

IBM Spectrum Protect
8.1.12

アプリケーション・プログラミング・イ
ンターフェースの使用



お願い

本書および本書で紹介する製品をご使用になる前に、[201 ページの『特記事項』](#)に記載されている情報をお読みください。

本書は、IBM Spectrum® Protect (製品番号 5725-W98、5725-W99、5725-X15) のこのエディションは、バージョン 8、リリース 1、モディフィケーション 12 に適用されます。および新しい版で明記されていない限り、それ以降のすべてのリリースおよびモディフィケーションに適用されます。

© Copyright International Business Machines Corporation 1993, 2021.

目次

本書について.....	vii
本書の対象読者	vii
資料.....	vii
本書で使用する規則.....	vii
バージョン 8.1.12 の新機能.....	ix
第 1 章 API 概要.....	1
構成ファイルとオプション・ファイルの理解.....	1
API 環境のセットアップ.....	3
第 2 章 API サンプル・アプリケーションの作成と実行.....	5
UNIX または Linux のサンプル・アプリケーション・ソース・ファイル.....	5
UNIX または Linux のサンプル・アプリケーションの作成.....	6
Windows 64 ビット・サンプル・アプリケーション.....	7
第 3 章アプリケーションの設計に関する考慮事項.....	11
サイズ限界の決定.....	14
API バージョン管理の保守.....	15
マルチスレッド化の使用.....	16
シグナルおよびシグナル・ハンドラー.....	17
セッションの開始または終了.....	17
セッション・セキュリティ.....	18
パスワード・ファイルへのアクセスの制御.....	21
クライアント所有者権限を持つ管理ユーザーの作成.....	22
オブジェクト名とオブジェクト ID.....	23
ファイル・スペース名.....	23
高位名と低位名.....	24
オブジェクト・タイプ.....	24
セッション所有者としてのオブジェクトのアクセス	24
ノード間および所有者間のオブジェクトのアクセス.....	25
ファイル・スペースの管理.....	25
オブジェクトと管理クラスとの関連付け.....	28
照会管理クラス.....	29
保留と保留解除の期限切れ/削除.....	30
アーカイブ・データ保存保護.....	31
IBM Spectrum Protect システムの照会.....	33
システム照会の例.....	35
サーバー効率.....	36
サーバーへのデータの送信.....	36
トランザクション・モデル.....	36
ファイルの集約.....	37
LAN フリー・データ転送.....	37
同時書き込み操作.....	37
API のパフォーマンスの向上.....	38
パフォーマンス・データを送信するための API の設定.....	38
クライアント・パフォーマンス・モニター・オプションの構成.....	39
サーバーへのオブジェクトの送信.....	41
バックアップ・オブジェクトおよびアーカイブ・オブジェクトの把握.....	41
圧縮.....	42

バッファ・コピー除去.....	44
API 暗号化.....	46
データ重複排除.....	49
API クライアント・サイド・データ重複排除.....	50
サーバー・サイドのデータ重複排除.....	53
アプリケーション・フェイルオーバー.....	53
フェイルオーバー状況情報.....	54
バックアップおよびアーカイブのフロー・ダイアグラムの例.....	56
IBM Spectrum Protect ストレージにデータを送信する API 機能のコーディング例.....	59
ファイルのグループ化.....	61
サーバーからのデータの受信.....	63
部分オブジェクト・リストア/リトリブ.....	64
データのリストア/リトリブ.....	64
リストアおよびリトリブのフロー・ダイアグラムの例.....	68
サーバーからデータを受信するコーディング例.....	70
サーバー上のオブジェクトの更新および削除.....	71
サーバーからのオブジェクトの削除.....	72
イベント・ロギング.....	72
IBM Spectrum Protect API の状態遷移図の要約.....	72
第 4 章インターオペラビリティについて.....	75
バックアップ/アーカイブ・クライアントのインターオペラビリティ.....	75
API オブジェクトの命名.....	75
API で使用できるバックアップ/アーカイブ・クライアント・コマンド.....	77
オペレーティング・システムのインターオペラビリティ.....	78
クライアント・ノード・プロキシ・サポートを使用した複数ノードのバックアップ.....	78
第 5 章ユニコードでの API の使用.....	81
ユニコードを使用する場合.....	81
ユニコードのセットアップ.....	81
第 6 章 API 関数呼び出し.....	83
dsmBeginGetData.....	85
dsmBeginQuery.....	87
dsmBeginTxn.....	91
dsmBindMC.....	92
dsmChangePW.....	93
dsmCleanUp.....	94
dsmDeleteAccess.....	95
dsmDeleteFS.....	95
dsmDeleteObj.....	96
dsmEndGetData.....	98
dsmEndGetDataEx.....	98
dsmEndGetObj.....	99
dsmEndQuery.....	99
dsmEndSendObj.....	100
dsmEndSendObjEx.....	100
dsmEndTxn.....	101
dsmEndTxnEx.....	103
dsmGetData.....	104
dsmGetBufferData.....	105
dsmGetNextQObj.....	106
dsmGetObj.....	109
dsmGroupHandler.....	110
dsmInit.....	111
dsmInitEx.....	115
dsmLogEvent.....	119

dsmLogEventEx.....	119
dsmQueryAccess.....	121
dsmQueryApiVersion	121
dsmQueryApiVersionEx.....	122
dsmQueryCliOptions.....	122
dsmQuerySessInfo.....	123
dsmQuerySessOptions.....	124
dsmRCMsg.....	125
dsmRegisterFS	126
dsmReleaseBuffer	127
dsmRenameObj.....	128
dsmRequestBuffer.....	129
dsmRetentionEvent.....	130
dsmSendBufferData.....	131
dsmSendData	132
dsmSendObj	134
dsmSetAccess.....	137
dsmSetUp.....	138
dsmTerminate.....	140
dsmUpdateFS	140
dsmUpdateObj.....	141
dsmUpdateObjEx.....	142
付録 A API 戻りコード・ソース・ファイル: dsmrc.h.....	145
付録 B API タイプ定義ソース・ファイル.....	155
付録 C API 関数定義ソース・ファイル.....	191
付録 D アクセシビリティー	199
特記事項.....	201
用語集.....	205
索引.....	207

本書について

本書には、以下の作業の実施に役立つ情報が収められています。

- 既存のアプリケーションに対するすべての IBM Spectrum Protect アプリケーション・プログラミング・インターフェース呼び出し
- IBM Spectrum Protect のサービスを獲得する汎用プログラム・インターフェースによる プログラムの作成

アプリケーション・プログラミング・インターフェース (API) に加えて、いくつかのオペレーティング・システムには、以下のプログラムも含まれています。

- バックアップ/アーカイブ・クライアント・プログラム。これは、ワークステーションまたはファイル・サーバー からストレージにファイルをバックアップまたはアーカイブし、ファイルのバックアップ・バージョンとアーカイブ・コピーを使用するローカル・ファイル・システムにリストア、およびリトリブします。
- Web バックアップ/アーカイブ・クライアント。これによって許可された管理者、サポート担当者、またはエンド・ユーザーは、リモート・マシン上の Web ブラウザーを使用するバックアップ、リストア、アーカイブ、およびリトリブのサービスを実行することができます。
- 管理クライアント・プログラム。これは、Web ブラウザーまたはコマンド・ライン からアクセスできます。このプログラムによって、管理者は、サーバーの活動を 制御およびモニターし、バックアップ、アーカイブ、およびスペース管理サービスのためのストレージ管理ポリシーを定義し、さらにこれらのサービスを定期的に 実行するスケジュールをセットアップできるようになります。

本書の対象読者

本書には、ユーザーが既存のアプリケーションに対して API 呼び出しを 追加する方法についての説明が記載されています。C プログラミング言語および IBM Spectrum Protect 機能に関する知識が必要です。

資料

IBM Spectrum Protect 製品ファミリーには、IBM Spectrum Protect Plus、IBM Spectrum Protect for Virtual Environments、IBM Spectrum Protect for Databases、およびその他の IBM® のストレージ管理製品が含まれています。

IBM 製品の資料については、[IBM Knowledge Center](#) を参照してください。

本書で使用される規則

本書では、以下の書体の規則を使用します。

例	説明
autoexec.ncf hsmgui.exe	拡張子の付いた一連の小文字は、プログラム・ファイル名を示します。
DSMI_DIR	一連の大文字は、戻りコードおよびその他の値を示します。
dsmQuerySessInfo	太字体は、コマンド・ラインに入力するコマンド、関数呼び出しの名前、あるいは構造体、構造体内のフィールド、またはパラメーターの名前を示します。
<i>timeformat</i>	太字イタリック体は、バックアップ/アーカイブ・クライアントのオプションを示します。太字体は、オプションの紹介に使用されるか、または例の中で使用されます。

例	説明
<i>dateformat</i>	イタリック体は、オプション、オプションの値、新規用語、指定する情報のプレースホルダー、またはテキスト内の特殊な強調のためのプレースホルダーを示します。
<code>maxcmdretries</code>	モノスペース体は、プログラム、または画面に表示される情報 (例えば、コマンド例) の一部を示します。
正符号 (+)	2 つのキーの間の正符号は、両方のキーを同時に押すことを示します。

バージョン 8.1.12 の新機能

IBM Spectrum Protect バージョン 8.1.12 では新機能と更新が導入されています。

このリリースの新機能および更新情報のリストについては、[API 更新](#)を参照してください。

この製品資料で新規の情報および変更された情報は、変更箇所の左側に縦棒 (|) を付けて示しています。

第 1 章 API 概要

IBM Spectrum Protect (TSM) のアプリケーション・プログラミング・インターフェース (API) によって、アプリケーション・クライアントはストレージ管理機能を使用できるようになります。

この API には、アプリケーションで次のような操作を実行するとき使用できる関数呼び出しが組み込まれています。

- セッションを開始または終了する
- サーバーに保管する前の、管理クラスのオブジェクトへの割り当て
- オブジェクトをサーバーにバックアップまたはアーカイブする
- オブジェクトをサーバーからリストアまたはリトリートする
- 保管されたオブジェクトに関する情報についてサーバーに照会する
- ファイル・スペースを管理する
- 保存イベントを送信する

アプリケーション開発者として API をインストールすると、アプリケーションのエンド・ユーザーに必要な以下のファイルを受け取ります。

- API 共用ライブラリー。
- メッセージ・ファイル。
- サンプル・クライアント・オプション・ファイル。
- アプリケーションに必要な API ヘッダー・ファイル用のソース・コード
- サンプル・アプリケーション用のソース・コードおよびそれを作成する MAKE ファイル

64 ビット・アプリケーションの場合、コンパイルは必ず、64 ビット・サポートを有効にするコンパイラー・オプションを使用して実行する必要があります。例えば、AIX® で API アプリケーションを作成する場合は '-q64' を使用し、Linux® で作成する場合は '-m64' を使用します。詳しくは、サンプル MAKE ファイルを参照してください。

重要: API をインストールするときは、すべてのファイルが同じレベルであることを確認してください。

API のインストールについては、[IBM Spectrum Protect バックアップ/アーカイブ・クライアントのインストール](#)を参照してください。

本書では、「UNIX」および「Linux」は AIX、HP-UX、Linux、Mac OS X、および Oracle Solaris のオペレーティング・システムを指しています。

構成ファイルとオプション・ファイルの理解

構成ファイルとオプション・ファイルによって、ユーザーのセッションを実行するときの条件および境界を設定します。

管理者またはエンド・ユーザーは、以下の目的でオプションの値を設定できます。

- サーバーへの接続をセットアップするため
- どのオブジェクトをサーバーに送信するか、およびそれらのオブジェクトをどの管理クラスに関連付けるかを制御するため

使用するワークステーションに API をインストールするときに、1 つまたは 2 つのファイルにオプションを定義します。

UNIX および Linux オペレーティング・システムでは、オプションは次の 2 つのオプション・ファイルに入っています。

- dsm.opt - クライアント・オプション・ファイル
- dsm.sys - クライアント・システム・オプション・ファイル

その他のオペレーティング・システムでは、クライアント・オプション・ファイル (dsm.opt) にすべてのオプションが入っています。

制約事項: API は、以下のバックアップ/アーカイブ・クライアント・オプション はサポートしません。

- autofsrename
- changingretries
- domain
- eventlogging
- groups
- subdir
- users
- virtualmountpoint

オプションはまた、**dsmInitEx** 関数呼び出しにも指定できます。オプション・ストリング・パラメーターまたは API 構成ファイル・パラメーターを使用してください。

同じオプションが複数の構成ソースで指定されている可能性があります。この場合は、最も優先順位の高いソースが優先します。[2 ページの表 1](#) は、優先順位のリストです。

表 1. 構成ソース (高い優先順位から順に)

優先度	UNIX および Linux	Windows	説明
1	dsm.sys ファイル (クライアント・システム・オプション)	適用外	<p>このファイルには、UNIX および Linux 用にのみ、システム管理者が設定するオプションが含まれます。</p> <p>ヒント: dsm.sys ファイルに、複数のサーバー・スタンザが含まれている場合、各スタンザの passwordaccess オプションを同じ値 (prompt または generate) に設定するようにしてください。</p>
2	オプション・ストリング (クライアント・オプション)	オプション・ストリング (すべてのオプション)	<p>これらのオプションのうちの 1 つが、パラメーターとして dsmInitEx 呼び出しに渡されると、有効になります。このリストには、compressalways、servername (UNIX および Linux のみ)、または tcpserveraddr (UNIX 以外) などのクライアント・オプションを入れることができます。</p> <p>API オプション・ストリングによって、アプリケーション・クライアントが、API 構成ファイルおよびクライアント・オプション・ファイル内のオプションの値を変更できるようになります。例えば、ユーザーのアプリケーションがエンド・ユーザーに対して圧縮が必要かどうかを照会するとします。ユーザーの応答に基づいて、このオプションを指定した API オプション・ストリングを構成し、それを dsmInitEx 呼び出しに渡すことができます。</p> <p>API オプション・ストリングのフォーマットについては、115 ページの『dsmInitEx』を参照してください。このパラメーターは、NULL に設定することもできます。これはこのセッションには API オプション・ストリングがないことを示します。</p>

表 1. 構成ソース (高い優先順位から順に) (続き)

優先度	UNIX および Linux	Windows	説明
3	API 構成ファイル (クライアント・オプション)	API 構成ファイル (すべてのオプション)	API 構成ファイルの中で設定した値は、クライアント・オプション・ファイルの中で設定した値を指定変更します。API 構成ファイル内のオプションに、ユーザーの IBM Spectrum Protect セッションに適切な値を設定します。その値は、API 構成ファイルの名前が dsmInitEx 呼び出しのパラメーターとして渡されたときに有効になります。 このパラメーターは、NULL に設定することもできます。これはこのセッションには API 構成ファイルがないことを示します。
4	dsm.opt ファイル (dsm.opt file) (クライアント・オプション)	dsm.opt ファイル (すべてのオプション)	UNIX および Linux オペレーティング・システムでは、dsm.opt ファイルにはユーザー・オプションのみが入ります。その他のオペレーティング・システムでは、すべてのオプションが dsm.opt ファイルに入ります。これらのファイルの中のオプションを指定変更するには、この表に説明してある方法に従ってください。

関連概念

[処理オプション](#)

API 環境のセットアップ

API は、ファイルを探し出すために固有な環境変数を使用します。これによって、バックアップ/アーカイブ・クライアントが使用するファイルとは異なるファイルを、API アプリケーションに対して使用できるようになります。各アプリケーションは **dsmSetup** 関数呼び出しを使用して環境変数によって設定された値を指定変更することができます。

ヒント: Windows では、デフォルトのインストール・ディレクトリーは次のようになります。

%SystemDrive%\Program Files\Common Files\Tivoli\TSM\api

3 ページの表 2 は、オペレーティング・システムごとの API 環境変数をリストしたものです。

表 2. API 環境変数

変数	UNIX および Linux	Windows
DSMI_CONFIG	クライアント・オプション・ファイル (dsm.opt) の完全修飾名。	クライアント・オプション・ファイル (dsm.opt) の完全修飾名。
DSMI_DIR	dsm.sys、en_US サブディレクトリー、およびその他の 各国語サポート (NLS) 言語を含むパスを指す。en_US サブディレクトリーには、dsmclientV3.cat が入っている必要がある。	dscenu.txt および NLS メッセージ・ファイルを含むパスを指す。

表 2. API 環境変数 (続き)

変数	UNIX および Linux	Windows
DSMI_LOG	dsierror.log ファイルのためのパスを指す。	dsierror.log ファイルのためのパスを指す。 クライアント errorlogname オプションが設定されている場合、そのオプションが指定する場所は、DSMI_LOG が指定するディレクトリーをオーバーライドします。

第 2 章 API サンプル・アプリケーションの作成と実行

API パッケージには、状況に応じた API 関数呼び出しを示すサンプル・アプリケーションが組み込まれています。関数呼び出しの使用方法についての理解を深めるためにサンプル・アプリケーションをインストールして、そのソース・コードを調べてください。

次の API サンプル・アプリケーション・パッケージの 1 つを選択してください。

- 対話式の単一スレッド・アプリケーション・パッケージ (dapi*)
- マルチスレッド・アプリケーション・パッケージ (callmt*)
- 論理オブジェクトのグループ化テスト・アプリケーション (dsmgrp*)
- イベント・ベースの保存ポリシー・サンプル・アプリケーション (callevnt)
- 削除保留サンプル・アプリケーション (callhold)
- データ保存保護サンプル・アプリケーション (callret)
- IBM Spectrum Protect データ・バッファ・サンプル・プログラム (callbuff)

開始できるように、ご使用の以下のプラットフォームで、dapismp サンプル・アプリケーションを作成する手順を検討します。

- UNIX または Linux アプリケーションの場合は、[5 ページの『UNIX または Linux のサンプル・アプリケーション・ソース・ファイル』](#)を参照してください。
- Windows アプリケーションの場合、[7 ページの『Windows 64 ビット・サンプル・アプリケーション』](#)を参照してください。

dapismp サンプル・アプリケーションは、オブジェクトのバックアップまたは アーカイブの際に独自のデータ・ストリームを作成します。ローカル・ディスク・ファイル・システム からオブジェクトを読み取ったり、ローカル・ディスク・ファイル・システムに オブジェクトを書き込んだりすることはありません。オブジェクト名はワークステーション上の どのファイルにも対応しません。実行した「シード・ストリング」によって、オブジェクトの リストアまたはリトリブ時に検査が可能なパターンが生成されます。サンプル・アプリケーションをコンパイルし、**dapismp** を実行してサンプル・アプリケーションを開始した後は、画面に表示 される指示に従ってください。

UNIX または Linux のサンプル・アプリケーション・ソース・ファイル

UNIX または Linux のサンプル・アプリケーションを作成して実行するには、特定のソース・ファイルが確実に存在する必要があります。サンプル・アプリケーションが作成されれば、それをコンパイルして実行することができます。

5 ページの表 3 に示されているファイルには、API パッケージに組み込まれているサンプル・アプリケーションの作成に必要な ソース・ファイルおよびその他のファイルが含まれています。

表 3. UNIX または Linux の API サンプル・アプリケーションを作成するのに必要なファイル

ファイル名	説明
README_api_enu	README ファイル
dsmrc.h	戻りコード・ヘッダー・ファイル
dsmapi.h	共通タイプ定義ヘッダー・ファイル
dsmapi.h	オペレーティング・システム固有のタイプ
dsmapi.h	定義ヘッダー・ファイル
release.h	関数プロトタイプ・ヘッダー・ファイル
	リリース値のヘッダー・ファイル

表 3. UNIX または Linux の API サンプル・アプリケーションを作成するのに必要なファイル (続き)

ファイル名	説明
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 コマンド・ライン指向のサンプル・アプリケーション用のモジュール
makesmp[64].xxx	ご使用のオペレーティング・システム用の dapismp を作成する MAKE ファイル xxx はオペレーティング・システム を示します。
callmt1.c callmt2.c	マルチスレッドのサンプル・ファイル
callmtu1.c callmtu2.c	マルチスレッドのユニコード・サンプル・ファイル
libApiDS.xx libApiDS64.xx または libApiTSM64.xx	共用ライブラリー (接尾部はプラットフォームによって異なる)
dsmgrp.c callevnt.c callhold.c callret.c callbuff.c dpstthread.c	グループ化のサンプル・ファイル イベント・ベースの保存ポリシーの サンプル・ソース・コード 削除保留のサンプル・ソース・コード データ保存保護のサンプル・ソース・コード

UNIX または Linux のサンプル・アプリケーションの作成

ご使用のオペレーティング・システムのコンパイラーを使用して、**dapismp** API サンプル・アプリケーションを作成します。

このタスクについて

UNIX または Linux の API サンプル・アプリケーションを作成するには、以下のコンパイラーをインストールする必要があります。

- IBM AIX - IBM Visual Age コンパイラー、バージョン 6 以降
- HP-IA64 - aCC コンパイラー A.05.50 以降
- Linux - GCC コンパイラー、バージョン 3.3.3 以降
- Mac OS X - GCC コンパイラー、バージョン 4.0 以降
- Oracle Solaris - Oracle Studio C++ コンパイラー、バージョン 11 以降

手順

1. API サンプルを作成するには、次のコマンドを実行します。

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


ここで、xxx はオペレーティング・システムを示しています。

2. サンプルの作成後に、環境変数 (DSMI_DIR を含む)、およびオプション・ファイルをセットアップします。詳しくは、[1 ページの『構成ファイルとオプション・ファイルの理解』](#)を参照してください。
3. 初めてログオンするときは、root ユーザーとしてログオンし、パスワードを登録します。

