennwort
    Sitzung 17
  mit dsmInitEx starten 15
  Sicherheit 17
Sortieren von Objekten
  nach Zurückschreibungsreihenfolge 72
Speicherpools
  gleichzeitiges Schreiben 39
Starten einer Sitzung 15
Stoppen einer Sitzung 15
STRG+C 15
Struktur
  qryRespBackupData 33
  qryRespFSData, Funktion 25
Strukturen
  Größenbeschränkungen 12, 36
  qMCData 35
Systemabfragen 33

T

Tastatur 225
TCA
  ohne passwordaccess 20
  Signale 15
  Sitzungssicherheit 17
  Versionssteuerung 12
TCPport 18
TCPserver-Adresse 18
tcpserveraddr 3
Transaktionsmodell
  dsmBeginTxn, Funktion 102
Trusted Communication Agent
  passwordaccess 20
  Signale 15
  Sitzungssicherheit 17
tsmapifp.h 89
tsmapifp.h, Headerdatei 219
tsmapitd.h 89
tsmapitd.h, Headerdatei 171

U

Übergeordnete Namen
  dsmRenameObj, Funktion 141
Übergeordnetes Qualifikationsmerkmal 83
Übernahme
  Statusinformationen 60
  Übersicht 59
Umgebung
  API-Umgebung konfigurieren 4
Umgebungsvariablen
  DSMI_CONFIG 4
  DSMI_DIR 4
  DSMI_LOG 4
  nach Betriebssystem 4
Unicode
  konfigurieren 89
  Mehrbytezeichensatz 89
  Nicht-Unicode-Dateibereiche 90
  Windows 89
UNIX oder Linux
  API-Musteranwendung 5
Untergeordnete Namen
  dsmRenameObj, Funktion 141
Untergeordnetes Qualifikationsmerkmal 83
Unterstützung für 128–Bit-AES-Verschlüsselung 49
Unterstützung für 256-Bit-AES-Verschlüsselung 49

V

Veröffentlichungen v
Verschlüsselung
  anwendungsverwaltete 49
  Authentifizierungseinstellung 49
  Interoperabilität 86
  transparente 51
Verschlüsselung und Komprimierung mit Pufferkopieneliminierung 48
Versionen
  aufbewahrte Dateien 28
Versionssteuerung
  API-Datenstrukturen 13
  dsmQueryApiVersionEx verwenden 12
  gesicherte Kopien verwalten 43
Verwaltungsklasse
  abfragen 30
  Binden und erneutes Binden an Dateien 29
  dsmBindMC, Zuordnung durch 28
  Objekte zuordnen 28
Verzeichnis
  Objekttyp 23

W

Windows 64-Bit
  Musteranwendung 7

Z

Zeichensätze 89
Zielknoten und traditionelle Knoten 87
Zugriff auf Objekte
  knotenübergreifend 25
  nach Benutzer 24
Zurückschreiben
  Objekte von einem Server 70
Zurückschreibung oder Abruf von Teilobjekten 70
Zustand
  InSession 132
Zustandsdiagramm
  Beispiel für Sicherung und Archivierung 62
  Zurückschreibung und Abruf 75

Programmnummer: 5725-W98
5725-W99
5725-X15