ヒント: compressalways オプションを no に設定すると、解凍されたオブジェクトが再送されない場合があります。この性質はアプリケーションの機能に応じて異なります。

AIX で共有メモリー通信方式を指定するには、IBM Spectrum Protect API クライアント・ユーザーは以下のいずれかの条件に従う必要があります。

- root ユーザーとしてログインする必要があります。
- IBM Spectrum Protect サーバーを実行しているプロセスと同じ UID を持っている必要があります。

詳しくは、アプリケーション・プログラムの資料を参照してください。

4. **dapismp** コマンドを実行して、アプリケーションを開始します。
5. 表示されるオプションのリストから、選択を行います。必ず、サインオン・アクションを実行した後、その他のアクションを実行してください。

要件: 名前を入力するときは必ず、ファイル・スペース名、高位名、および低位名の前に正しいパス区切り文字 (/) を接頭部として付けてください。例: /myfilespace。この接頭部は、アスタリスク (*) のワイルドカード文字を指定する場合でも使用する必要があります。

関連概念

[環境変数 \(UNIX および Linux システム\)](#)

Windows 64 ビット・サンプル・アプリケーション

Microsoft Windows 64 ビット・システム用のサンプル・アプリケーションを作成して実行するには、IBM Spectrum Protect API をインストールする必要があり、特定のソース・ファイルが確実に存在している必要があります。

制約事項:

- 最良の結果を得るには、動的ロードを使用します。例えば、サンプル・コードの dynaload.c ファイルと実装を参照してください。
- サンプル・アプリケーションのファイルは、以下のディレクトリーに入っています。

api64¥obj

API サンプル・プログラムのオブジェクト・ファイルが入っています。

api64¥samprun

サンプル・プログラム **dapismp** が入っています。このサンプル・プログラムには、実行ディレクトリーが含まれています。

- DLL の tsmapi64.dll は、64 ビットの DLL です。
- API サンプル・アプリケーション **dapismp** をコンパイルするには、Microsoft C/C++ コンパイラーのバージョン 15、および MAKE ファイル makesmp64.mak を使用します。MAKE ファイルは、ご使用の環境に合わせて調整が必要になる場合があります (特に、ライブラリーまたは組み込みディレクトリー)。
- サンプル・アプリケーションをコンパイル後に実行するには、api64¥samprun ディレクトリーから **dapismp** コマンドを発行します。
- 表示されるオプションのリストから、選択を行います。必ず、サインオン・アクションを実行した後に、その他のアクションを実行してください。
- 名前を入力するときは必ず、ファイル・スペース名、高位名、および低位名の前に正しいパス区切り文字 (¥) を接頭部として付けてください。例: ¥myfilespace。この接頭部は、アスタリスク (*) のワイルドカード文字を指定する場合でも使用する必要があります。

Windows オペレーティング・システムの場合、サンプル・アプリケーションの作成に必要なソース・ファイルは、[8 ページの表 4](#) にリストされています。サンプル・アプリケーションは、API パッケージに組

み込まれています。便宜上、プリコンパイル済みの実行可能ファイル (dapismp.exe) も組み込まれています。

表 4. Windows 64 ビット API サンプル・アプリケーションを作成するためのファイル

ファイル名	説明
api.txt	README ファイル
tsmapi64.dll	API DLL
dsmrc.h	戻りコード・ヘッダー・ファイル
dsmapi64.h	共通タイプ定義ヘッダー・ファイル
dsmapi64.h	オペレーティング・システム固有のタイプ定義 ヘッダー・ファイル
dsmapi64.h	関数プロトタイプ・ヘッダー・ファイル
dsmapi64.h	動的にロードされた関数プロトタイプ・ ヘッダー・ファイル
release.h	リリース値のヘッダー・ファイル
dapidata.h dapint64.h dapitype.h dapiutil.h	ソース・コード・ヘッダー・ファイル
tsmapi64.lib	暗黙ライブラリー
dapibkup.c dapiinit.c dapint64.c dapipref.c dapiproc.c dapiproc.h dapipw.c dapiqry.c dapirc.c dapismp64.c dapiutil.c dynaload.c	dapismp.exe のソース・コード・ファイル
makesmpx64.mak (Windows x64) makesmp64.mak (Windows IA64)	サンプル・アプリケーションを作成する MAKE ファイル
callmt1.c callmt2.c callmtu164.c callmtu264.c	マルチスレッドのサンプル・ファイル
dpsthread.c	サンプル・ファイルのソース・コード

表 4. Windows 64 ビット API サンプル・アプリケーションを作成するためのファイル (続き)

ファイル名	説明
callevnt.c	イベント・ベースの保存ポリシーのソース・コード
callhold.c	削除保留のサンプル・ソース・コード
callret.c	データ保存保護のサンプル・ソース・コード
callbuff.c	共有バッファ (コピーなし) のサンプル・ソース・コード

第3章 アプリケーションの設計に関する考慮事項

アプリケーションを設計する場合は、API の多数の側面を幅広く理解しておく必要があります。

API を理解するには、以下のトピックを検討してください。

- [14 ページの『サイズ限界の決定』](#)
- [15 ページの『API バージョン管理の保守』](#)
- [16 ページの『マルチスレッド化の使用』](#)
- [17 ページの『シグナルおよびシグナル・ハンドラー』](#)
- [17 ページの『セッションの開始または終了』](#)
- [23 ページの『オブジェクト名とオブジェクト ID』](#)
- [21 ページの『パスワード・ファイルへのアクセスの制御』](#)
- [24 ページの『セッション所有者としてのオブジェクトのアクセス』](#)
- [25 ページの『ノード間および所有者間のオブジェクトのアクセス』](#)
- [25 ページの『ファイル・スペースの管理』](#)
- [28 ページの『オブジェクトと管理クラスとの関連付け』](#)
- [30 ページの『保留と保留解除の期限切れ/削除』](#)
- [33 ページの『IBM Spectrum Protect システムの照会』](#)
- [36 ページの『サーバーへのデータの送信』](#)
- [56 ページの『バックアップおよびアーカイブのフロー・ダイアグラムの例』](#)
- [61 ページの『ファイルのグループ化』](#)
- [72 ページの『IBM Spectrum Protect API の状態遷移図の要約』](#)

アプリケーションを設計するときは、[11 ページの表 5](#) の考慮事項を検討してください。**memset** フィールドを使用した開始構造は、後続のリリースで変更される場合があります。**stVersion** 値は、製品の機能拡張ごとに増加します。

表 5. アプリケーションの設計に関する API の考慮事項

設計項目	考慮事項
ロケールの設定	<p>API を呼び出す前に、アプリケーションのロケールを設定しておく必要があります。ロケールをデフォルト値に設定するには、次のコードをアプリケーションに追加します。</p> <pre>setlocale(LC_ALL, "");</pre> <p>ロケールを別の値に設定するには、2 番目のパラメーターに適切なロケールを指定した、同じ呼び出しを使用します。詳しくは、使用する各オペレーティング・システムに関する資料を調べてください。</p>

表 5. アプリケーションの設計に関する API の考慮事項 (続き)

設計項目	考慮事項
セッション制御	<p>セッション制御に、以下の指針を適用します。</p> <ul style="list-style-type: none"> • IBM Spectrum Protect バックアップ/アーカイブ・クライアントおよび IBM Spectrum Protect API クライアント 製品ごとに固有のノード名を割り当てます。以下の製品は、これらのクライアントの例です。 <ul style="list-style-type: none"> – IBM Spectrum Protect for Mail – または IBM Spectrum Protect HSM for Windows • バックアップ手順とリストア手順を通して一貫した所有者名を使用します。 • 保護されたパスワード・ファイルへのアクセスを管理するために、passwordaccess オプションを使用します。 • データ移動のセッションは、タスク完了後に終了して、サーバー上の装置が他のセッション用に解放されるようにしてください。 • LAN フリー・データ転送を許可するには、マルチスレッド・フラグをオンに設定して dsmSetup 関数呼び出しを使用します。 • AIX でマルチスレッド・アプリケーションまたは LAN フリーを使用している場合、特に複数のプロセッサを搭載したマシンで実行している場合は、パフォーマンスの向上とより確かなスケジューリングのために、アプリケーションを開始する前に環境内で AIXTHREAD_SCOPE 環境変数を S に設定してください。以下に例を挙げます。 <div data-bbox="440 898 1474 953" style="background-color: #f0f0f0; padding: 5px; margin: 10px 0;"> <pre>EXPORT AIXTHREAD_SCOPE=S</pre> </div> <p>AIXTHREAD_SCOPE を S に設定すると、デフォルトの属性で作成されたユーザー・スレッドが、システム全体の競合スコープに置かれます。このユーザー・スレッドはカーネル・スレッドにバインドされ、カーネルによってスケジュールされます。基盤となるカーネル・スレッドは、その他のユーザー・スレッドとは共有されません。詳しくは、16 ページの『マルチスレッド化の使用』を参照してください。</p> • どの時点でも、ある API 関数を呼び出すスレッドが 1 つのセッション内で 1 つだけであるようにしてください。同じセッション・ハンドルを使用して複数のスレッドを使用する各アプリケーションは、API 呼び出しを同期化する必要があります。例えば、mutex を使用して次のように API 呼び出しを同期化します。 <pre>getTSMMutex() issue TSM API call releaseTSMMutex()</pre> <p>この手法は、スレッドがハンドルを共有する場合にのみ使用してください。各呼び出しが異なるセッション・ハンドルを使用する場合は、API 関数の並列呼び出しを使用できます。</p>

表 5. アプリケーションの設計に関する API の考慮事項 (続き)

設計項目	考慮事項
セッション制御 (継続)	<ul style="list-style-type: none"> データ移動のための消費先/作成元のスレッド化モデルを実装してください。API 呼び出しは同期式であり、dsmGetData 関数呼び出しおよび dsmSendData 関数呼び出しは、呼び出しが完了するまでブロックされます。消費先/作成元モデルを使用することにより、アプリケーションは、ネットワークの待機期間中に次のバッファを読み取ることができます。また、データの読み取り/書き込みとネットワークの分離により、ネットワークのボトルネックや遅延がある場合でもパフォーマンスが向上します。一般に、次のようになります。 <pre data-bbox="440 485 1471 558">Data thread <---> shared queue of buffers <---> communication thread (issue calls to the IBM Spectrum Protect API)</pre> <ul style="list-style-type: none"> オーバーヘッドを生じないように、複数の操作に同じセッションを使用します。多数の小さなオブジェクトを取り扱うアプリケーションでは、同じセッションを複数の小さな操作にわたって使用できるよう、セッション・プールを実装します。オーバーヘッドは、IBM Spectrum Protect サーバーへのセッションのオープンとクローズに関連があります。dsmInit/dsmInitEX 呼び出しは、たとえばマルチスレッド・アプリケーションでも一度に 1 つのスレッドだけがサインオンできるよう、直列化されます。また、サインオン時に、API はサーバーがすべての操作を実行できるよう、サーバーに対していくつかの一回限りの照会を送信します。これらの照会には、ポリシー、オプション、ファイル・スペース、およびローカル構成が含まれます。
操作手順	<p>IBM Spectrum Protect サーバーは、操作中にファイル・スペースのデータベース項目をロックすることがあります。以下の規則は、IBM Spectrum Protect API アプリケーションの設計時に適用されます。</p> <ul style="list-style-type: none"> 照会によって、トランザクション全体にわたってファイル・スペースがロックされます。 照会によるロックは、他の照会操作と共有できるため、同一のファイル・スペースで複数の照会操作を並行して実行することができます。 IBM Spectrum Protect サーバーのデータベースを変更 (DB Chg) するために、送信、取得、名前変更、更新、および削除の各操作が使用されます。 DB Chg 操作を完了するには、トランザクションの最後でデータベース変更の間、ファイル・スペースをロックする必要があります。 同一ファイル・スペースで複数の DB Chg 操作を並行して実行することができます。トランザクションの終了時に、シーケンスがロックを待っている間に遅延が生じる場合があります。 照会によるロックは DB Chg 操作と共有できません。DB Chg 操作は同一ファイル・スペースでの照会の開始を遅らせるため、同一ファイル・スペースの DB Chg 操作から照会を分離して直列化するようにアプリケーションを設計してください。
オブジェクトの命名	<p>オブジェクトに命名するときは、以下の要因を考慮してください。</p> <ul style="list-style-type: none"> 具体的なオブジェクト名は、高位オブジェクト名および低位オブジェクト名です。例えば、日付スタンプのような固有 ID が名前に含まれている場合、バックアップ・オブジェクトは常に活動状態です。オブジェクトは、dsmDeleteObj 関数呼び出しによって非活動としてマーク付けされた場合にのみ、有効期限が切れます。 オブジェクトのリストア方式によって、照会が容易な名前の形式が決まります。部分オブジェクト・リストア (POR) を使用する場合は、圧縮は使用できません。圧縮を抑止するには、dsmSendObj objAttr objCompressed=bTrue 関数を使用します。

表 5. アプリケーションの設計に関する API の考慮事項 (続き)

設計項目	考慮事項
オブジェクトのグループ化	ファイル・スペースを使用することにより、オブジェクトを論理的にグループ化します。ファイル・スペースはサーバー上のコンテナであり、オブジェクトのグループ化カテゴリを提供します。API は初期サインオン時にすべてのファイル・スペースを照会し、照会時にもそれを行うため、ファイル・スペースの数を制限する必要があります。妥当な想定として、1つのアプリケーションがセットアップするファイル・スペースの数は、1ノードあたり 20 個から 100 個です。API はそれより多くのファイル・スペースに対応できませんが、それぞれのファイル・スペースでセッションのオーバーヘッドが生じます。より細かく分離するためには、アプリケーションで <code>directory</code> オブジェクトを使用します。
オブジェクト処理	将来のリストアに使用するために <code>objectID</code> 値を保管しておくのはやめてください。これらの値は、オブジェクトの存続期間中、それらが永続的であることは保証されません。 リストア時には、リストアの順序に特に注意してください。照会の後、リストアの前に上記の ID 値に基づいてソートを行います。複数のタイプのシリアル・メディアを使用している場合は、さまざまなタイプのメディアに別々のセッションでアクセスしてください。詳しくは、次のトピックを参照してください。 65 ページの『リストア順序によるオブジェクトの選択およびソート』
管理クラス	アプリケーションが、アプリケーション・オブジェクトに関連付けられる管理クラスに対してどれだけの制御を持つ必要があるかを考慮してください。include ステートメントを定義することもできますし、また <code>dsmSendObj</code> 関数呼び出し時に名前を指定することもできます。
オブジェクト・サイズ	IBM Spectrum Protect は、各オブジェクトのサイズ見積もりを知る必要があります。アプリケーションでのオブジェクト・サイズの見積もり方法を考慮してください。オブジェクト・サイズは、過小に見積もるより過大に見積もることをお勧めします。

サイズ限界の決定

API 内の特定のデータ構造またはフィールドには、サイズの制限があります。これらの構造の名前またはその他のテキスト・フィールドは、事前に決められた長さを超えることができない場合がよくあります。

以下のフィールドは、サイズ限界があるデータ構造の例です。

- アプリケーション・タイプ
- アーカイブ記述
- コピー・グループ先
- コピー・グループ名
- ファイル・スペース情報
- 管理クラス名
- オブジェクト所有者名
- パスワード

これらの限界は、ヘッダー・ファイル `dsmapi.h` 内で定数として定義されます。ストレージ割り振りはすべて、入力する数字ではなく、これらの定数に基づいて行う必要があります。詳しくは、[155 ページの『付録 B API タイプ定義ソース・ファイル』](#)を参照してください。

API バージョン管理の保守

すべての API にはバージョン管理の形式がいくつかあります。アプリケーションでご使用の API のバージョンと、ユーザー がそれぞれのワークステーションにインストールしている API ライブラリー のバージョンの間に互換性がなければなりません。

API を使用する時に入力する最初の API 呼び出しは **dsmQueryApiVersionEx** でなければなりません。この呼び出しは、以下のタスクを実行します。

- API ライブラリーがエンド・ユーザーのシステムにインストールされ、使用できる ことを確認する
- アプリケーションによってアクセスされる API ライブラリーのバージョン・レベル を返す

API は上方互換性をもつように設計されています。API ライブラリーの旧バージョンまたは旧リリース用に作成されたアプリケーションは、それより新しいバージョンで実行 していれば、正しく作動します。

API ライブラリーのリリースを判別することは重要です。それは、一部のリリース には異なるメモリー所要量とデータ構造定義があるためです。下方互換性はありません。ご使用のプラットフォームの詳しくは、[15 ページの表 6](#) を参照してください。

表 6. プラットフォームの互換性情報

プラットフォーム	説明
Windows	メッセージ・ファイルは、ライブラリー (DLL) と同じレベルでなければなりません。
UNIX または Linux	API ライブラリーおよびメッセージ・ファイルは同じレベルでなければなりません。

dsmQueryApiVersionEx 呼び出しは、エンド・ユーザーのワークステーションにインストールされている API ライブラリーのバージョンを戻します。これにより、戻された値と アプリケーション・クライアントが使用している API のバージョンとを比較できます。

アプリケーション・クライアントの API のバージョン番号は、`dsmapi.h` に定義されている次の 4 つの定数のセットとして、コンパイル済みオブジェクト・コード内に入力されています。:

```
DSM_API_VERSION
DSM_API_RELEASE
DSM_API_LEVEL
DSM_API_SUB_LEVEL
```

[155 ページの『付録 B API タイプ定義ソース・ファイル』](#)を参照してください。

アプリケーション・クライアントの API バージョンは、ユーザーのシステムに インストールされている API ライブラリーと等しいか、またはそれ以前の バージョンでなければなりません。それ以外の場合は注意が必要です。**dsmQueryApiVersionEx** 呼び出し は、API セッションが開始されているかどうかに関係なく、いつでも出すことができます。

API によって使用されるデータ構造も、その中にバージョン制御情報を持っています。構造は、最初のフィールドにバージョン情報を持っています。構造は拡張されているため、バージョン番号が増えています。バージョン・フィールドの初期設定するときは、`dsmapi.h` 内の定義済み構造 バージョン値を使用してください。

[16 ページの図 1](#) は、ヘッダー・ファイル `dsmapi.h` からの構造 **dsmApiVersionEx** のタイプ定義を示しています。次に、この例では、**apiLibVer** という名前のグローバル変数を定義しています。この例は、エンド・ユーザーの API ライブラリーのバージョンを戻すために、**dsmQueryApiVersionEx** に対する呼び出しの中でこの変数を使用する方法も示しています。最後に、戻された値が、アプリケーション・クライアントの API のバージョン番号と比較されます。

```

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

dsmApiVersionEx apiLibVer;

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

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

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

```

図 1. API のバージョン・レベルの取得の例

マルチスレッド化の使用

マルチスレッド API を使用すると、アプリケーションは、同じプロセス内で IBM Spectrum Protect サーバーとのセッションを複数作成することができます。API は再度入力することができます。複数の呼び出しを、異なるスレッド内から 並列で実行できます。

ヒント: マルチスレッド API を想定しているアプリケーションを実行する場合には、**dsmQueryAPIVersionEx** 呼び出しを使用します。

マルチスレッド・モードで API を実行するには、**dsmSetUp** 呼び出しの *mtflag* 値を **DSM_MULTITHREAD** に設定します。**dsmSetUp** 呼び出しは、**dsmQueryAPIVersionEx** 呼び出しの後に出す最初の呼び出しでなければなりません。この呼び出しはスレッドが **dsmInitEx** を呼び出す前に戻る必要があります。すべてのスレッドが処理を完了したら、**dsmCleanUp** への呼び出しを入力します。すべてのスレッドが処理を完了するまで、最初の処理は終了してはなりません。サンプル・アプリケーションの *callmt1.c* を参照してください。

制限: API のデフォルトは単一スレッド・モードになります。アプリケーションが *mtflag* 値を **DSM_MULTITHREAD** に設定して **dsmSetUp** を呼び出さない限り、API は各処理ごとに 1 つのセッションしか許可しません。

dsmSetUp が正常に完了したら、アプリケーションは複数のスレッドを開始して、複数の **dsmInitEx** 呼び出しを入力できます。それぞれの **dsmInitEx** 呼び出しが、そのセッションのハンドルを戻します。そのスレッドでの、そのセッションのためのそれ以降の呼び出しは、そのハンドル値を使用しなければなりません。特定の値は、処理内で使用する環境変数 (**dsmSetUp** で設定した値) です。各 **dsmInitEx** 呼び出しは、オプションを再解析します。**dsmInitEx** 呼び出しで上書きファイルまたはオプション・ストリングを指定することによって、異なるオプションで各スレッドを実行できます。これによって、異なるスレッドが異なるサーバーに接続したり、異なるノード名を使用することができます。

推奨: HP では、スレッド・スタックを 64K 以上に設定してください。デフォルト値 (32K) ではスレッド・スタックが不足する可能性があります。

アプリケーション・ユーザーに LAN フリー・セッションの使用を許可するには、アプリケーションで **dsmSetUp** *mtFlag* **DSM_MULTITHREAD** を指定します。単一スレッド・アプリケーションの場合でも、この指定は必要です。このフラグは、IBM Spectrum Protect の LAN フリー・インターフェースに必要なスレッド化を活動化します。

シグナルおよびシグナル・ハンドラー

ユーザーまたはオペレーティング・システムからの信号を処理するのは、アプリケーションの責任です。ユーザーが **CTRL+C** のキー・ストローク・シーケンスを入力すると、アプリケーションはそのシグナルをキャッチし、活動スレッドのそれぞれに対して **dsmTerminate** 呼び出しを送信する必要があります。その後、**dsmCleanUp** を呼び出して終了しなければなりません。セッションが正しくクローズされなかった場合は、サーバー上で予期しない結果が生じる可能性があります。

アプリケーションには、そのアプリケーションを終了させるシグナルとして、例えば SIGPIPE や SIGUSR1 などのシグナル・ハンドラーが必要です。これによって、アプリケーションは API から戻りコードを受け取ります。例えば、SIGPIPE を無視するには、**signal(SIGPIPE, SIG_IGN)** という命令をアプリケーション内に追加します。この情報を追加すると、パイプの切断が発生しても、アプリケーションは終了せず、正しい戻りコードが返されます。

セッションの開始または終了

IBM Spectrum Protect はセッション・ベースの製品であり、すべての活動を IBM Spectrum Protect セッション内で実行する必要があります。セッションを開始するには、アプリケーションが **dsmInitEx** 呼び出しを開始します。この呼び出しは、**dsmQueryApiVersionEx**、**dsmQueryCliOptions**、または **dsmSetUp** 以外の他のすべての API 呼び出しの前に実行する必要があります。

dsmQueryCliOptions 関数は、**dsmInitEx** 呼び出しの前でのみ、呼び出すことができます。この関数は、オプション・ファイル、圧縮設定値、および通信パラメーターなど、重要なオプションの値を返します。**dsmInitEx** 呼び出しは、この呼び出しで渡されたパラメーターに指示されているとおりか、オプション・ファイルに定義されているとおりに、サーバーとのセッションをセットアップします。

クライアントのノード名、所有者名、およびパスワードの各パラメーターが、**dsmInitEx** 呼び出しに渡されます。所有者名では、大文字小文字の区別がありますが、ノード名およびパスワードでは区別がありません。アプリケーション・クライアント・ノードは、セッションが開始される前にサーバーに登録されている必要があります。

API アプリケーション・クライアント がサーバーとのセッションを開始するたびに、クライアントのアプリケーション・タイプがサーバーに登録されます。アプリケーション・タイプ値には、常にオペレーティング・システムの省略語を指定してください。この値は、サーバーのプラットフォーム・フィールドに入力されるからです。ストリングの最大長は **DSM_MAX_PLATFORM_LENGTH** です。

dsmInitEx 関数 呼び出しは、アプリケーション・クライアント の API 構成ファイルとオプション・リストとの IBM Spectrum Protect セッションを確立します。アプリケーション・クライアントは、API 構成ファイルおよびオプション・リストを使用して、多数の IBM Spectrum Protect オプションを設定できます。これらの値は、インストール時にユーザー構成ファイルに設定された値を指定変更します。ユーザーは、管理者が定義したオプションを変更できません。アプリケーション・クライアントが特定の構成ファイルおよびオプション・リストを持っていない場合は、これらのパラメーターの両方を NULL に設定できます。構成ファイルの詳細は、次のトピックを参照してください。

1 ページの『構成ファイルとオプション・ファイルの理解』

dsmInitEx 関数呼び出しは、拡張検査を許可するパラメーターを使用して、IBM Spectrum Protect セッションを確立します。

dsmInitEx 関数呼び出し と **dsmInitExOut** 情報戻りコードをチェックしてください。戻りコードが OK (RC=ok) で、情報戻りコード (infoRC) が **DSM_RC_REJECT_LASTSESS_CANCELED** である場合は、管理者によって最後のセッションが取り消されています。現行セッションを即時に終了するには、**dsmTerminate** を呼び出してください。

dsmQuerySessOptions 呼び出しは、**dsmQueryCliOptions** 呼び出しと同じフィールドを 戻します。この呼び出しは、セッション内でのみ送信できます。これらの値はオプション・ファイル、および

dsmInitEx 呼び出しでの指定変更に基づいて、そのセッション内で有効なクライアント・オプションを表します

セッションが開始された後、アプリケーションは **dsmQuerySessInfo** の呼び出しを送信して、このセッション用に設定されたサーバー・パラメーターを判別できます。ポリシー・ドメインおよびトランザクション限界のような項目は、この呼び出しによってアプリケーションに戻されます。

dsmTerminate 呼び出しでセッションを終了します。サーバーとのすべての接続がクローズされ、このセッションに関連付けられているすべてのリソースが解放されます。

セッションの開始と終了の例については、次のトピックを参照してください。

20 ページの図 2

この例では、**dsmInitEx** および **dsmTerminate** に対する呼び出しで使用される、いくつかのグローバル変数とローカル変数を定義しています。

dsmInitEx 呼び出しは **dsmHandle** へのポインターをパラメーターとして使用します。**dsmTerminate** 呼び出しは **dsmHandle** をパラメーターとして使用します。

20 ページの図 3 の例は、**rcApiOut** の詳細を示しています。関数 **rcApiOut** は、戻りコードをメッセージに変換する API 関数 **dsmRCMsg** を呼び出します。

次に、**rcApiOut** 呼び出しは、そのメッセージをユーザー用に出力します。**rcApiOut** のバージョンが API サンプル・アプリケーションに組み込まれています。**dsmApiVersion** 関数は、ヘッダー・ファイル **dsmapi.h** 内にある型定義です。

セッション・セキュリティ

IBM Spectrum Protect セッション・ベース・システムは、アプリケーションが安全な方法でセッションを開始できるようにするセキュリティ・コンポーネントを備えています。これらのセキュリティ基準は、サーバーに対する無許可アクセスを禁止し、システム保全性の保証に役立ちます。

サーバーで開始されるすべてのセッションは、サインオン・プロセスを完了する必要があるため、これにはパスワードが必要です。クライアントのノード名と結合されたパスワードは、サーバーへの接続時に適切な許可をもっていることを保証します。アプリケーション・クライアントは、セッションを開始するために API にこのパスワードを提供します。

パスワード処理の方式には **passwordaccess=prompt** と **passwordaccess=generate** の 2 つがあります。**passwordaccess=prompt** オプションを使用する場合は、各 **dsmInitEx** 呼び出しにパスワード値を組み込まなければなりません。または、**dsmInitEx** 呼び出しでノード名および所有者名を提供することもできます。

パスワードには、それぞれに有効期限が付けられています。**dsmInitEx** 呼び出しがパスワード期限切れ戻りコード (**DSM_RC_REJECT_VERIFIER_EXPIRED**) を出して失敗する場合には、アプリケーション・クライアントは、**dsmInitEx** で戻されるハンドルを使用して、**dsmChangePW** を呼び出す必要があります。これによって、パスワードが更新されるので、セッションを正常に確立することができます。21 ページの図 4 の例は、**dsmChangePW** を使用してパスワードを変更する手順を具体的に示します。ログイン所有者は、パスワードを変更する際に root ユーザー ID または許可ユーザー ID を使用する必要があります。

2 番目の方式の **passwordaccess=generate** は、パスワードの値を暗号化してファイルに保管します。ノード名および所有者名を **dsmInitEx** 呼び出しで提供することはできません。システム・デフォルト値が使用されます。これによって、パスワード・ファイルのセキュリティが保たれます。パスワードが失効すると、**generate** パラメーターによって新規のパスワードが作成され、パスワード・ファイルが自動的に更新されます。

ヒント:

1. 2 台の異なる物理マシンに同じ IBM Spectrum Protect ノード名が付いているか、または複数のサーバー・スタンプを使用して 1 つのノードに複数のパスが定義される場合、**passwordaccess=generate** は、パスワード期限切れ後、最初に使用されるスタンプに対してのみ動作することがあります。最初のクライアント/サーバー接続時に、各サーバー・スタンプに関して別々に同じパスワードのプロンプトが表示され、各スタンプに対してパスワードのコピーが別々に保管されます。パスワードの有効期限が切

れると、最初にクライアント/サーバー接続を行うスタンザに対して新規パスワードが生成されます。その後のほかのサーバー・スタンザを介しての接続の試みはすべて失敗します。それは、古いパスワードのそれぞれのコピーと、パスワード期限切れ後に最初に使用されたスタンザによって生成された更新済みコピーとの間に、論理リンクが存在しないからです。この場合、有効期限前、またはこの状況からのリカバリーとして期限後に、パスワードを更新する必要があります。これは、次の手順で行います。

- a. **dsmadm** を実行して、サーバー上のパスワードを更新します。
 - b. **dsmc -servername=stanza1** を実行し、新規パスワードを使用して適切なエントリーを生成します。
 - c. **dsmc -servername=stanza2** を実行し、新規パスワードを使用して適切なエントリーを生成します。
2. UNIX または Linux の場合: *passwordaccess=prompt* を使用してパスワードを変更できるのは、root ユーザーまたは許可ユーザーのみです。 *passwordaccess=generate* を使用してパスワード・ファイルを開始できるのは、root ユーザーまたは許可ユーザーのみです。

制約事項: オプションの **users** と **groups** は認識されません。

アプリケーションは、アクセス・フィルタを設定するなどの他の方法によってユーザーによるアクセスを制限できます。

複数の IP 接続を使用して単一の IBM Spectrum Protect サーバーに接続するアプリケーションでは、使用するノード名と IBM Spectrum Protect クライアント・パスワードをどのセッションでも同じにする必要があります。これを実現するには、次のステップに従います。

1. **dsm.sys** ファイル内に IBM Spectrum Protect サーバー・スタンザを 1 つ定義します。 .
2. デフォルトの IP アドレスを使用しない接続に関しては、**dsmInitEx** 呼び出しで **TCPserver** アドレスと **TCPport** のオプション値を指定します。

上記の値は、IP 接続の設定情報を上書きします。ただし、**dsm.sys** 内のスタンザと同じノード情報とパスワード情報が引き続きセッションで使用されます。

注: クラスタ内のノードは、1 つのパスワードを共有します。

```

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);
}

```

図 2. セッションの開始と終了の例

```

void rcApiOut (dsUInt32_t handle, dsInt16_t rc)
{
    char *msgBuf ;

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

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

```

図 3. rcApiOut の詳細


```

printf("Enter your current password:");
gets(current_pw);
printf("Enter your new password:");
gets(new_pw1);
printf("Enter your new password again:");
gets(new_pw2);
/* 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;

```

図 4. パスワードの変更の例

パスワード・ファイルへのアクセスの制御

UNIX および Linux のシステム上の保護されたパスワード・ファイルへのアクセスを制御する場合、許可ユーザーとしてログオンし、passwordaccess オプションを generate に設定します。.

手順

passwordaccess を generate に設定するときは、以下の手順を実行します。:

1. **dsmSetUp** への呼び出しを使用して *argv[0]* を渡すアプリケーションを作成します。*argv[0]* には、API を呼び出すアプリケーションの名前が含まれています。アプリケーションは、許可ユーザーの実行を許可されますが、管理者は許可ユーザーのログイン名を決定する必要があります。
2. アプリケーション実行可能ファイルの有効ユーザー ID ビット (S ビット) を On に設定します。これによって、アプリケーション実行可能ファイルの所有者が許可ユーザーになり、パスワード・ファイルの作成、パスワードの更新、およびアプリケーションの実行ができるようになります。アプリケーション実行可能ファイルの所有者は、プログラムを実行するユーザー ID と同じでなければなりません。以下の例では、User が user1 で、アプリケーション実行可能ファイルの名前が app1A で、user1 は /home/user1 ディレクトリーの読み取り/書き込み許可を持っているものとします。app1A 実行可能ファイルには、以下の許可があります。

```
-rwsr-xr-x user1    group1    app1A
```

3. アプリケーションのユーザーに、ログインする際は許可ユーザー名を使用するように指示します。IBM Spectrum Protect は、ログイン ID がアプリケーション実行可能ファイルの所有者と一致することをチェックしてから保護パスワード・ファイルへのアクセスを許可します。
4. このユーザーが読み取り/書き込みアクセス権限を持っているディレクトリーを指すように、dsm.sys ファイル内の passworddir オプションを設定します。
例えば、dsm.sys ファイルのサーバー・スタンザ内に以下の行を入力します。

```
passworddir /home/user1
```

5. パスワード・ファイルを作成して、許可ユーザーがファイルを確実に所有するようにします。
6. user1 としてログオンし、app1A を実行します。
7. **dsmSetUp** を呼び出して、*argv* で渡します。

クライアント所有者権限を持つ管理ユーザーの作成

クライアント所有者権限を持つ管理ユーザーは、**dsmInitEx** 関数呼び出しのパラメーターを設定して、セッションを開始できます。このユーザーは、定義されたノードに対するバックアップおよびリストアの権限を持つ「管理ユーザー」になることができます。

手順

クライアント所有者権限を受信するには、サーバー上で以下のステップを実行します。

1. 管理ユーザーを定義します。

```
REGister Admin admin_name password
```

各構成要素について説明します。

- *admin_name* は管理ユーザー名です。
 - *password* は管理者のパスワードです。
2. 権限レベルを定義します。システム権限またはポリシー権限を持つユーザーにも、クライアント所有者権限があります。

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

各構成要素について説明します。

- *admin_name* は管理ユーザーです。
 - *classes* はノードです。
 - *authority* は、以下のいずれかのレベルの権限になります。
 - *owner*: ノードのフルバックアップと権限のリストア
 - *node*: 単一ノード
 - *domain*: ノードのグループ
3. 単一のノードに対するアクセスを定義します。

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

各構成要素について説明します。

- *node_name* はクライアント・ユーザー・ノードです
- *password* は、クライアント・ユーザー・ノードのパスワードです。
- *user_id* は、管理ユーザー名です。

タスクの結果

アプリケーションが管理ユーザーを使用する場合、**dsmInitEx** 関数は、*userName* パラメーターと *userNamePswd* パラメーターで呼び出されます。

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

passwordaccess オプションは、*generate* または *prompt* のどちらかに設定することができます。どちらのパラメーターを使用する場合でも、*userNamePswd* の値によってセッションが開始されます。セッションが開始されると、そのノードに対して任意のバックアップ・プロセスやリストア・プロセスを実行できます。

オブジェクト名とオブジェクト ID

IBM Spectrum Protect サーバーは、オブジェクト・ストレージ・サーバーであり、その主要な機能は、指定されたオブジェクトを効率よく保管し、リトリートすることです。オブジェクト ID は、各オブジェクトに固有であり、エクスポートまたはインポートを使用する場合を除き、オブジェクトの存続期間中は保持されます。

この要件を満たすために、IBM Spectrum Protect には 2 つの主ストレージ域 (データベースとデータ・ストレージ) があります。

- データベースには、オブジェクトに関連したすべてのメタデータ (名前、属性など) が入ります。
- データ・ストレージにはオブジェクト・データが入ります。データ・ストレージは、実際にシステム管理者によって定義されたストレージ階層です。データは、コストおよびアクセスの必要に応じて、オンラインまたはオフライン・メディア上に効率よく保管され、管理されます。

サーバーに保管された各オブジェクトには、それぞれサーバーに関連した名前が付けられます。クライアントは、次のようなこの名前のキー・コンポーネントを制御します。

- ファイル・スペース名
- 高位名
- 低位名
- オブジェクト・タイプ

アプリケーション用のオブジェクトを命名する場合には、エンド・ユーザーに対しては、全オブジェクト名の外部名を使用する必要があります。特に、アプリケーションが実行されるとき、エンド・ユーザーは INCLUDE または EXCLUDE ステートメントでオブジェクトの指定が必要になることがあります。これらのステートメントにおけるオブジェクト名の正確な構文は、プラットフォームに依存します。Windows オペレーティング・システムでは、INCLUDE または EXCLUDE ステートメントにおいて、ファイル・スペース名自体ではなく、ファイル・スペースに関連したドライブ文字が使用されます。

オブジェクトの作成時に割り当てたオブジェクト ID の値は、リストア・プロセスを実行するときには異なる値になっている場合があります。アプリケーションは、オブジェクト名を保管した上で、リストアを実行する前に現行のオブジェクト ID を照会して取得する必要があります。

ファイル・スペース名

ファイル・スペース名は最も重要なストレージ・コンポーネントの 1 つです。この名前は、ファイル・システムの名前、ディスク・ドライブの名前、あるいは関連データをグループ化する他の任意の高位修飾子にすることができます。

IBM Spectrum Protect は、ファイル・スペースを使用して、データが配置されているファイル・システムまたはディスク・ドライブを識別します。この方法で、指定のファイル・スペース内のすべてのオブジェクトを照会するなどのアクションを、ファイル・スペース内のすべてのエンティティ上で実行できます。ファイル・スペースは、IBM Spectrum Protect 命名規則の非常に重要なコンポーネントなので、ファイル・スペースの登録、更新、照会、および削除の際にはユーザーは特別な呼び出しを使用します。

サーバーには、IBM Spectrum Protect ストレージ内の所定のノード上のファイル・スペースを照会し、必要に応じてそれらのファイル・スペースを削除するための管理コマンドもあります。アプリケーション・クライアントによって保管されるすべてのデータには、ファイル・スペース名が関連付けられていなければなりません。システム内の類似のデータをグループ化するように、気を付けながら名前を選択してください。

干渉が起らないようにするには、アプリケーション・クライアントはバックアップ/アーカイブ・クライアントが使用するファイル・システム名とは異なるファイル・スペース名を選択する必要があります。アプリケーション・クライアントは、必要ならば、エンド・ユーザーが INCLUDE および EXCLUDE ステートメント用のオブジェクトを識別できるように、そのファイル・スペース名をパブリッシュする必要があります。

注: Windows プラットフォームでは、ドライブ文字は、ファイル・スペースに関連付けられます。ファイル・スペースを登録したり更新する場合にはドライブ文字を指定しなければなりません。include-exclude リストでドライブ文字を参照するので、各ドライブ文字とそれに関連するファイル・スペースを記録する

必要があります。サンプル・プログラム `dapism` では、ドライブ文字はデフォルトで「G」に設定されています。

サンプル・プログラムについての詳細は [5 ページの『第 2 章 API サンプル・アプリケーションの作成と実行』](#) を参照してください。

高位名と低位名

オブジェクト名の他の 2 つのコンポーネントは、高位名修飾子および低位名修飾子です。高位名修飾子はオブジェクトが属するディレクトリー・パスであり、低位名修飾子は そのディレクトリー・パス内のオブジェクトの実際の名前です。

ファイル・スペース 名、高位名、および低位名を連結する場合、それらは、クライアントが実行しているオペレーティング・システムでの、構文上正しい名前になっていなければなりません。この名前は、オブジェクトとしてシステムに存在している必要もなければ、ローカル・ファイル・システム上の実データに似ている必要もありません。しかし、**dsmBindMC** 呼び出しによって正しく処理されるように、標準命名規則を満たすものでなければなりません。ポリシー管理に関連した命名の考慮事項については、[41 ページの『バックアップ・オブジェクトおよびアーカイブ・オブジェクトの把握』](#) を参照してください。

オブジェクト・タイプ

オブジェクト・タイプは、オブジェクトをファイルまたはディレクトリーとして識別します。ファイルは、属性と 2 進データの両方を含むオブジェクトです。ディレクトリーは、属性のみを含むオブジェクトです。

[24 ページの表 7](#) は、プラットフォームごとのオブジェクト名用のアプリケーション・クライアントのコーディング例です。

表 7. プラットフォームごとのアプリケーション・オブジェクト名の例	
プラットフォーム	オブジェクト名用のクライアント・コード
UNIX または Linux	/myfs/highlev/lowlev
Windows	"myvol¥¥highlev¥¥lowlev" 注：Windows プラットフォーム上では、円記号 (¥) はエスケープ文字であるため、2 個の円記号が 1 個の円記号に変換されます。ファイル・スペース名は、UNIX または Linux プラットフォーム上ではスラッシュで始まりますが、Windows プラットフォーム上ではスラッシュで始まりません。

セッション所有者としてのオブジェクトのアクセス

各オブジェクトには、関連する所有者名が付いています。アクセス先のオブジェクトを決定する規則は、セッションの開始時に使用された所有者名に依存します。このセッション所有者の値を使用して、オブジェクトへのアクセスを制御できます。

セッション所有者は、**dsmInitEx** の呼び出し中に `clientOwnerNameP` パラメーターで設定されます。**NULL** の **dsmInitEx** 所有者名を使用してセッションを開始し、`passwordaccess=prompt` を使用した場合、そのセッション所有者は、セッション (root または 許可ユーザー) 権限を使用して処理されます。これは、root ユーザー ID または 許可ユーザー ID を使用してログインし、`passwordaccess=generate` を使用した場合にも当てはまります。この方法で開始されたセッションでは、当該オブジェクトの実際の所有者とは無関係に、このノードによって所有される任意のオブジェクト上で任意のアクションを実行できます。

セッションが特定の所有者名を指定して開始された場合、そのセッションは、そのオブジェクト所有者名に関連付けられたオブジェクトに関するアクションしか実行できません。システム内へのバックアップまたはアーカイブはすべて、この所有者名に関連付けられていなければなりません。実行されるどの照会もこの所有者名に関連付けられた値のみを戻します。オブジェクト所有者の値は、**dsmSendObj** 呼び出し中に、**ObjAttr** 構造の **Owner** フィールドに設定されます。所有者名では、大文字小文字が区別されます。[25 ページの表 8](#) は、ユーザーがオブジェクトへのアクセス権限を持つ条件の要約です。

表 8. オブジェクトへのユーザー・アクセスの要約

セッション所有者	オブジェクト所有者	ユーザー・アクセスの可否
NULL (root、システム所有者)	『』 (空ストリング)	Yes
NULL	特定の名前	Yes
特定の名前	『』 (空ストリング)	No
特定の名前	同じ名前	Yes
特定の名前	異なる名前	No

ノード間および所有者間のオブジェクトのアクセス

dsmSetAccess、**dsmDeleteAccess**、および **dsmQueryAccess** の 3 つの関数呼び出しは、同じプラットフォーム上の複数ノード間、複数所有者間のアクセスをサポートします。これらの関数は、**dsmInitEx** で渡される **-fromnode** および **-fromowner** ストリング・オプションと一緒に使用して、API を介して完全なノード間照会、リストアおよびリトリブの処理ができるようにします。

例えば、ノード A のユーザー A が **dsmSetAccess** 関数呼び出しを使用して、/db ファイル・スペースにあるユーザー A のバックアップへのアクセスをノード B のユーザー B に許可する場合を考えます。アクセス・ルールは、次のように表示されます。

ID	タイプ	Node	ユーザー	パス
1	バックアップ	ノード B	ユーザー B	/db/*/*

ユーザー B がノード B でログオンしているときの、**dsmInitEx** へのオプション・ストリングは、次のとおりです。

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

これらのオプションは、このセッションのために設定されているものです。すべての照会が、ノード A のファイル・スペースおよびファイルにアクセスします。バックアップおよびアーカイブは許可されません。ユーザー B がアクセスを許可されているファイル・スペースからは、照会、リストア、およびリトリブの処理のみが許されます。**-fromnode** または **-fromowner** オプションを設定してサインインしている場合に、アプリケーションが **dsmBeginTxn** を使用する操作 (例えば、バックアップまたは更新) を実行しようとする、**dsmBeginTxn** は失敗し、戻りコード **DSM_RC_ABORT_NODE_NOT_AUTHORIZED** が返されます。詳しくは、個々の関数呼び出しおよび 115 ページの『**dsmInitEx**』を参照してください。

ヒント : UNIX および Linux では、**dsmInitEx** に **-fromowner=root** を指定することができます。これによって、**set access** が実行された場合に、非 root ユーザーが、ルートによって所有されているファイルにアクセスすることができます。

dsmInitEx のオプション・ストリングに **asnodename** オプションを指定し、適切な関数と組み合わせて使用すると、IBM Spectrum Protect サーバーに登録されているターゲット・ノード名を使用してデータのバックアップ、アーカイブ、リストア、リトリブ、照会、または削除を実行できます。このオプションを使用可能にする情報については、78 ページの『クライアント・ノード・プロキシ・サポートを使用した複数ノードのバックアップ』を参照してください。

ファイル・スペースの管理

ファイル・スペースはシステムの操作上重要であるため、ファイル・スペース ID の登録、更新、および削除には、別々の呼び出しのセットが使用されます。システム上のファイル・スペースに関連したオブジェクトを保管する前に、最初に、IBM Spectrum Protect を使用してファイル・スペースを登録する必要があります。

dsmRegisterFS 呼び出しを使用して、このタスクを実行してください。オブジェクト名と ID の詳しくは、23 ページの『オブジェクト名とオブジェクト ID』を参照してください。

ファイル・スペース ID は、3 つの部分からなる名前階層の中の最高位修飾子です。ファイル・スペース内の関連データをグループ化すると、そのデータの管理が大幅に容易になります。例えば、アプリケーション・クライアント または IBM Spectrum Protect サーバー管理者は、ファイル・スペースおよびそのファイル・スペース 内のすべてのオブジェクトを削除することができます。

ファイル・スペースを使用すると、アプリケーション・クライアントはそのファイル・スペースに関する情報をサーバーに提供でき、次に管理者がその情報をサーバーに照会できます。この情報は、照会によって **qryRespFSData** 構造に戻され、次のファイル・システム情報が含まれます。

タイプ	定義
fstype	ファイル・スペース・タイプ。このフィールドは、アプリケーション・クライアント が設定する文字ストリングです。
fsAttr[platform].fsInfo	クライアントに特定のデータに使用されるクライアント情報フィールド。
capacity	ファイル・スペース内のスペースの合計。
occupancy	ファイル・スペース内で現在占有されているスペースの量。
backStartDate	最後にバックアップが開始されたときのタイム・スタンプ (dsmUpdateFS 呼び出しで設定される)。
backCompleteDate	最後にバックアップが完了したときのタイム・スタンプ (dsmUpdateFS 呼び出しで設定される)。

capacity および **occupancy** の使用は、アプリケーション・クライアントによって異なります。一部のアプリケーションは、ファイル・スペースのサイズに関する情報を必要としない場合があります。その場合、これらのフィールドはデフォルトの 0 に設定できます。ファイル・スペースの照会に関する詳細は、[33 ページの『IBM Spectrum Protect システムの照会』](#)を参照してください。

ファイル・スペースがシステムに登録されたら、いつでもオブジェクトのバックアップまたはアーカイブを行うことができます。バックアップ操作またはアーカイブ操作の後にファイル・スペースの配置フィールドと容量フィールドを更新するには、**dsmUpdateFS** を呼び出します。この呼び出しによって、ファイル・システムの配置と容量に関する値が現行値であることが保証されます。**fsinfo**、**backupstart**、および **backupcomplete** の各フィールドを更新することもできます。

最後のバックアップ日付をモニターしたいときは、バックアップを開始する前に、**dsmUpdateFS** 呼び出しを行います。更新アクションを **DSM_FSUPD_BACKSTARTDATE** に設定します。こうすることにより、サーバーは、ファイル・スペースの **backStartDate** フィールドに現在の時刻を設定します。そのファイル・スペースについてのバックアップが完了した後、更新アクションを **DSM_FSUPD_BACKCOMPLETEDATE** に設定して、**dsmUpdateFS** 呼び出しを行います。この呼び出しによって、バックアップ終了時にタイム・スタンプが作成されます。

ファイル・スペースが不要になったら、**dsmDeleteFS** コマンドを使用してこれを削除することができます。UNIX または Linux オペレーティング・システムでは、root ユーザーまたは許可ユーザーのみがファイル・スペースを削除できます。

[27 ページの図 5](#) の例は、UNIX または Linux の場合の 3 つのファイル・スペース呼び出しの使用法を示しています。Windows の場合の 3 つのファイル・スペース呼び出しの使用法の例については、ご使用のシステムにインストールされているサンプル・プログラム・コードを参照してください。

```

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

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

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

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

```

図 5. ファイル・スペース処理の例 (その 1)

```

/* Update the file space. */

dsmFSUpd      updFilespace;          /* for update FS */

updFilespace.stVersion = dsmFSUpdVersion;
updFilespace.fsType = 0;              /* no change */
updFilespace.occupancy.hi = 0;
updFilespace.occupancy.lo = 50;
updFilespace.capacity.hi = 0;
updFilespace.capacity.lo = 200;
strcpy(updFilespace.fsAttr.unixFSAttr.fsInfo,
    "My update for filesystem");
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\nrc=%d\n", rc);

```

図 6. ファイル・スペース処理の例 (その 2)

```

/* Delete the file space. */

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

```

図 7. ファイル・スペース処理の例 (その 3)

オブジェクトと管理クラスとの関連付け

IBM Spectrum Protect の最も重要な機能に、オブジェクトをどのように IBM Spectrum Protect ストレージ内に保管し、管理するかを定義するためのポリシー (管理クラス) の使用があります。オブジェクトは、バックアップまたはアーカイブされたときに、管理クラスに関連付けられます。

この管理クラスは、次のことを決定します。

- バックアップする場合に保持されるオブジェクトのバージョン数
- アーカイブ・コピーを保持する期間
- オブジェクトを挿入するサーバーのストレージ階層内の位置

管理クラスは、バックアップ・コピー・グループとアーカイブ・コピー・グループの両者からなります。コピー・グループは、バックアップまたはアーカイブされるオブジェクトについて管理ポリシーを定義する属性のセットです。バックアップ操作が実行される場合は、バックアップ・コピー・グループ内の属性が適用されます。アーカイブ操作が実行される場合は、アーカイブ・コピー・グループ内の属性が適用されます。

特定の管理クラス内のバックアップまたはアーカイブ・コピー・グループは、空または NULL の場合があります。NULL バックアップ・コピー・グループにバインドされているオブジェクトを、バックアップすることはできません。オブジェクトが NULL アーカイブ・コピー・グループにバインドされている場合は、そのオブジェクトをアーカイブすることはできません。

ポリシーの使用は IBM Spectrum Protect の重要なコンポーネントであるため、API は、サーバーに送信されるすべてのオブジェクトがまず **dsmBindMC** 呼び出しを使用して、管理クラスを割り当てられることを必要とします。IBM Spectrum Protect ソフトウェアでは、include-exclude リストを使用して管理クラスのバインドを制御できます。**dsmBindMC** 呼び出しは、現行の include-exclude リストを使用して、管理クラスのバインドを実行します。

include ステートメントは、特定の管理クラスにバックアップまたはアーカイブ・オブジェクトに関連付けることができます。exclude ステートメントは、オブジェクトのバックアップがとられないようにできますが、アーカイブがされないようにはできません。

API を使用する場合、オブジェクトのバックアップまたはアーカイブの前に **dsmBindMC** の呼び出しが必要です。**dsmBindMC** 呼び出しは、オブジェクトに関連した管理クラスおよびコピー・グループについての情報が入っている **mcBindKey** 構造を返します。送信の前にコピー・グループの宛先を確認してください。単一トランザクションで複数のオブジェクトを送信する場合は、それらのオブジェクトのコピー・グループの宛先が同じでなければなりません。**dsmBindMC** 関数呼び出しは、以下の情報を返します。

表 9. **dsmBindMC** 呼び出しで戻される情報

通知	説明
管理クラス	これは、オブジェクトにバインドされた管理クラスの名前です。アプリケーション・クライアントは dsmBeginQuery を出して、この管理クラスのすべての属性を判別できます。
バックアップ・コピー・グループ	この管理クラスにバックアップ・コピー・グループが存在するかどうかを示します。バックアップ操作を実行しようとしているのにバックアップ・コピー・グループが存在しない場合は、このオブジェクトをストレージに送信できません。 dsmSendObj 呼び出しを使用して、このオブジェクトの送信を試みると、エラー・コードを受け取ります。
バックアップ・コピーの宛先	このフィールドは、データが送信される先のストレージ・プールを識別します。複数のオブジェクトのバックアップ・トランザクションを実行する場合、そのトランザクション内のすべてのコピー先を同じにする必要があります。トランザクション内のオブジェクトのコピー先が直前のオブジェクトのものと異なる場合は、現行トランザクションを終了し、新規トランザクションを開始してからでなければ、そのオブジェクトを送信できません。同じトランザクション内で異なるコピー宛先にオブジェクトを送信しようとすると、エラー・コードを受け取ります。

表 9. *dsmBindMC* 呼び出しで戻される情報 (続き)

通知	説明
アーカイブ・コピー・グループ	この管理クラスにアーカイブ・コピー・グループが存在するかどうかを示します。アーカイブ操作を実行しようとしているのにアーカイブ・コピー・グループが存在しない場合、このオブジェクトをストレージに送信できません。 dsmSendObj 呼び出しを使用して、このオブジェクトの送信を試みると、エラー・コードを受け取ります。
アーカイブ・コピーの宛先	このフィールドは、データが送信される先のストレージ・プールを識別します。複数のオブジェクトのアーカイブ・トランザクションを実行する場合、そのトランザクション内のすべてのコピー先を同じにする必要があります。トランザクション内のオブジェクトのコピー先が直前のオブジェクトのものと異なる場合は、現行トランザクションを終了し、新規トランザクションを開始してからでなければ、そのオブジェクトを送信できません。同じトランザクション内で異なるコピー宛先にオブジェクトを送信しようとすると、エラー・コードを受け取ります。

オブジェクトのバックアップ・コピーは、オリジナルと異なる管理クラスを使用して同じオブジェクト名でその後のバックアップが行われると、別の管理クラスに再バインドされることがあります。例えば、ObjectA をバックアップして Mgmtclass1 にバインドし、その後 ObjectA をバックアップして Mgmtclass2 にバインドすると、最新のバックアップですべての非活動のコピーが Mgmtclass2 に再バインドされます。Mgmtclass2 に定義されているパラメーターで、すべてのコピーが制御されるようになります。しかし、宛先が異なる場合、データは移動しません。

dsmUpdateObj または **dsmUpdateObjEx** 呼び出しと DSM_BACKUPD_MC アクションを使用して、バックアップ・コピーを別の管理クラスに再バインドすることもできます。

関連資料

[Deduplication オプション](#)

照会管理クラス

アプリケーションは、与えられたノードに対してどの管理クラスが使用可能であるかや、その管理クラス内の属性を判別するために、管理クラスを照会できます。

オブジェクトを管理クラスにバインドするには、**dsmBindMC** 呼び出しを使用します。ユーザーのアプリケーションは、管理クラス属性について照会し、エンド・ユーザーに表示することができます。詳しくは、33 ページの『[IBM Spectrum Protect システムの照会](#)』を参照してください。

30 ページの図 8 の例では、**dsmBindMC** を呼び出す際、バックアップ操作とアーカイブ操作を区別するために、switch ステートメントを使用しています。この呼び出しから戻される情報は、**MCBindKey** 構造内に保管されます。

```

dsUInt16_t    send_type;
dsUInt32_t    dsmHandle;
dsmObjName    objName;      /* structure containing the object name */
mcBindKey     MCBindKey;    /* management class information */
char          *dest;        /* save destination value */

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

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

```

図 8. 管理クラスとオブジェクトを関連付けする例

保留と保留解除の期限切れ/削除

特定のデータを保留にする必要のある保留アクションあるいは進行中のアクションに応じて、特定のアーカイブ・オブジェクトの削除および期限切れを保留することができます。データにアクセスする必要のあるアクションが開始された場合は、そのアクションが完了して、プロセスの一部としてのデータへのアクセスが不要になるまで、そのデータを使用可能にしておく必要があります。中断不要 (保留解除) と判別されると、通常の削除と期限切れのタイミングが、元の保存期間で再開されます。

始める前に

テストとしての **dsmRetentionEvent** 呼び出しを実行して、次のようにサーバーがライセンス 交付を受けていることを検証します。

1. 保留にする 1 つのオブジェクトについて照会し、ID を取得します。
2. **dsmBeginTxn**、Hold を指定した **dsmRetentionEvent**、および **dsmEndTxn** を実行します。
3. サーバーのライセンス 交付を受けていない場合、打ち切りと断定され、理由コード **DSM_RC_ABORT_LICENSE_VIOLATION** を受け取ります。

制約事項:

1. 1 つのトランザクションで、**dsmRetentionEvent** 呼び出しを複数回行うことはできません。
2. 既に保留状態であるオブジェクトに対して保留を出すことはできません。

手順

1. オブジェクトを保留するには、以下のステップを実行します。
 - a) 保留状態にするすべてのオブジェクトについて、サーバーに照会します。各オブジェクトのオブジェクト ID を取得します。
 - b) **dsmBeginTxn** 呼び出しを実行し、次にオブジェクトのリストを指定して **dsmRetentionEvent** 呼び出しを実行し、その後に **dsmEventType: eventHoldObj** 呼び出しを続けます。オブジェクト数が **maxObjPerTxn** の値を超える場合は、複数のトランザクションを使用します。

- c) **dsmGetNextQObj** 関数呼び出しで **qryRespArchiveData** 応答を使用し、オブジェクトが保留状態になっていることを確認します。 **qryRespArchiveData** の **objHeld** の値を調べます。
2. オブジェクトを保留状態から解放するには、以下のステップを実行します。
 - a) 保留状態を解除するすべてのオブジェクトについて、サーバーに照会します。各オブジェクトのオブジェクト ID を取得します。
 - b) **dsmBeginTxn** 呼び出しを実行し、次にオブジェクトのリストを指定して **dsmRetentionEvent** 呼び出しを実行し、その後に **dsmEventType: eventReleaseObj** 呼び出しを続けます。オブジェクト数が **maxObjPerTxn** の値を超える場合は、複数のトランザクションを使用します。
 - c) **dsmGetNextQObj** 関数呼び出しで **qryRespArchiveData** 応答を使用して、オブジェクトが保留状態から解放されているかどうかを確認します。 **qryRespArchiveData** の **objHeld** の値を調べます。

アーカイブ・データ保存保護

IBM Spectrum Protect 管理下のデータは、無許可のエージェント（個人またはプログラムなど）により変更できません。この保護が拡張され、保存期間の期限切れ前に、どのエージェントによってもデータ（アーカイブ・オブジェクトなど）が削除されないようにします。

このタスクについて

アーカイブ保存を保護することにより、個人やプログラムが IBM Spectrum Protect 管理下にあるデータを故意に、または誤って削除できないようにします。アーカイブ保存保護サーバーに送信されたアーカイブ・オブジェクトは、誤って削除されないように保護されており、保存期間が確実に守られます。アーカイブ保存保護には、以下の制約事項があります。

- 保存保護サーバーではアーカイブ操作のみが許可されています。
- **dsmBindMc** 関数呼び出しの値、または **INCLUDE** または **EXCLUDE** ステートメントを使用して明示的に管理クラスにバインドされていないオブジェクトは、デフォルト管理クラスの明示名にバインドされます。例えば、ノードのポリシーのデフォルト管理クラスが **MC1** の場合、オブジェクトは **DEFAULT** ではなく **MC1** に明示的にバインドされます。照会応答時、オブジェクトは **MC1** へのバインドとして表示されます。
- アーカイブ・データ保存保護を使用可能にした後は、保存期間の期限切れ前のオブジェクトを削除しようとすると、トランザクションの終了時にコード **DSM_RC_ABORT_DELETE_NOT_ALLOWED** が返されます。

アーカイブ・オブジェクトに対する保存保護の設定方法については、IBM Spectrum Protect サーバーの資料を参照してください。

手順

アーカイブ・データ保存保護を設定するには、次のステップを完了します。

1. 新規サーバー・インストール時に、前のデータがない状態で **SET ARCHIVERETENTIONPROTECTION ON** コマンドを実行します。
2. **dsmInit** または **dsmInitEx** 関数呼び出し時に API オプション・ストリングに次の指示に従って入力します。

```
-ENABLEARCHIVERETENTIONPROTECTION=yes
```

enablearchiveretentionprotection オプションを、UNIX 以外のシステムの **dsm.opt** ファイル内、または UNIX システムに **dsm.sys** ファイル内に次のように設定することもできます。

```
SERVERNAME srvr1.ret
TCPSPORT 1500
TCPSEVERADDRESS node.domain.company.com
COMMMETHOD TCPIP
ENABLEARCHIVERETENTIONPROTECTION YES
```

このオプションについて詳しくは、32 ページの『[enablearchiveretentionprotection オプション](#)』を参照してください。

3. IBM Spectrum Protect サーバーがアーカイブ 保存保護を使用可能にしていることを確認するため、サーバーに照会を出します。dsmQuerySessInfo 構造内の archiveRetentionProtection フィールドの値を確認してください。

enablearchiveretentionprotection オプション

enablearchiveretentionprotection オプションは、IBM Spectrum Protect サーバーでアーカイブ・オブジェクトのデータ保存保護を使用可能にするかどうかを指定します (このサーバーは、この目的のためにのみ使用されます)。サーバー管理者は、まだオブジェクト (バックアップ、アーカイブ、スペース管理) を保管していない新規のサーバーでデータ保存保護を活動化する必要があります。API アプリケーションがバックアップ・バージョンまたはスペース管理オブジェクトをこのサーバーに格納しようとすると、エラー・メッセージが出されます。

11 ページの『第 3 章 アプリケーションの設計に関する考慮事項』の注にある「今後のリストアで使うためにオブジェクト ID 値を保管しておくのはやめてください。オブジェクトの存続期間中、それらが永続的であることは保証されません」という指示は、Archive Manager アプリケーションの場合は無視してもかまいません。(Archive Manager サーバーはエクスポートまたはインポートをサポートしません。) Archive Manager アプリケーションでは、オブジェクト ID を保管して、オブジェクトのリストア時のパフォーマンスを向上させるために使用することができます。

サーバーが **SET ARCHIVERETENTIONPROTECTION ON** コマンドを発行する場合は、アーカイブ・コピー・グループのポリシー・パラメーターが満たされるまで、**delete filespace** コマンドを使用してサーバーからアーカイブ・オブジェクトを削除することはできません。管理クラスの設定方法については、該当するサーバーの資料を参照してください。

イベント・ベースの保存ポリシー

イベント・ベースの保存ポリシーでは、銀行口座の解約などのビジネス・イベントによってアーカイブ・オブジェクトの保存期間が開始されます。イベント・ベースの保存を使用すると、IBM Spectrum Protect データ保存ポリシーを、データのビジネス要件と密接に関連付けられます。イベントが発生すると、アプリケーションはそのオブジェクトの **eventRetentionActivate** イベントをサーバーへ送信し、保存を開始します。

手順

イベント・ベース保存ポリシーを使用するには、以下のステップを完了してください。

1. サーバー上に、アーカイブ **copygroup** にタイプ **EVENT** を指定して管理クラスを作成します。詳しくは、IBM Spectrum Protect サーバー資料を参照してください。
2. 管理クラスを照会して、その管理クラスがイベント・ベースであることを確認します。管理クラスがイベント・ベースである場合、**archDetailCG** 構造の **retainInit** フィールドは **ARCH_RETINIT_EVENT** です。
3. **dsmSendObj** 関数呼び出し時に、**include**、**archmc** を使用するか、または **ObjAttr** 構造の **mcNameP** 属性を明示的に使用して、オブジェクトをイベント・ベースの管理クラスにバインドします。
4. オブジェクトの保存を開始する時点で、影響を受けるすべてのオブジェクトについてサーバーを照会します。オブジェクトが **PENDING** 状態であるかどうかを確認して、オブジェクト ID を取得します。保留状態の場合、**qryRespArchiveData** 構造の **retentionInitiated** フィールドには **DSM_ARCH_RETINIT_PENDING** が示されます。
5. **dsmBeginTxn** 呼び出しを実行し、次にオブジェクトのリストを指定して **dsmRetentionEvent** 呼び出しを実行し、その後に **dsmEventType:eventRetentionActivate** 呼び出しを続けます。オブジェクト数が **maxObjPerTxn** の値を超える場合は、複数のトランザクションを使用します。

制約事項: トランザクションごとに **dsmRetentionEvent** 呼び出しを 1 回のみ発行できます。

6. オブジェクトを照会し、保存が活動化されているかどうか確認します。保存が開始されている場合、**qryRespArchiveData** 構造の **retentionInitiated** フィールドの値は **I** になります。

IBM Spectrum Protect システムの照会

API には、アプリケーションが使用できる管理クラス照会などの照会がいくつか用意されています。

手順

dsmBeginQuery 呼び出しを使用するすべての照会は、以下のステップに従ってください。

1. 以下のいずれかの照会タイプを指定して、**dsmBeginQuery** 呼び出しを送信する。

- バックアップ
- Archive
- 活動バックアップ・オブジェクト
- ファイル・スペース
- 管理クラス (Management class)

dsmBeginQuery 呼び出しにより、サーバーから戻されるデータ・フォーマットが API に通知されます。**dsmGetNextQObj** 呼び出しによって渡されるデータ構造に、適切な フィールドを置くことができます。**begin query** (照会開始) 呼び出しは、その **begin query** 呼び出しで適切にパラメーターを指定することにより、アプリケーション・クライアントが照会の有効範囲を設定できるようにします。

制約事項: UNIX または Linux システムでは、活動バックアップ・オブジェクトの照会を実行できるのは、root ユーザーのみです。この照会タイプは「ファスト・パス」と呼ばれています。

2. 照会から各レコードを取得するために、**dsmGetNextQObj** 呼び出しを入力する。この呼び出しは、照会から戻されたデータを保持するのに十分な大きさのバッファを渡します。各照会タイプは、戻されたデータ用に対応するデータ構造をもっています。例えば、バックアップ照会タイプは、**dsmGetNextQObj** 呼び出しが送信されたときに取り込まれる、関連した **qryRespBackupData** 構造をもっています。

3. **dsmGetNextQObj** 呼び出しは、通常、次のどれか 1 つのコードを戻します。

- DSM_RC_MORE_DATA: **dsmGetNextQObj** 呼び出しを再送信してください。
- DSM_RC_FINISHED: データはもうありません。 **dsmEndQuery** 呼び出しを送信してください。

4. **dsmEndQuery** 呼び出しを送信する。すべての照会データが検索されたか、追加の照会データが必要な場合には、**dsmEndQuery** 呼び出しを入力して照会処理を終了してください。API は、照会ストリームの残りのデータをフラッシュし、その照会で使用したすべてのリリースを解放します。

タスクの結果

34 ページの図 9 は、照会操作の状態遷移を示しています。

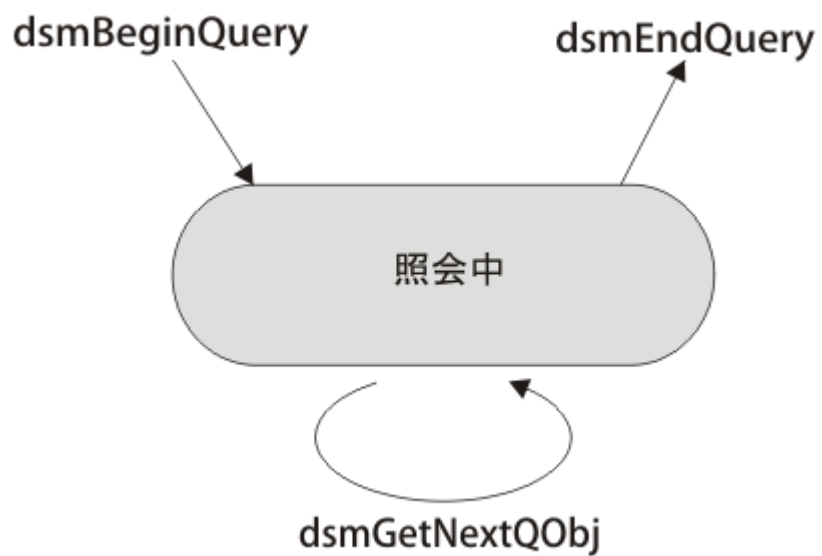


図 9. 一般照会の状態遷移

34 ページの図 10 は、照会操作のフローチャートを示しています。

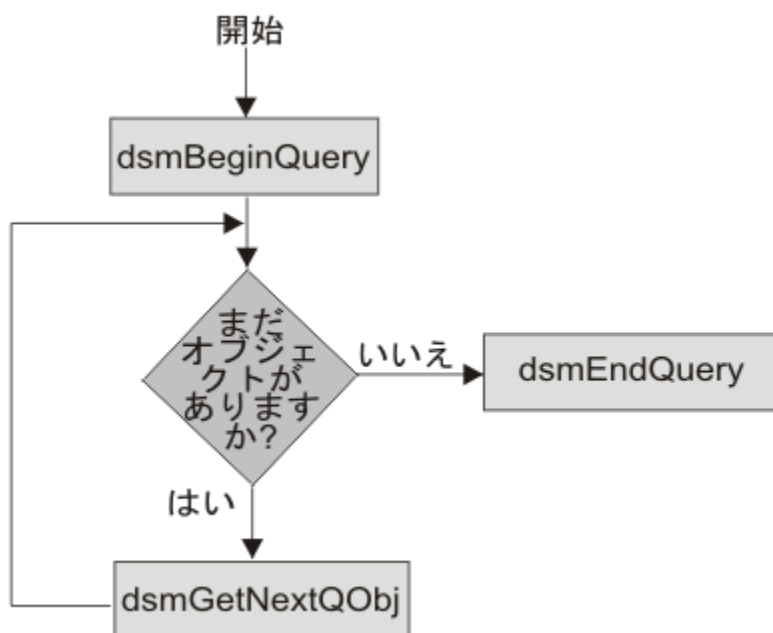


図 10. 一般照会のフローチャート

システム照会の例

この例では、管理クラス照会により、特定の管理クラスについて バックアップおよびアーカイブ・コピー・グループ内のすべての フィールドの値が印刷されます。

```
dsInt16          rc;
qryMCData        qMCData;
DataBlk          qData;
qryRespMCDetailData qRespMCData, *mcResp;
char             *mc, *s;
dsBool_t         done = bFalse;
dsUInt32_t       qry_item;

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

/* Set parameters of the data block used to get or send data */
qData.stVersion = DataBlkVersion;
qData.bufferLen = sizeof(qryRespMCDetailData);
qData.bufferPtr = (char *)&qRespMCData;

qRespMCData.stVersion = qryRespMCDetailDataVersion;

if ((rc = dsmBeginQuery
(dsmHandle, qtMC, (dsmQueryBuff *)&qMCData)))
{
    printf("*** dsmBeginQuery
failed: ");
    rcApiOut(dsmHandle, rc);
    rc = (RC_SESSION_FAILED);
}
else
{
    done = bFalse;
    qry_item = 0;
    while (!done)
    {
        rc = dsmGetNextQObj
(dsmHandle, &qData);
        if ((rc == DSM_RC_MORE_DATA)
            || (rc == DSM_RC_FINISHED))
            && qData.numBytes)
        {
            qry_item++;
            mcResp = (qryRespMCDetailData *)qData.bufferPtr;
            printf("Mgmt. Class
Name:
Backup CG Name:
. /* other fields of backup and archive copy groups */
printf(" Copy Destination:
}
else
{
    done = bTrue;
    if (rc != DSM_RC_FINISHED)
    {
        printf("*** dsmGetNextQObj
failed: ");
        rcApiOut(dsmHandle, rc);
    }
    if (rc == DSM_RC_FINISHED) done = bTrue;
}
    rc = dsmendQuery (dsmHandle);
}
}
```

図 11. システム照会を実行する例

サーバー効率

IBM Spectrum Protect サーバーからのリトリートを行うかまたはそのサーバーにオブジェクトを送信する場合は、以下の指針を使用します。

- IBM Spectrum Protect サーバーからオブジェクトをリトリートする場合は、以下の指針に従います。
 - IBM Spectrum Protect サーバーによって提供されるリストア順序でデータをリトリートします。リストア順序は磁気テープ装置にとって特に重要です。これは、順序付けられていないデータをリトリートすると、テープの巻き戻しおよびマウントが生じる可能性があるためです。
 - データがディスク装置に保管される場合でも、リトリートが順序付けられていると、時間を節約することができます。
 - 単一 IBM Spectrum Protect サーバー・セッションでできるだけ多くの作業を行います。
 - 複数のセッションを開始および停止してはなりません。
- オブジェクトを IBM Spectrum Protect サーバーに送信する場合は、以下の指針に従います。
 - 単一トランザクションで複数のオブジェクトを送信します。
 - トランザクションごとの 1 つのオブジェクトの送信は回避します (特に、データが磁気テープ装置に直接送信される場合)。磁気テープ装置のトランザクションの一部は、テープの RAM バッファ内のデータが確実にメディアに書き込まれるようにすることです。

関連概念

65 ページの『リストア順序によるオブジェクトの選択およびソート』

バックアップまたはアーカイブ照会が実行されたら、アプリケーション・クライアントは、どのオブジェクト (ある場合) をリストアまたはリトリートすべきかを判断する必要があります。

関連情報

17 ページの『セッションの開始または終了』

IBM Spectrum Protect はセッション・ベースの製品であり、すべての活動を IBM Spectrum Protect セッション内で実行する必要があります。セッションを開始するには、アプリケーションが **dsmInitEx** 呼び出しを開始します。この呼び出しは、**dsmQueryApiVersionEx**、**dsmQueryCliOptions**、または **dsmSetUp** 以外の他のすべての API 呼び出しの前に実行する必要があります。

サーバーへのデータの送信

API を使用することで、アプリケーション・クライアントは、データあるいは指定されたオブジェクトとその関連データを、IBM Spectrum Protect サーバー・ストレージに送信できます。

ヒント: データをバックアップすることもアーカイブすることもできます。すべての送信操作をトランザクション内で実行してください。

トランザクション・モデル

バックアップまたはアーカイブ操作中に IBM Spectrum Protect ストレージに送信されるデータは、すべてトランザクション内で送信されます。トランザクション・モデルにより、高水準のデータ保全性が提供されますが、アプリケーション・クライアントが考慮する必要のある、ある種の制限が生じることになります。

トランザクションは、**dsmBeginTxn** の呼び出しで開始し、**dsmEndTxn** の呼び出しで終了します。単一トランザクションは、分割不可能な処理です。あるトランザクションの境界の中で送信されたデータは、そのトランザクションの終了時にシステムにコミットされるか、またはトランザクションが早期に終了してしまった場合には、ロールバックされます。

トランザクションは、単一のオブジェクト送信からなる場合と複数のオブジェクト送信からなる場合があります。システム・オーバーヘッドを下げることによってシステム・パフォーマンスを向上させるには、小さいオブジェクトを複数オブジェクト・トランザクションで送信する必要があります。アプリケーション・クライアントは、単一トランザクションと複数トランザクションのどちらが適切かを判断します。

複数オブジェクト・トランザクション内のすべてのオブジェクトは、同じコピー宛先に送信してください。直前のオブジェクトの宛先とは異なる宛先にオブジェクトを送信する必要がある場合は、現行トランザク

ションを終了して新規トランザクションを開始する必要があります。新規トランザクション内では、オブジェクトを新しいコピー先に送信できます。

注: ビット・データを含んでいないオブジェクト (*sizeEstimate=0*) についてはコピー宛先の整合性はチェックされません。

IBM Spectrum Protect は、複数オブジェクト・トランザクションで送信できるオブジェクトの数を制限します。この限界値を知るには、**dsmQuerySessInfo** を呼び出して、**maxObjPerTxn** フィールドを調べてください。このフィールドには、ユーザーのサーバーに設定されている **TXNGroupmax** オプションの値が表示されます。

トランザクションが早期に終了した場合、アプリケーション・クライアントは再試行処理またはエラー処理を実行するために、トランザクション内で送信されたオブジェクトを追跡する必要があります。サーバーまたはクライアントのいずれかによって、いつでもトランザクションを停止することができます。アプリケーション・クライアントは、自ら開始したのではないトランザクションが突然終了した場合に対処できるよう準備する必要があります。

ファイルの集約

IBM Spectrum Protect サーバーは、「ファイルの集約」という機能を使用します。ファイル集合では、単一トランザクションで送信されるすべてのオブジェクトは、一緒に保管されます。これによって、スペースの節約とパフォーマンスの改善が行われます。これまでと同じように、オブジェクトを個別に照会したり、リストアすることができます。

この機能を使用するには、トランザクション内のすべてのオブジェクトが同じファイル・スペース名を持つようにする必要があります。トランザクション内でファイル・スペース名が同じでない場合には、サーバーは、既存の集合されたオブジェクトをクローズして、新規のオブジェクトを開始します。

LAN フリー・データ転送

API 使用時に、マルチスレッド化用の **dsmSetUp** オプションを ON に設定すると、LAN フリー・データ転送を利用できます。API は、**Query Mgmt Class** 応答構造の **archDetailCG** または **backupDetailCG** のフィールドの **bLanFreeDest** に、LAN フリー宛先の存在を返します。

ストレージ・エージェントによってサポートされるプラットフォーム上で LAN フリー操作を使用することができます。Macintosh オペレーティング・システムは除外されます。

LAN フリー情報は次の出力構造で提供されます。**dsmEndGetData** の出力構造 (**dsmEndGetDataExOut_t**) には、フィールド **totalLFBytesRecv** が含まれています。これは受信した LAN フリーの合計バイト数です。**dsmEndSendObjEx** の出力構造 (**dsmEndSendObjExOut_t**) には、フィールド **totalLFBytesSent** が含まれています。これは、送信された LAN フリーの合計数です。

関連情報

[LAN フリー・データ移動: ストレージ・エージェントの概要](#)

同時書き込み操作

IBM Spectrum Protect のサーバー・ストレージ・プールは、バックアップ時またはアーカイブ時に 1 次ストレージ・プールとコピー・ストレージ・プールに同時に書き込みが行われるように構成することができます。オブジェクトの複数のコピーを作成するには、この構成を使用します。

同時書き込み操作が失敗した場合、**dsmEndTxn** 関数の戻りコードが **DSM_RC_ABORT_STGPOOL_COPY_CONT_NO** となることがあります。これは、いずれかのコピー・ストレージ・プールへの書き込みが失敗したこと、および IBM Spectrum Protect のストレージ・プール・オプション **COPYCONTINUE** が NO に設定されていたことを示しています。アプリケーションは終了し、IBM Spectrum Protect サーバー管理者は、この問題を解決する必要があります。

同時書き込み操作の設定方法について詳しくは、IBM Spectrum Protect サーバーの資料を参照してください。

API のパフォーマンスの向上

API のパフォーマンスを向上させるには、`tcpbuffsize` と `tcpnodelay` のクライアント・オプション、および **DataBlk** API パラメーターを使用できます。

パフォーマンスの向上につながるバックアップ・アーカイブ・オプションと API パラメーターでは、API のパフォーマンスを向上させるために実行できるアクションを説明しています。

表 10. パフォーマンスの向上につながるバックアップ・アーカイブ・オプションと API パラメーター	
バックアップ・アーカイブ・クライアント・オプション	説明
<code>tcpbuffsize</code>	TCP バッファのサイズを指定します。デフォルト値は 31 KB です。パフォーマンスを向上させるには、値を 32 KB に設定します。
<code>tcpnodelay</code>	小規模のバッファを保持しないでサーバーに送信するかどうかを指定します。パフォーマンスを向上させるには、すべてのプラットフォームについて、このオプションを <code>yes</code> に設定します。このオプションは、Windows と AIX にのみ有効です。
API パラメーター	説明
DataBlk	このパラメーターは <code>dsmSendData</code> 関数呼び出しで使用され、アプリケーション・バッファ・サイズを決定します。最良の結果を得るには、このパラメーターを、 <code>tcpbuffsize</code> で指定する <code>tcpbuffsize</code> 値の倍数から 4 バイトを引いたサイズとして設定します。例えば、 <code>tcpbuffsize</code> の値を 32 KB (32,768 バイト) に設定する場合は、このパラメーターの値を 32,764 バイトに設定します。

各 `dsmSendData` 呼び出しは同期し、`dataBlkPtr` で API に転送されたデータがネットワークにフラッシュされるまで戻りません。API により、ネットワークに配置される各トランザクション・バッファに 4 バイトのオーバーヘッドが追加されます。

例えば、トランザクション・バッファ・サイズが 32 KB で、アプリケーション **DataBlk** バッファ・サイズが 31 KB の場合、各アプリケーション **DataBlk** バッファは通信バッファに収まるので、直ちにフラッシュすることができます。しかし、アプリケーションの **DataBlk** バッファがちょうど 32 KB の場合、API でトランザクション・バッファごとに 4 バイトのオーバーヘッドが追加されるので、32 KB と 4 バイトの 2 回のフラッシュが行われます。また、`tcpnodelay` オプションを `no` に設定すると、4 バイトのフラッシュに最大 200 ミリ秒を要する場合があります。

パフォーマンス・データをクライアント・パフォーマンス・モニターに送信するための API の設定

クライアント・パフォーマンス・モニターは、Tivoli® Storage Manager 管理センターのコンポーネントで、API によって収集されたパフォーマンス・データを表示するために使用されます。クライアント・パフォーマンス・モニターは、クライアント・バックアップ、アーカイブ、およびリストアなどの操作のパフォーマンス・データを記録し、表示します。

パフォーマンス・モニターを有効にすると、パフォーマンス・モニターを使用して API に収集されたパフォーマンス・データを表示することができます。パフォーマンス・モニターは、Tivoli Storage Manager 管理センターで使用できます。バージョン 7.1 以降、管理センター・コンポーネントは、Tivoli Storage Manager あるいは IBM Spectrum Protect のディストリビューションに含まれなくなりました。以前のサーバー・リリースでインストールされた管理センターがある場合には、引き続きその管理センターを使用してパフォーマンス・データを表示することができます。管理センターをまだインストールしていない場合は、前にリリースされたバージョンを <http://public.dhe.ibm.com/storage/tivoli-storage-management/maintenance/admincenter/v6r3/> からダウンロードできます。パフォーマンス・モニターの使用方法について詳しくは、Tivoli Storage Manager バージョン 6.3 サーバーの資料を参照してください。

クライアント・パフォーマンス・モニター・オプションの構成

クライアント・オプション・ファイルにパラメーターを指定して、パフォーマンス・モニターを使用するように IBM Spectrum Protect クライアントを有効にします。以下のオプションをモニターするクライアントごとに指定します。

始める前に

UNIX コンピューターおよび Linux コンピューターでパフォーマンスをモニターする場合は、以下のコマンドを使用してオープン・ファイル記述子制限を 1024 以上に設定します。

```
ulimit -n 1024
```

手順

クライアント・パフォーマンス・モニター・オプションを構成するには、以下のステップを実行します。

1. モニターしているクライアントごとにクライアント・オプション・ファイルを開きます。クライアント・オプション・ファイルは、構成に応じて以下のファイルのどちらかになります。

- dsm.opt
- dsm.sys

2. クライアント・オプション・ファイルに次のオプションを追加します。

```
PERFMONTCPSERVERADDRESS
```

```
PERFMONTCPPORT
```

```
PERFMONCOMMTIMEOUT
```

PERFMONTCPSERVERADDRESS

PERFMONTCPSERVERADDRESS オプションは、クライアント・パフォーマンス・モニターがインストールされているシステムの ホスト名または IP アドレスを指定します。

サポートされるクライアント

このオプションは、プラットフォームから独立しており、すべてのクライアントに対してサポートされます。

オプション・ファイル

このオプションは、クライアント・オプション・ファイル (dsm.opt または dsm.sys) で指定してください。

構文

```
➡ PERFMONTCPServeraddress server ➡
```

パラメーター

server

クライアント・パフォーマンス・モニターがインストールされているシステムのサーバー・ホスト名、または IP アドレス (これは Administration Center を稼働する同一のサーバーです)。

例

オプション・ファイル:

```
PERFMONTCPSERVERADDRESS 131.222.10.5
```

コマンド・ライン:

このオプションは、コマンド・ラインを使用して設定することはできません。

PERFMONTCPPORT

クライアント・パフォーマンス・モニターがクライアントからのパフォーマンス・データを listen するポート番号。

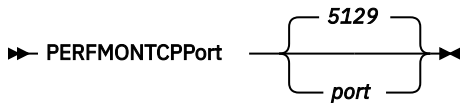
サポートされるクライアント

このオプションは、プラットフォームから独立しており、すべてのクライアントに対してサポートされます。

オプション・ファイル

このオプションは、クライアント・オプション・ファイル (dsm.opt または dsm.sys) で指定してください。

構文



パラメーター

port

クライアント・パフォーマンス・データ用にモニターされるポート。ポート 5129 は、デフォルトのポートです。

例

オプション・ファイル:

```
PERFMONTCPPPORT 5000
```

コマンド・ライン:

このオプションは、コマンド・ラインを使用して設定することはできません。

PERFMONCOMMTIMEOUT

セッションが終了した後、dsmTerminate 呼び出しがパフォーマンス・データが到着するのを待機する 最大時間 (秒数) を指定します。

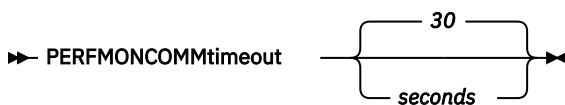
サポートされるクライアント

このオプションは、プラットフォームから独立しており、すべてのクライアントに対してサポートされます。

オプション・ファイル

このオプションは、クライアント・オプション・ファイル (dsm.opt または dsm.sys) で指定してください。

構文



パラメーター

seconds

セッションを終了する前に、残りのパフォーマンス・データが到着するのを待機する時間。

例

オプション・ファイル:
PERFMONCOMMTIMEOUT 60

コマンド・ライン:
このオプションは、コマンド・ラインを使用して設定することはできません。

サーバーへのオブジェクトの送信

アプリケーション・クライアントは、データあるいは指定されたオブジェクトとその関連データを、APIのバックアップおよびアーカイブ機能を使用して IBM Spectrum Protect ストレージに送信できます。システムのバックアップとアーカイブのコンポーネントによって、ストレージに送信するデータに、さまざまな管理手順を使用することができます。

サイズ見積属性は、サーバーに送信するデータ・オブジェクトの合計サイズの見積もりです。アプリケーションが正確なオブジェクト・サイズを知らない場合は、*sizeEstimate* をより高い見積もりに設定します。見積もりが実際のサイズより小さいと、IBM Spectrum Protect サーバーは、追加のスペース割り振りを管理するために追加のリソースを使用します。

ヒント:

- このサイズ見積もりは、可能な限り正確に出してください。サーバーは、そのストレージ・リソース内でのスペース割り振りおよびオブジェクトの配置を効率よく行うために、この属性を使用します。
- 見積もりが実際のサイズより小さいと、キャッシングを行うサーバーは、追加のスペース割り振らないので、この送信を停止します。

sizeEstimate があまりに大きい場合は、問題が起こる可能性があります。サーバーに、見積サイズ分のスペースはなくても、実際のサイズ分のスペースはある場合もあります。または、サーバーが低速デバイスを使用していることがあります。

サイズが 2 ギガバイトより大きいオブジェクトをバックアップまたはアーカイブすることができます。オブジェクトは圧縮しても圧縮しなくてもかまいません。

送信操作を開始するには、**dsmSendObj** を呼び出してください。一度に送信できないほど多くのデータがある場合は、**dsmSendData** の呼び出しを繰り返し行って、残りの情報を転送できます。送信操作を終了するときは、**dsmEndSendObj** を呼び出してください。

バックアップ・オブジェクトおよびアーカイブ・オブジェクトの把握

IBM Spectrum Protect システムのバックアップ・コンポーネントは、サーバーに保管される名前付きのオブジェクトについて、複数のバージョンをサポートします。

既にサーバーに保管されているオブジェクトと同じ名前をもつオブジェクトが当該クライアントからサーバーにバックアップされると、それらのオブジェクトはすべてバージョン制御の対象となります。オブジェクトはサーバー上で活動状態または非活動状態であると見なされます。非活動にされていないサーバー上のオブジェクトの最新コピーが、活動状態です。その他のオブジェクトは、すべて、より古いバージョンも、非活動にされているコピーも、非活動であるとは見なされます。管理クラス構成は、さまざまな管理基準を定義します。これらの管理基準は、サーバー上の活動オブジェクトと非活動オブジェクトに割り当てられます。

41 ページの表 11 に、活動状態と非活動状態に該当するコピー・グループのフィールドを示します。

表 11. バックアップ・コピー・グループのフィールド	
フィールド	説明
VEREXISTS	活動バージョンが存在する場合の非活動バージョンの数。
VERDELETED	活動バージョンが存在しない場合の非活動バージョンの数。
RETEXTRA	非活動バージョンを保持する日数。
REONLY	活動バージョンが存在しない場合の最新非活動バージョンを保持する日数。

各バックアップ・バージョンの名前が固有になっている場合 (例えば、名前の中に タイム・スタンプを使用している場合など) は、バージョン管理は自動的に は行われず、すべてのオブジェクトが活動バージョンとなります。活動オブジェクト は期限切れにはならないので、アプリケーション で **dsmDeleteObj** 呼び出しを使用して、これらを非活動にする 必要があります。このような場合、アプリケーションは、非活動にした オブジェクトをできるだけ早く期限切れにする必要がある場合もあります。ユーザーはバックアップ・コピー・グループを VERDELETED=0 および RETONLY=0 で定義します。

IBM Spectrum Protect システムのアーカイブ・コンポーネントを使用すれば、バージョン制御の代わりに保存期間 制御または有効期限制御を指定して、サーバーにオブジェクトを保管できます。保管する 各オブジェクトは、その名前が既にアーカイブされた名前と同じであっても、固有のものであると見なされます。アーカイブ・オブジェクトには、照会時に特定のオブジェクトを識別するために使用できる、メタデータに関連付けられた説明 フィールドがあります。

IBM Spectrum Protect サーバー上のすべてのオブジェクトには、固有のオブジェクト ID が割り当てられています。元の値がオブジェクトの存続中は保証されるということはありません (特に、エクスポートまたはインポート後)。したがって、アプリケーションは、元のオブジェクト ID を、後のリストアで使用するために 照会したり、保管すべきではありません。むしろアプリケーションは、オブジェクト名を保管して日付を挿入すべきです。リストア中にこの情報を使用してオブジェクトを照会し、挿入日付を検証することができます。その後、現行のオブジェクト ID を使用してオブジェクトをリストアすることができます。

圧縮

所定のノードの構成オプションと **dsmSendObj** objCompressed オプションによって、IBM Spectrum Protect で送信時にオブジェクトが圧縮されるかどうかが決まります。また、DSM_MIN_COMPRESS_SIZE より小さい sizeEstimate を指定したオブジェクトは圧縮されません。

既に圧縮されているオブジェクト (objCompressed=bTrue) が、再び圧縮されることはありません。圧縮されていない場合、IBM Spectrum Protect は、管理者によって設定され、API 構成ソース内で設定された compression (圧縮) オプションの値に基づいて、オブジェクトを圧縮するかどうかを決定します。

管理者は、ノードの登録 (register node) コマンド (compression=yes、no、または client-determined) を使用してサーバー上の圧縮しきい値を変更できます。これが client-determined になっている場合、圧縮動作は構成ソース内の圧縮オプション値によって決まります。

データのタイプ (例えば、既に圧縮されているデータなど) によっては、圧縮アルゴリズムを使用して処理した場合に、実際にはサイズが増大する 可能性があります。こうしたことが起こると、戻りコード DSM_RC_COMPRESS_GREW が生成されます。こうしたことが起こる可能性が分かっている、やはり送信操作を続行したい場合は、エンド・ユーザーのそれぞれのオプション・ファイルに、次のオプションを指定してからアプリケーションを実行するよう指示します。

COMPRESSAlways Yes

圧縮を使用可能にして **dsmSendData** 関数を実行したときに、DSM_RC_COMPRESS_GREW 戻りコードを受け取った場合は、操作をやり直して、圧縮なしでオブジェクトを再送信することもできます。これを強制的に行うには、**dsmSendObj** ObjAttr.objCompressed を bTrue に設定します。

dsmSendObj 時の実際の圧縮動作についての情報は、**dsmEndSendObjEx** 呼び出しで戻されます。objCompressed は圧縮が行われた場合に指定されます。totalBytesSent は、アプリケーションで送信されたバイト数です。totalCompressedSize は、圧縮後のバイト数です。**dsmEndSendObjEx** 呼び出しには、LAN フリーで送信された合計バイト数が入る totalLFBytesSent フィールドもあります。



重要: アプリケーションで部分オブジェクト・リストア/リトリブを計画している場合には、データの送信中にデータを圧縮することはできません。これを強制的に行うには、**dsmSendObj** ObjAttr.objCompressed を bTrue に設定します。

圧縮タイプ

クライアントが使用する圧縮のタイプは、バックアップまたはアーカイブの処理中に使用される圧縮とクライアント・サイドのデータ重複排除の有無の組み合わせによって決まります。

クライアントにより使用される圧縮アルゴリズムは、**qryRespArchiveData** ストラクチャーと **qryRespBackupData** ストラクチャーの新規フィールドに API によって報告されます。

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

以下のタイプの圧縮が報告されます。

LZ4

以下のいずれかの状況でクライアントが使用する高速で効率的な圧縮方式です。

- ・ クライアント・サイドの重複排除オブジェクトが、IBM Spectrum Protect サーバーのコンテナ・ストレージ・プールに送信される場合。サーバーのバージョンは 7.1.5 以降でなければなりません。
- ・ オブジェクトがクライアント・サイドのデータ重複排除を実行していない場合。
- ・ オブジェクトが従来のサーバー・サイドのデータ重複排除のみを実行している場合。

LZW

クライアントが使用する従来型の圧縮は、クライアントにより重複除外されたオブジェクトがサーバーの従来の (非コンテナ) ストレージ・プールに送信される場合に使用されます。

ブランク・フィールド

オブジェクトはクライアントによって圧縮されません。compression オプションが no に設定されているか、バックアップまたはアーカイブの処理中に指定されていないために、オブジェクトは圧縮されません。オブジェクトはクライアントによって圧縮されませんが、サーバーによって圧縮される可能性があります。

圧縮タイプが構成不可能です。圧縮タイプは、バックアップまたはアーカイブの処理時にバックアップ/アーカイブ・クライアントによって判別されます。

例

以下の例は、64 ビットのサンプル・アプリケーション **dapismp** のバックアップおよびアーカイブの照会の出力に含まれる Compresssion Type フィールドを示します。

```
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

バッファ・コピー除去

バッファ・コピー除去機能を使用すると、アプリケーションと IBM Spectrum Protect サーバーの間でのデータ・バッファのコピーが不要になります。その結果、プロセッサ使用率が向上します。効果を最大限に高めるため、この手法は LAN フリー環境で使用してください。

データ移動用のバッファは IBM Spectrum Protect によって割り振られ、アプリケーションにはポインターが返されます。アプリケーションは提供されたバッファにデータを入れ、そのバッファは、通信層を経由して (共有メモリーを使用して) ストレージ・エージェントに渡されます。その後、渡されたデータは磁気テープ装置に移動されます。このようにして、データのコピーを不要にしています。バックアップ操作とアーカイブ操作のどちらでも、この機能を使用できます。



重要: この方式を使用する場合は、バッファの取り扱いとバッファのサイズ設定が適切に行われるように特に注意する必要があります。バッファはコンポーネント間で共用されるので、プログラミング・エラーのためにメモリーが上書きされると、重大なエラーが発生することになります。

バックアップ/アーカイブを呼び出す全体的な順序は、次のようになります。

```
dsmInitEx (UseTsmBuffers = True, numTsmBuffers = [how many IBM Spectrum Protect
-allocated buffers the application needs to allocate])
dsmBeginTxn
txn 内の各オブジェクトごとに
    dsmBindMC
    dsmSendObject
    dsmRequestBuffer
    dsmSendBufferData (使用したバッファの送信と解放)
    dsmEndSendObjEx
dsmEndTxn
まだ保留中の各バッファごとに
    dsmReleaseBuffer
dsmTerminate
```

dsmRequestBuffer 関数は、numTsmBuffers オプションで指定された値まで、何回でも呼び出すことができます。アプリケーションは 2 つのスレッドを持ちます。一方はバッファにデータを入れる作成元スレッドであり、もう一方は **dsmSendBufferData** 呼び出しでそれらのバッファを IBM Spectrum Protect に送信する消費先スレッドです。 **dsmRequestBuffer** 呼び出しが発行され、 **numTsmBuffers** に到達している場合、 **dsmRequestBuffer** 呼び出しは、バッファが解放されるまでブロックされます。バッファが解放されるのは、 **dsmSendBufferData** (バッファの内容を送信してバッファを解放する) を呼び出したときか、 **dsmReleaseBuffer** を呼び出したときです。詳しくは、API サンプル・ディレクトリーの `callbuff.c` を参照してください。

どの時点でも、送信に失敗した場合、アプリケーションは保留しているすべてのバッファを解放し、セッションを終了する必要があります。例えば次のとおりです。

```
失敗した場合
for each data buffer held by application
    call dsmReleaseBuffer
dsmTerminate
```

アプリケーションが **dsmTerminate** を呼び出しても、バッファがまだ保留されていると、API は終了しません。DSM_RC_CANNOT_EXIT_MUST_RELEASE_BUFFER というコードが返されます。アプリケーションでは、バッファを解放できない場合、プロセスを終了して強制的にクリーンアップを行う必要があります。

バッファ・コピー除去とリストアおよび検索

IBM Spectrum Protect サーバーは、リストアとリトリーブでのテープ・アクセスの最適化に基づいて、バッファに入れるデータの量を制御します。この方式は、アプリケーションにとって、通常のデータ取得方式ほど有利ではありません。プロトタイピングのときにバッファ・コピー除去方式のパフォーマンスを検査し、価値がある改善が見られる場合にのみ、この方法を使用してください。

IBM Spectrum Protect サーバーから返される単一バッファの最大データ量は、256 K バイトです (ヘッダー・オーバーヘッドを含む)。このため、小規模なバッファ書き込みを処理するアプリケーションだけが、このデータ取得メカニズムの恩恵を受けます。オブジェクト・サイズ、ネットワーク、およびその他の境界条件によっては、アプリケーションでバッファ内のバイト数に特別な注意を払う必要があります。バッファ・コピー除去を使用すると、場合によっては、通常のリストアを行う場合よりもパフォーマンスが低下する可能性があります。API は通常、データをキャッシュし、一定量ずつアプリケーションに返します。これにより、アプリケーションでは、ディスクへのデータ書き戻しの回数を制御することができます。

バッファ・コピー除去を使用する場合は、優先される書き込みバッファ・サイズに満たないバッファのデータ・キャッシング・メカニズムを作成します。例えば、アプリケーションが 64 K のデータ・ブロックをディスクに書き込む場合、そのアプリケーションでは以下のアクションを実行する必要があります。

1. **dsmGetBufferData** を呼び出す。
2. 64 K のブロックを書き出す。
3. 最終ブロックで、残ったデータを **tempBuff** にコピーし、別の **dsmGetBufferData** 呼び出しを発行し、**tempBuff** を残りのデータで満たす。
4. 64 K のブロックの書き込みを続行する。

```
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
```

この例では、6 回のディスク書き込みが直接書き込みで、1 回はキャッシュ後の書き込みです。

リストアとリトリブの呼び出しの全体的な順序は、次のようになります。

dsmInitEx (UseTsmBuffers = True numTsmBuffers = アプリケーションが割り振るバッファの数)。

```
dsmBeginGetData
obj id で実行中
  dsmGetObj (この呼び出しで、データはリストアされない
    - バッファは NULL に設定される)
  データの読み取り中
    dsmGetBufferData (returns the data in the data buffer)
    ... データの処理 ...
  dsmReleaseBuffer
dsmEndGetObj
dsmEndGetData
```

どの **dsmGetBufferData** 呼び出しにも、対応する **dsmReleaseBuffer** 呼び出しを実装します。

dsmReleaseBuffer は、対応する **dsmGetBufferData** の直後に呼び出さなくてもかまいません。アプリケーションでは、複数のバッファを取得するために先に **dsmGetBufferData** 呼び出しを複数回実行してから、対応する複数の **dsmReleaseBuffer** 呼び出しを実行することができます。この関数を使用するサンプル・コードについては、API サンプル・ディレクトリーの **callbuff.c** を参照してください。

制約事項: API はバッファを提供し、最終目標はプロセッサ使用率を最小化することなので、バッファ内のデータの処理が増えることは許されません。アプリケーションでは、バッファ・コピー除去と一緒に暗号化や圧縮を使用することはできません。そのどちらの操作でも、データの処理とコピーが必要になるからです。

通常データ移動パスとバッファ・コピー除去の両方のパスを実装しておき、ユーザーがこの 2 つのパスを必要に応じて切り替えて使用できるようにしてください。データを圧縮または暗号化する場合がある場合は、従来のメカニズムを使用します。プロセッサの制約がある場合は、新しいメカニズムを使用します。これら 2 つのメカニズムは補完し合うものです。一方が他方に完全に置き換わるものではありません。

API 暗号化

データを暗号化するために 2 つの方式 (アプリケーション管理暗号化と IBM Spectrum Protect クライアント暗号化) を使用できます。

データの暗号化には、これらの方式のどちらか 1 つのみを選択して使用してください。これらの方式を同時に使用することはできません。両方の方式を使用してデータを暗号化すると、一部のデータのリストアおよびリトリブが不可能になります。例えば、あるアプリケーションで、アプリケーション管理暗号化を使用してオブジェクト A を暗号化し、IBM Spectrum Protect クライアント暗号化を使用してオブジェクト B を暗号化すると想定します。リストア操作時にそのアプリケーションで IBM Spectrum Protect クライアント暗号化を使用するオプションを設定し、両方のオブジェクトをリストアしようとすると、オブジェクト B のみがリストアされます。オブジェクト A は、クライアントではなくアプリケーションによって暗号化されたためリストアできません。

使用されている暗号化方式とは関係なく、IBM Spectrum Protect はパスワード認証を有効にする必要があります。デフォルトでは、サーバーは SET AUTHENTICATION ON を使用します。

API は AES 128 ビット暗号化または AES 256 ビット暗号化のどちらかを使用します。AES 256 ビット・データ暗号化は、AES 128 ビット・データ暗号化よりも高いレベルのデータ暗号化を提供します。AES 256 ビットの暗号化を使用してバックアップしたファイルは、旧レベルのクライアントではリストアできません。暗号化は圧縮ありまたは圧縮なしで使用できます。暗号化を使用する場合、部分オブジェクト・リストアとリトリブおよびバッファ・コピー除去機能は使用できません。

アプリケーション管理暗号化

アプリケーション管理暗号化では、アプリケーションから API に鍵パスワードを提供し (鍵 DSM_ENCRYPT_USER を使用)、アプリケーションが鍵パスワードを管理します。



重要: 暗号鍵を保管せずに暗号鍵を忘れてしまうと、データはリカバリー不能です。

アプリケーションからの鍵パスワードの提供には **dsmInitEx** 呼び出しを使用します。リストア時には、アプリケーション側から正しい鍵パスワードを提示しなければなりません。



重要: 鍵パスワードが失われると、データをリストアする方法はなくなります。

操作対象のオブジェクトが同じである限り、バックアップとリストア (またはアーカイブとリトリブ) には同一の鍵パスワードを使用しなければなりません。この方法を使用できるかどうかは、IBM Spectrum Protect サーバーのレベルとは関係がありません。この方法をセットアップするには、アプリケーションで以下の手順を実行する必要があります。

1. **dsmInitEx** 呼び出しで `bEncryptKeyEnabled` 変数を `bTrue` に設定し、また `encryptionPasswordP` 変数が暗号鍵パスワードのストリングを指すように設定します。
2. `include.encrypt` を設定して、オブジェクトを暗号化します。例えば、すべてのデータを暗号化する場合は、次のように設定します。

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

および

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

オブジェクト /FS1/DB2/FULL を暗号化する場合は、次のように設定します。

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

3. Windows では、**dsmInitEx** 呼び出しで API に渡されるオプション・ストリングに `ENCRYPTKEY=PROMPT|SAVE` を指定します。このオプションは、`dsm.opt` (Windows) または `dsm.sys` (UNIX または Linux) でも設定できます。

encryptkey オプションのデフォルトは prompt です。この設定は、鍵が自動的に保管されないようにするためです。encryptkey save が指定されていると、鍵は IBM Spectrum Protect によってローカル・マシンに保管されますが、ノード名が同じすべての IBM Spectrum Protect 操作で有効になるのは 1 つの鍵だけです。

オブジェクトの送信後に **dsmEndSendObjEx** で、オブジェクトが暗号化されたかどうかと、どの暗号化方式を使用したかを指定します。encryptionType フィールドに指定できる値は、次のとおりです。

- DSM_ENCRYPT_NO
- DSM_ENCRYPT_USER
- DSM_ENCRYPT_CLIENTENCRKEY

次の表に、API での暗号化のタイプ、前提条件、および使用可能な機能を示します。

表 12. API での暗号化のタイプ、前提条件、および使用可能な機能		
タイプ	前提条件	使用可能な機能
ENCRYPTIONTYPE	None	Windows では、 dsmInitEx 呼び出しで API に渡されるオプション・ストリングに ENCRYPTIONTYPE を指定します。デフォルトでは ENCRYPTIONTYPE=AES128 です。
EncryptKey=save	None	API およびバックアップ/アーカイブ
EncryptKey=prompt	None	API およびバックアップ/アーカイブ
EncryptKey=generate	None	API およびバックアップ/アーカイブ
EnableClientEncryptKey	None	API のみ

注：サーバー認証を ON にすることを推奨します。認証が OFF になっていると、鍵が暗号化されません（ただし、この場合もデータは暗号化されます）。ただし、これはお勧めできません。

47 ページの表 13 に、バックアップ操作時やリストア操作時にデータの暗号化と暗号化解除がどのように行われるかを、許可ユーザーの場合と非許可ユーザーの場合の両方について、passwordaccess オプションの指定値ごとに示します。以下の許可ユーザー操作および非許可ユーザー操作を実行するには、TSM.KDB、TSM.sth、および TSM.IDX ファイルが存在する必要があります。許可ユーザーは、encryptkey オプションを **save** に設定し、passwordaccess オプションを **generate** に設定します。

表 13. アプリケーション管理の鍵を使用したデータの暗号化および暗号化解除 (UNIX または Linux)			
操作	passwordaccess オプション	encryptkey オプション	結果
許可ユーザーによるバックアップ	generate	save	データは暗号化されます。
	generate	prompt	encryptionPasswordP に暗号化パスワードが入っていれば、データは暗号化されます。
	prompt	save	encryptionPasswordP に暗号化パスワードが入っていれば、データは暗号化されます。
	prompt	prompt	encryptionPasswordP に暗号化パスワードが入っていれば、データは暗号化されます。
許可ユーザーによるリストア	generate	save	データは暗号化されます。
	generate	prompt	encryptionPasswordP に暗号化パスワードが入っていれば、データは暗号化されます。
	prompt	save	encryptionPasswordP に暗号化パスワードが入っていれば、データは暗号化されます。
	prompt	prompt	encryptionPasswordP に暗号化パスワードが入っていれば、データは暗号化されます。

表 13. アプリケーション管理の鍵を使用したデータの暗号化および暗号化解除 (UNIX または Linux) (続き)

操作	passwordaccess オプション	encryptkey オプション	結果
非許可ユーザー によるバックア ップ	generate	save	データは暗号化されます。
	generate	prompt	encryptionPasswordP に暗号化パスワードが入 ていれば、データは暗号化されます。
	prompt	save	encryptionPasswordP に暗号化パスワードが入 ていれば、データは暗号化されます。
	prompt	prompt	encryptionPasswordP に暗号化パスワードが入 ていれば、データは暗号化されます。
非許可ユーザー によるリストア	generate	save	データは暗号化されます。
	generate	prompt	encryptionPasswordP に暗号化パスワードが入 ていれば、データは暗号化されます。
	prompt	save	encryptionPasswordP に暗号化パスワードが入 ていれば、データは暗号化されます。
	prompt	prompt	encryptionPasswordP に暗号化パスワードが入 ていれば、データは暗号化されます。

IBM Spectrum Protect クライアント暗号化

IBM Spectrum Protect クライアント暗号化は、DSM_ENCRYPT_CLIENTENCRKEY 値に管理されるキーを使用してデータを保護します。クライアント暗号化は、API を使用しているアプリケーションに対しては透過的です。ただし、例外があり、部分オブジェクト・リストア操作およびリトリブ操作は、暗号化または圧縮されているオブジェクトに対して実行できません。

IBM Spectrum Protect クライアント暗号化とアプリケーション管理暗号化のどちらの場合も、暗号化パスワードは、実際の暗号鍵を生成するために使用されるストリング値を参照します。暗号化パスワード・オプションの値は、1 文字から 63 文字の長さに指定することができますが、生成される鍵は常に、56 DES の場合で 8 バイト、128 AES の場合で 16 バイト、256 AES の場合で 32 バイトです。



重要: 暗号鍵が利用不能の場合は、データのリストアまたはリトリブができません。暗号化に ENABLECLIENTENCRYPTKEY を使用する場合は、暗号鍵がサーバー・データベースに保管されます。この方法を使用するオブジェクトの場合はサーバー・データベースが必要であり、適正なリストアのために、オブジェクトに適切な値を設定する必要があります。データ損失を防ぐため、サーバー・データベースは頻繁に必ずバックアップしてください。

これは、セッションごとに 1 つのランダムな暗号鍵が生成され、IBM Spectrum Protect サーバー上のサーバー・データベースにオブジェクトとともに保管されるという方式であり、容易にインプリメントできます。リストア時には、保管されていた鍵を使用して暗号化解除が行われます。この方式を使用する場合、鍵の管理を行うのは IBM Spectrum Protect です。アプリケーションが鍵を管理する必要はまったくありません。サーバー・データベースに鍵が保管されるので、暗号化されているオブジェクトのリストア操作のためには、有効な状態の IBM Spectrum Protect データベースが必要です。鍵が API とサーバーの間で送信されるときには、鍵も暗号化されます。鍵の送信は保護され、その鍵が IBM Spectrum Protect のサーバー・データベース内に保管されるときは暗号化されています。鍵が平文のままエクスポート・データ・ストリームに送られるのは、サーバー間でノードのデータがエクスポートされるときだけです。

IBM Spectrum Protect クライアント暗号化を有効にするには、以下の手順を実行します。

1. dsmInitEx 呼び出しで API に渡すオプション・ストリングに -ENABLECLIENTENCRYPTKEY=YES を指定するか、あるいはシステム・オプション・ファイル dsm.opt (Windows) または dsm.sys (UNIX または Linux) にこのオプションを指定します。
2. include.encrypt を設定して、オブジェクトを暗号化します。例えば、すべてのデータを暗号化する場合は、次のように設定します。

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

および

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

オブジェクト /FS1/DB2/FULL を暗号化する場合は、次のように設定します。

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

データ重複排除

データ重複排除は、冗長データを除去することでストレージ必要量を削減する方法です。

概説

IBM Spectrum Protect では、クライアント・サイドのデータ重複排除とサーバー・サイドのデータ重複排除の2つのタイプのデータ重複排除を使用できます。

クライアント・サイドのデータ重複排除は、バックアップおよびアーカイブ処理中、データが IBM Spectrum Protect サーバーに転送される前に、バックアップ/アーカイブ・クライアントが冗長データを削除するために使用するデータ重複排除技法です。クライアント・サイドのデータ重複排除を使用すると、ローカル・エリア・ネットワーク上で送信されるデータの容量を削減できます。

サーバー・サイド・データ重複排除は、サーバーによって行われるデータ重複排除手法です。IBM Spectrum Protect の管理者は、**REGISTER NODE** または **UPDATE NODE** サーバー・コマンドの **DEDUP** パラメーターと併用するデータ重複排除ロケーション (クライアントまたはサーバー) を指定することができます。

機能拡張

クライアント・サイドのデータ重複排除では、以下の操作を実行できます。

- ・クライアント上の特定のファイルをデータ重複排除から除外します。
- ・クライアントとサーバー間のネットワーク・トラフィックを削減する、データ重複排除キャッシュを使用可能にします。このキャッシュには、以前の増分バックアップ操作でサーバーに送信されたエクステントが含まれています。クライアントは、サーバーに照会してエクステントがあるかどうかを確認する代わりに、クライアント自体のキャッシュに照会します。

クライアント・キャッシュのサイズと場所を指定します。サーバーとローカル・キャッシュ間に不整合が検出される場合、ローカル・キャッシュは除去され、再取り込みされます。

注：IBM Spectrum Protect API を使用するアプリケーションの場合は、データ重複排除キャッシュを使用してはなりません。キャッシュが IBM Spectrum Protect サーバーと同期していないことが原因でバックアップ障害が起きる可能性があるためです。複数の同時バックアップ/アーカイブ・クライアント・セッションが構成されている場合、セッションごとに別々のキャッシュが構成されていなければなりません。

- ・サーバーによって保管されるデータの容量を削減するために、クライアント・サイドのデータ重複排除と圧縮の両方を使用可能にします。各エクステントは、サーバーに送信される前に圧縮されます。ストレージの節約と、クライアント・データの圧縮に必要な処理能力との間には、トレードオフがあります。一般に、クライアント・システムでデータの圧縮と重複排除を行う場合、データ重複排除のみを行う場合の約2倍の処理能力を使用します。

サーバーは、重複排除と圧縮が行われたデータを処理できます。さらに、V6.2 より前のバックアップ・アーカイブ・クライアントは、重複排除と圧縮が行われたデータをリストアできます。

クライアント・サイド・データ重複排除では、次のプロセスを使用します。

- ・クライアントがエクステントを作成します。エクステントは、重複を識別するために他のファイル・エクステントと比較されるファイルのパーツです。
- ・クライアントとサーバーが連携して重複エクステントを識別します。クライアントが、非重複エクステントをサーバーに送信します。
- ・それ以降のクライアント・データ重複排除操作で新しいエクステントが作成されます。それらのエクステントの一部または全部が、以前のデータ重複排除操作で作成され、サーバーに送信されたエクステントと一致する可能性があります。一致するエクステントは、サーバーに再送信されません。

利点

クライアント・サイドのデータ重複排除には、以下のようないくつかの利点があります。

- ローカル・エリア・ネットワーク (LAN) 経由で送信されるデータの容量を削減できます。
- 重複データの識別に必要な処理能力が、サーバーからクライアント・ノードにオフロードされます。重複排除対応のストレージ・プールでは、サーバー・サイドのデータ重複排除は常に使用可能です。しかし、重複排除対応のストレージ・プール内にあるファイルで、クライアントによって重複排除されたものには、追加の処理は必要ありません。
- サーバー上の重複データの除去に要求される処理能力が不要になります。そのため、サーバー上のスペースが即時に節約できます。

クライアント・サイドのデータ重複排除には、潜在的な欠点があります。クライアント・エクステントを含む 1 次ストレージ・プールを、重複排除されていないコピー・ストレージ・プールにバックアップするまで、サーバーにはクライアント・ファイル全体のコピーはありません。(エクステントは、データ重複排除プロセス中に作成されるファイルの部分です。) 非重複排除ストレージ・プールへのストレージ・プール・バックアップ中に、クライアント・エクステントは再組み立てされて連続したファイルになります。

デフォルトでは、データ重複排除用にセットアップされた 1 次順次アクセス・ストレージ・プールは、レクラメーション処理の前、および重複データの削除前に、重複排除されていないコピー・ストレージ・プールにバックアップしておく必要があります。このデフォルトの処理によって、サーバーでは常に 1 次ストレージ・プールまたはコピー・ストレージ・プールのどちらかにファイル全体のコピーが存在するようになります。

重要: データ削減をさらに行う場合、クライアント・サイド・データ重複排除と圧縮を一緒に使用可能にすることができます。各エクステントは、サーバーに送信される前に圧縮されます。圧縮はスペースを節約しますが、クライアント・ワークステーション上の処理時間は増えます。

以下のオプションは、データ重複排除に関係するものです。

- 重複排除
- Dedupcachepath
- Dedupcachesize
- Enablededupcache
- Exclude.dedup
- Include.dedup

API クライアント・サイド・データ重複排除

クライアント・サイドのデータ重複排除は、データが IBM Spectrum Protect サーバーに転送される前に、バックアップおよびアーカイブ処理中に冗長データを除去するためにバックアップ/アーカイブ・クライアント上の API によって使用されます。

クライアント・サイドのデータ重複排除は、API に使用され、冗長データについて、IBM Spectrum Protect サーバーに転送される前、バックアップおよびアーカイブ処理中に除去します。クライアント・サイド・データ重複排除を使用して、ローカル・エリア・ネットワーク上で送信されるデータの容量を削減できます。クライアント・サイドのデータ重複排除を使用すると、IBM Spectrum Protect サーバーのストレージ・スペースを削減することもできます。

クライアントがクライアント・サイド・データ重複排除に対応していて、バックアップまたはアーカイブ操作を実行する場合は、データは複数のエクステントとしてサーバーに送信されます。バックアップまたはアーカイブ操作が次回に実行されるときに、クライアントとサーバーは、すでにバックアップまたはアーカイブされたデータ・エクステントを識別し、ほかに存在しないデータのエクステントのみをサーバーに送信します。

クライアント・サイド・データ重複排除の場合、サーバーおよび API はバージョン 6.2 以降でなければなりません。

クライアント・サイド・データ重複排除を使用してファイルをバックアップまたはアーカイブする前に、システムは以下の要件を満たす必要があります。

- クライアントの deduplication オプションが有効になっていなければならない。
- サーバーは、**REGISTER NODE** または **UPDATE NODE** コマンドの **DEDUP=CLIENTORSERVER** パラメーターを使用して、クライアントに対してクライアント・サイド・データ重複排除を使用可能にする必要がある。
- データのストレージ・プール宛先が、データ重複排除が使用可能になったストレージ・プールになっている。データ重複排除対応のストレージ・プールはファイル装置タイプのみです。
- ファイルが正しい管理クラスにバインドされていることを確認する。
- ファイルをクライアント・サイド・データ重複排除処理から除外できる。デフォルトでは、すべてのファイルが処理対象に含まれます。
- ファイルは 2 KB より大きくなければなりません。
- サーバー上の CLIENTDEDUPTXNLIMIT オプションを設定することによって、データ重複排除の最大トランザクション・サイズをサーバーが制限できる。このオプションについては、サーバー 資料の情報を参照してください。

上記のいずれかの要件が満たされていない場合は、データは通常に処理され、クライアント・サイド・データ重複排除は行われません。

データ重複排除に関する制限を以下に示します。

- LAN フリー・データ移動およびクライアント・サイド・データ重複排除は同時には使用できません。LAN フリー・データ移動とクライアント・サイド・データ重複排除の両方を使用可能にした場合は、LAN フリー・データ移動操作が実行され、クライアント・サイド・データ重複排除は無視されます。
- 暗号化およびクライアント・サイド・データ重複排除は同時には使用できません。暗号化とクライアント・サイド・データ重複排除の両方を使用可能にした場合は、暗号化操作が実行され、クライアント・サイド・データ重複排除は無視されます。暗号化ファイル、およびクライアント・サイド・データ重複排除の対象となるファイルは同じ操作で処理できますが、個別のトランザクションで行われます。

要件：

1. いずれのトランザクションでも、すべてのファイルがデータ重複排除の対象として包含されるかまたは除外されるかのいずれかである必要があります。トランザクションに混合ファイルが含まれている場合は、トランザクションは失敗し、戻りコード DSM_RC_NEEDTO_ENDTXN が API によって戻されます。
 2. ストレージ装置の暗号化を、クライアント・サイド・データ重複排除と一緒に使用します。SSL はクライアント・サイドの重複排除と組み合わせて使用されるため、クライアントの暗号化は不要です。
- 以下の機能は、クライアント・サイド・データ重複排除には使用できません。
 - IBM Spectrum Protect for Space Management (HSM) クライアント
 - API 共有バッファ
 - NAS
 - サブファイルのバックアップ
 - バッファ・コピー除去は、圧縮、暗号化、およびデータ重複排除などのデータ形式変更と併用することはできません。
 - クライアント・サイド非重複化を使用する場合、API は、バックアップを検出し、このバックアップ (RC=254 付き) を不合格とします。これは、データをサーバーに送信中に、サーバー上で有効期限が切れたというマークを付けられたファイル・エクステンツのバックアップです。この操作を再試行したい場合、このプログラミングを呼び出しアプリケーションに組み込む必要があります。
 - サーバー上の同時書き込み操作は、クライアント・サイド・データ重複排除に優先します。同時書き込み操作が使用可能にされている場合は、クライアント・サイド・データ重複排除は行われません。

制約事項：クライアント・サイド・データ重複排除が使用可能にされていると、定義済みの次のプールがある場合でも、API は、サーバーが宛先プール上のストレージを使い尽くしている状態から回復することはできません。停止理由コード DSM_RS_ABORT_DESTINATION_POOL_CHANGED が戻され、操作は失敗します。この状態から回復するには、次の 2 つの方法があります。

1. 追加のスクラッチ・ボリュームを元のファイル・プールに追加するように管理者に依頼します。

2. データ重複排除を使用不可能状態にして操作を再試行します。

ネットワーク帯域幅の節約をより一層行うために、データ重複排除のためのローカル・キャッシュを使用可能にすることができます。ローカル・キャッシュにより、照会が IBM Spectrum Protect サーバーへ送られずに済みます。ENABLEDEDUPCACHE のデフォルト値は NO です。そのため、キャッシュはサーバーと同期化されます。キャッシュがサーバーと同期化されない場合は、アプリケーションがすべてのデータを再送します。ご使用のアプリケーションが、失敗したトランザクションを再試行でき、かつローカル・キャッシュを使用するようにしたい場合は、dsm.opt (Windows) または dsm.sys (UNIX) ファイルの ENABLEDEDUPCACHE オプションを YES に設定します。

リストアの終了時に、データのすべてが API を介してリストアされ、オブジェクトがクライアントによって重複排除された場合は、エンドツーエンド・ダイジェストが計算され、バックアップ時に計算された値と比較されます。これらの値が一致しない場合は、エラー DSM_RC_DIGEST_VALIDATION_ERROR が戻れます。アプリケーションがこのエラーを受け取った場合は、データが壊れています。このエラーはネットワーク上の一時的エラーの結果である可能性もあるため、リストアまたはリトリブを再試行してください。

次に、データ重複排除の情報を示す query session コマンドの例があります。

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

次に、データ重複排除の情報を示す query managementclass コマンドの例があります。

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

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


関連資料

Deduplication オプション

データ重複排除からのファイルの除外

データ重複排除からバックアップ・ファイルまたはアーカイブ・ファイルを除外することを選択できます。

データ重複排除処理からファイルを除外するには、以下のステップに従ってください

1. 除外するオブジェクトに対して `exclude.dedup` オプションを設定します。

例えば、UNIX システム用のすべての重複排除データを除外するには、次のように設定します。

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

2. Windows システム用のすべての重複排除データを除外するには、次のように設定します。

```
exclude.dedup *¥...¥*
```

重要: オブジェクトがデータ重複排除プールに送信されると、そのオブジェクトがクライアント・サイド・データ重複排除から除外される場合でも、データ重複排除はサーバー上で行われます。

データ重複排除用のファイルの組み込み

データ重複排除用にバックアップ・ファイルまたはアーカイブ・ファイルを組み込むことを選択できます。

組み込むファイルのリストを詳細化するには、`include.dedup` オプションを `exclude.dedup` オプションに組み合わせて使用することができます。

デフォルトでは、すべての適格オブジェクトがデータ重複排除用に組み込まれます。

以下に、UNIX および Linux の例を示します。

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

以下に、Windows の例を示します。

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

サーバー・サイドのデータ重複排除

サーバー・サイドのデータ重複排除は、サーバーによって行われるデータ重複排除です。

IBM Spectrum Protect の管理者は、**REGISTER NODE** または **UPDATE NODE** サーバー・コマンドの **DEDUP** パラメーターと併用するデータ重複排除ロケーション (クライアントまたはサーバー) を指定することができます。

データ重複排除対応のストレージ・プール (ファイル・プール) では、データ・エクステントの 1 つのインスタンスのみが保存されます。同じデータ・エクステントのその他のインスタンスは、保存されるインスタンスへのポインターに置き換えられます。

サーバー・サイドのデータ重複排除について詳しくは、IBM Spectrum Protect サーバーの資料を参照してください。

アプリケーション・フェイルオーバー

IBM Spectrum Protect サーバーが障害のために使用不可能になった場合、API を使用するアプリケーションは、データ復旧のため 2 次サーバーに自動的にフェイルオーバーすることができます。

通常の実動プロセス時にクライアントと API が接続している IBM Spectrum Protect サーバーは、1 次サーバーと呼ばれています。1 次サーバーがノード複製のためにセットアップされている場合、そのサーバー

は、ソース複製サーバーとも呼ばれます。ソース複製サーバー上のクライアント・ノード・データは、ターゲット複製サーバーに複製することができます。このサーバーは、2次サーバーとも呼ばれ、1次サーバーの障害が発生した時にクライアントが自動的にフェイルオーバーする先のサーバーになります。

クライアントと API は、自動クライアント・フェイルオーバー用に構成し、クライアント・ノード・データの複製を行うバージョン 7.1 以降のサーバーに接続する必要があります。API の構成は、バックアップ/アーカイブ・クライアントの構成と同じです。

通常の操作中、ログオンを処理している時に 2 次サーバーに関する接続情報が 1 次サーバーからクライアントに自動的に送信されます。2 次サーバー情報は、クライアント・オプション・ファイルに自動的に保存されます。

クライアント・アプリケーションは、IBM Spectrum Protect サーバーにログオンするたびに、1 次サーバーに接続しようと試みます。1 次サーバーが使用不可能な場合、そのアプリケーションは、クライアント・オプション・ファイル内の 2 次サーバー情報を使用して、2 次サーバーに自動的にフェイルオーバーします。フェイルオーバー・モードでは、アプリケーションは 2 次サーバーを照会して、複製されたデータをリストアまたはリトリブすることが可能です。

少なくとも 1 回はアプリケーションを 1 次サーバーにバックアップする必要があります。API が 2 次サーバーにフェイルオーバーしてデータをリカバリーできるのは、クライアント・ノードのデータが 1 次サーバーから 2 次サーバーに複製されている場合のみです。

関連概念

[自動クライアント・フェイルオーバーの構成と用途](#)

フェイルオーバー状況情報

API が提供する状況情報をアプリケーションが使用すると、2 次サーバーに関するフェイルオーバー状況と複製されたクライアント・データの状況を判別することができます。

複製状況は、最新のバックアップが 2 次サーバーに複製されているかどうかを示します。API の最新のバックアップ操作のタイム・スタンプが 2 次サーバーのバックアップのタイム・スタンプと一致している場合、複製状況は「最新」になります。2 つのタイム・スタンプが一致していない場合、複製状況は最新ではなく、複製データが古いものである可能性があります。

以下の複製状況情報は、**qryRespFSDData** 構造の **dsmGetNextQObj** 関数呼び出しの **query filespace** 応答で戻されます。

表 14. API に報告される複製状況情報		
状況情報	タイプ	定義
最後の複製の開始	lastReplStartDate	複製が開始された最終時刻。
最後の複製の終了	lastReplCmpltDate	複製が完了した最終時刻 (失敗した場合も含む)。
最終バックアップの保管日付 (サーバー)	lastBackOpDateFromServer	サーバーに保管された最後の保管タイム・スタンプ。
最終バックアップの保管日付 (ローカル)	lastBackOpDateFromLocal	クライアントに保管された最後の保管タイム・スタンプ。

フェイルオーバー状況は、**dsmInitExOut_t** 構造の **bIsFailOverMode** フィールドで報告されます。

API の構造とタイプ定義については、155 ページの『付録 B API タイプ定義ソース・ファイル』を参照してください。

DSM_RC_SIGNON_FAILOVER_MODE 戻りコードは、クライアントと API が 2 次サーバーにフェイルオーバーされたこと、およびフェイルオーバー・モードで稼働中であることを示します。

フェイルオーバー中のサインオンの例

以下の出力例は、フェイルオーバー中のサーバーのサインオンの例を示します。


```

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

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

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

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

```

query session コマンドの例

以下の出力例は、2 次 (複製) サーバー情報を表示する **query session** コマンドでの例を示します。

```

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

```

query filespace コマンドの例

以下の出力例は、2 次サーバーのファイル・スペースの複製状況を表示する **query filespace** コマンドの出力例を示します。

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

Filespace Name	Type	Occupancy	Capacity	Start	End
/fs	API:Sample	100	300	0/0/0 0:0:0	0/0/0 0:0:0
Start of last Replication : 2013/5/21 21:3:2					
End of last Replication : 2013/5/21 21:3:3					
Server Local					
Last backup store date : 2013/5/21 21:18:25 2013/5/21 21:18:25					
Last archive store date : 0/0/0 0:0:0 0/0/0 0:0:0					
Last HSM store date : 0/0/0 0:0:0 0/0/0 0:0:0					
FSINFO : Sample API FS Info					

関連資料

106 ページの『[dsmGetNextQObj](#)』

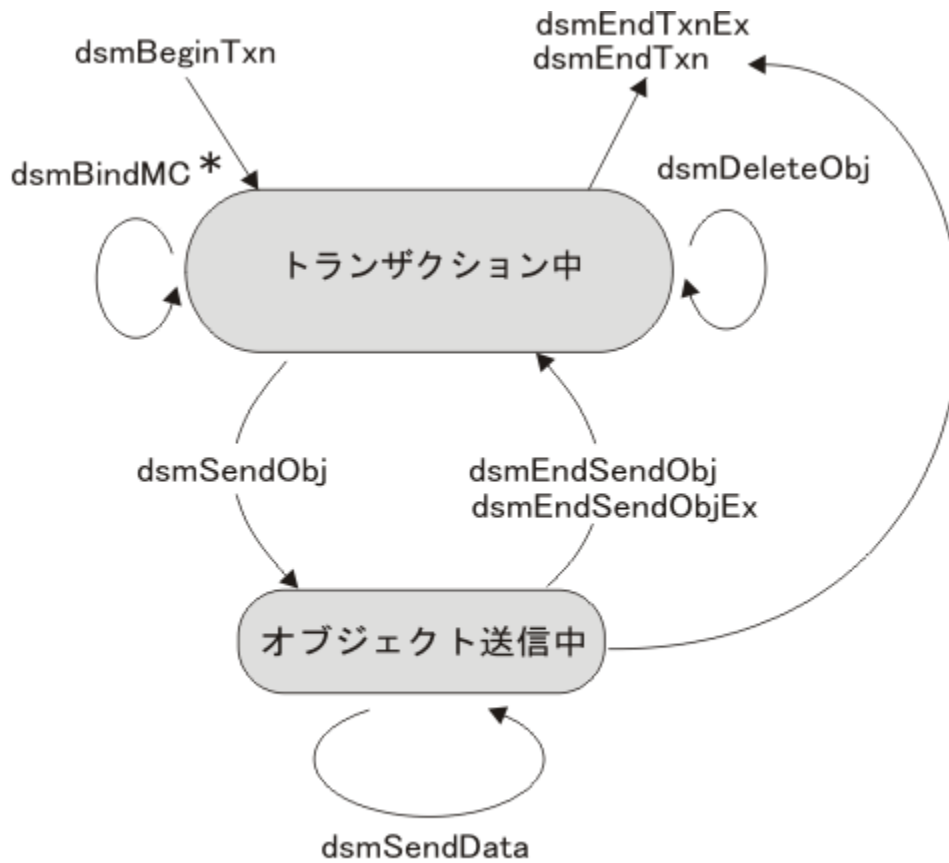
dsmGetNextQObj 関数呼び出しは、直前の **dsmBeginQuery** 呼び出しから次の照会応答を取得し、その応答を呼び出し元のバッファに入れます。

バックアップおよびアーカイブのフロー・ダイアグラムの例

API は、論理の流れが単純になるように、アプリケーション・クライアントの各種の状態の遷移が明瞭になるように設計されています。この明瞭な状態遷移によって、論理の欠陥やプログラム・エラーが開発サイクルの初期に突きとめられることから、システムの品質および信頼性は大幅に向上します。

例えば、トランザクションが既に開始済みで、バックアップされるオブジェクトについて **dsmBindMC** 呼び出しが前に行われていない限り、**dsmSendObj** 呼び出しを行うことはできません。

57 ページの図 12 は、トランザクション内でバックアップまたはアーカイブ操作を実行する場合の状態遷移を示しています。「オブジェクト送信中」から **dsmEndTxn** に向かう矢印は、**dsmSendObj** または **dsmSendData** の呼び出しの後で **dsmEndTxn** 呼び出しを開始できることを示します。これは、オブジェクトの送信中にエラー状態が起きて、操作全体を停止したい場合に行えます。その場合、DSM_VOTE_ABORT の vote を使用する必要があります。しかし通常は、トランザクションの終了前に、**dsmEndSendObj** を呼び出します。



* トランザクションの内側または外側である場合があります。

図 12. バックアップおよびアーカイブ操作の状態遷移

58 ページの図 13 は、トランザクション内でバックアップまたは アーカイブ 操作を実行する場合のフローチャートを示しています。

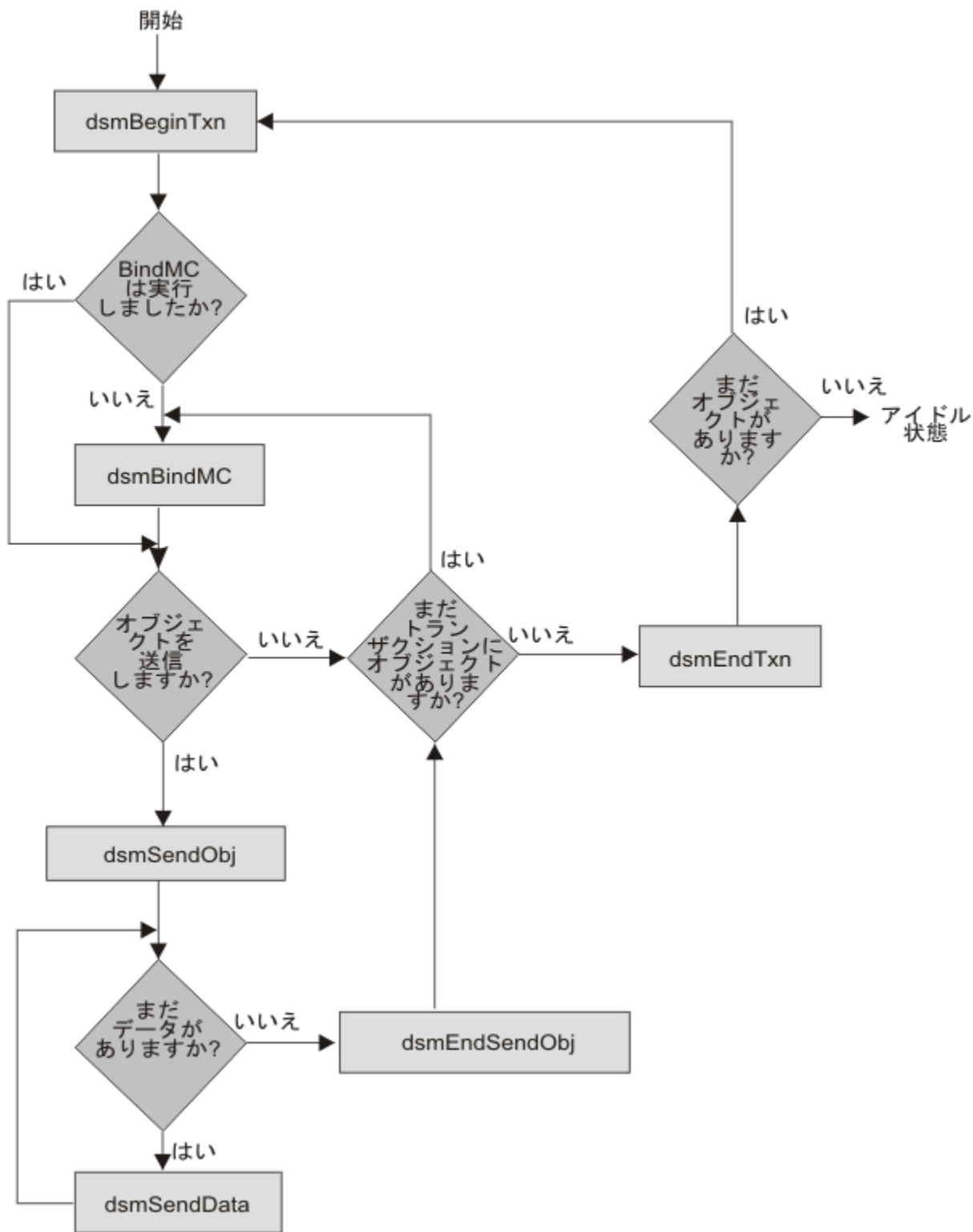


図 13. バックアップおよびアーカイブ操作のフローチャート

この 2 つの図における重要な機能は、トランザクション内からの次の API 呼び出し間のループです。

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

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

dsmBindMC 呼び出しは、トランザクション境界の内側または外側から出すことができるという点に特徴があります。この呼び出しは、必要に応じて、別のトランザクションから開始することもできます。

dsmBindMC 呼び出しの唯一の要件は、呼び出しをオブジェクトのバックアップまたはアーカイブに先立って行うということです。バックアップまたはアーカイブするオブジェクトが管理クラスに関連付けられていない場合は、**dsmSendObj** からエラー・コードが戻されます。この場合、トランザクションは

dsmEndTxn を呼び出すことによって終了します (このエラー条件は、フローチャートには示されていません)。

このフローチャートは、アプリケーションが複数オブジェクト・トランザクションをどのように使用するかを図解したものです。このフローチャートは、送信されるオブジェクトがそのトランザクション内に収まるか、あるいは新規トランザクションを開始する必要があるかを判別する際の判断のポイントをどこに置くかを示します。

IBM Spectrum Protect ストレージにデータを送信する API 機能のコーディング例

この例は IBM Spectrum Protect ストレージにデータを送信する API 機能の使用法を具体的に示しています。バックアップ操作またはアーカイブ操作のどちらが実行されているかに応じて、別のパラメーターを呼び出すことができるように、switch ステートメント内に **dsmSendObj** 呼び出しが示されています。

dsmSendData 呼び出しは、プログラム実行のループを終了するフラグが設定されるまでデータの送信を繰り返すループの内側から呼び出されます。送信操作全体がトランザクション内から行われることに注意してください。

dsmSendObj 呼び出しの 3 番目のパラメーターは、アーカイブ記述を含むバッファです。バックアップ・オブジェクトには記述がないので、オブジェクトをバックアップする場合は、このパラメーターは NULL です。

30 ページの図 8 は **dsmBindMC** 関数呼び出しの使用を示す例を示しています。

```

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

/* Call dsmBindMC
if not done previously */
objAttr.sizeEstimate.hi = 0; /* estimate of */
objAttr.sizeEstimate.lo = 32000; /* object size */
switch (send_type)
{
    case (Backup_Send) :
        rc = dsmSendObj
(dsmHandle,stBackup,
    NULL,&objName,&objAttr,NULL);
        break;
    case (Archive_Send) :
        archData.stVersion = sndArchiveDataVersion;
        archData.descr = desc;
        rc = dsmSendObj
(dsmHandle,stArchive,
    &archData,&objName,&objAttr,NULL);
        break;
    default : ;
}
if (rc)
{
    printf("*** dsmSendObj
failed: ");
    rcApiOut(dsmHandle, rc);
    return;
}
done = bFalse;
while (!done)
{
    dataBlk.stVersion = DataBlkVersion;
    dataBlk.bufferLen = send_amt;
    dataBlk.numBytes = 0;
    dataBlk.bufferPtr = bkup_buff;
    rc = dsmSendData
(dsmHandle,&dataBlk);
    if (rc)
    {
        printf("*** dsmSendData
failed: ");
        rcApiOut(dsmHandle, rc);
        done = bTrue;
    }
    /* Adjust the dataBlk buffer for the next piece to send */
}
rc = dsmEndSendObj
(dsmHandle);
if (rc)
{
    printf("*** dsmEndSendObj
failed: ");
    rcApiOut(dsmHandle, rc);
}
txn_reason = 0;
rc = dsmEndTxn
(dsmHandle, /* API session handle */
    DSM_VOTE_COMMIT, /* Commit transaction */
    &txn_reason); /* Reason if txn aborted */
if (rc || txn_reason)
{
    printf("*** dsmEndTxn
failed: rc = ");
    rcApiOut(dsmHandle, rc);
    printf(" reason =
}

```

図 14. サーバーへのデータの送信の例

ファイルのグループ化

IBM Spectrum Protect API には、個々のオブジェクトを複数まとめて関連付ける、論理ファイル・グループ化プロトコルがあります。これらのグループは、サーバー上の論理グループとして参照し、管理できます。論理グループでは、すべてのグループ・メンバーとグループ・リーダーが、サーバー上の同一のノードとファイル・スペースに属していることが必要です。

各グループは、グループ・リーダーを 1 つ持ちます。グループ・リーダーが削除されると、そのグループは削除されます。グループの一部であるメンバーを削除することはできません。グループのすべてメンバーの期限切れは、グループ・リーダーによって決まります。例えば、あるメンバーが期限切れとマークを付けられても、グループ・リーダーが期限切れになるまでは、そのメンバーは期限切れになりません。ただし、メンバーが期限切れとマークされなくても、グループ・リーダーが期限切れになると、すべてのメンバーが期限切れになります。

ファイル・グループはバックアップ・データのみを含んでおり、アーカイブ・データを含むことはできません。アプリケーションで必要であれば、アーカイブ・オブジェクトで **アーカイブの説明 (Archive Description)** フィールドを使用して、1 つのタイプのグループ化を行えます。

操作をグループ化するのは、**dsmGroupHandler** 呼び出しです。**dsmGroupHandler** 関数の呼び出しは、トランザクション内から行う必要があります。ほとんどのグループ・エラー状態は、**dsmEndTxn** 呼び出しまたは **dsmEndTxnEx** 呼び出しで検出されます。

dsmEndTxnEx の out 構造には、新しいフィールド **groupLeaderObjId** が含まれています。このフィールドには、グループがそのトランザクションでオープンされた場合、グループ・リーダーのオブジェクト ID が入ります。グループは、複数のトランザクションにまたがって作成できます。グループは、クローズされるまで、サーバーでコミットも保管もされません。**dsmGroupHandler** は、5 つの異なる操作を受け入れるインターフェースです。この 5 つの操作は、次のとおりです。

- DSM_GROUP_ACTION_OPEN
- DSM_GROUP_ACTION_CLOSE
- DSM_GROUP_ACTION_ADD
- DSM_GROUP_ACTION_ASSIGNTO
- DSM_GROUP_ACTION_REMOVE

61 ページの表 15 に **dsmGroupHandler** 関数呼び出しのアクションを示します。

表 15. dsmGroupHandler 関数	
アクション	説明
OPEN	OPEN アクションは、グループを作成します。次に送信されるオブジェクトが、グループ・リーダーになります。グループ・リーダーには内容を含めることができません。最初のオブジェクトの後のオブジェクトは、すべて、このグループに加えられるメンバーです。グループを作成するためには、グループをオープンし、グループを識別させるために固有のストリングを渡します。この固有の ID があるために、同じ名前をもつ複数のグループがオープン 状態になることができます。グループがオープンされた後、送信される次のオブジェクトがオブジェクト・リーダーになります。送信されるその他のオブジェクトは、すべてグループ・メンバーです。

表 15. dsmGroupHandler 関数 (続き)

アクション	説明
CLOSE	<p>CLOSE アクションによって、オープン・グループがコミットされ、保管されます。グループをクローズするには、オープン操作で使ったオブジェクト名と固有ストリングを渡します。アプリケーションはオープンされているグループの有無を検査し、必要であれば、それらのグループをクローズまたは削除する必要があります。グループは、クローズされるまではコミットも保存もされません。CLOSE アクションは、以下の状態では失敗します。</p> <ul style="list-style-type: none"> クローズしようとするグループに、既存のオープン・グループと同じ名前が付いている。 クローズされている現行グループと、同じ名前のクローズされる新規グループとの間で、管理クラスの非互換性が存在する。この場合は、以下の手順を実行します。 <ol style="list-style-type: none"> 前にクローズしたグループを照会します。 クローズされている既存グループの管理クラスが、現在オープンしているグループに関連付けられている管理クラスと異なっている場合は、タイプ DSM_BACKUPD_MC を指定して dsmUpdateObject を発行します。このコマンドは、既存グループを新しい管理クラスに更新します。 CLOSE アクションを発行します。
ADD	ADD アクションによって、オブジェクトがグループに追加されます。ADD アクションの後に送信される オブジェクトは、すべて、そのグループに割り当てられます。
ASSIGNTO	<p>ASSIGNTO アクションによって、クライアントは、サーバー上に存在する オブジェクトを、宣言済みのピア・グループに割り当てることができます。このトランザクションは PEER グループ関連をセットアップします。ASSIGNTO アクションは、以下の点を除いて、ADD アクションによく似ています。</p> <ul style="list-style-type: none"> ADD アクションは、未完了トランザクション内のオブジェクトに適用されます。 ASSIGNTO アクションは、サーバー上にあるオブジェクトに適用されます。
REMOVE	REMOVE アクションによって、グループからメンバーまたはメンバー・リストが除去されます。グループ・リーダーをグループから除くことはできません。グループ・メンバーを削除するには、前もってそのメンバーを REMOVE (除去) することが必要です。

グループ・サポートには、以下の照会タイプを使用します。

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

qtBackupGroups は、クローズされているグループを照会し、**qtOpenGroups** はオープンしているグループを照会します。これらの新しいタイプのための照会バッファーには、**groupLeaderObjId** と **objType** のためのフィールドがあります。照会の方法は、これら 2 つのフィールドの値によって異なります。次の表に、これらの照会の相違の例を示します。

表 16. 照会の例

groupLeaderObjId. hi	groupLeaderObjId.l o	objType	結果
0	0	NULL	すべてのグループ・リーダーのリストを戻す。
grpLdrObjId.hi	grpLdrObjId.lo	0	指定のグループ・リーダー (grpLdrObjId) に割り当てられているすべてのグループ・メンバーのリストを返す。

表 16. 照会の例 (続き)

groupLeaderObjId. hi	groupLeaderObjId.l o	objType	結果
grpLdrObjId.hi	grpLdrObjId.lo	objType	指定されたグループ・リーダー (grpLdrObjId) に割り当てられていて、オブジェクト・タイプ (objType) に一致する各グループ・メンバーのリストを返す (BackQryRespEnhanced3 を使用)。

dsmGetNextQObj からの応答構造 (**qryRespBackupData**) には、グループ・サポートのための次の 2 つのフィールドがあります。

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

これらのフィールドは、ブール・フラグです。以下に、グループを作成し、そのグループにメンバーを追加し、グループをクローズして IBM Spectrum Protect サーバーにグループをコミットする例を示します。

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

図 15. グループを作成するために使用する疑似コードの例

コード例については、API `sampsrc` ディレクトリーに含まれているサンプル・グループ・プログラム `dsmgrp.c` を参照してください。

サーバーからのデータの受信

アプリケーション・クライアントは、リストアおよび検索機能を使用して、IBM Spectrum Protect ストレージからデータ、あるいは指定のオブジェクトとその関連データを受信できます。リストア機能はあらかじめバックアップがとられていたオブジェクトにアクセスすることであり、リトリブ機能はあらかじめアーカイブされていたオブジェクトにアクセスすることです。

制約事項: API 呼び出しによってバックアップまたはアーカイブされたオブジェクト については、API は、リストアまたはリトリブのみができます。

リストア機能およびリトリブ機能はどちらも照会操作によって開始します。照会は、初めにデータのバックアップがとられたか、またはアーカイブされたかに応じて、別々の情報を戻します。例えば、バックアップ・オブジェクトの照会はオブジェクトが活動、非活動のいずれであるかについての情報を戻すのに対して、アーカイブ・オブジェクトの照会はオブジェクト記述のような情報を戻します。どちらの照会も、サーバーのオブジェクトを固有に識別するために使用されるオブジェクト ID を戻します。

部分オブジェクト・リストア/リトリート

アプリケーション・クライアントはオブジェクトの一部分のみを受け取ることができます。これは、「部分オブジェクト・リストア」または「部分オブジェクト・リトリート」と呼ばれます。



重要: 圧縮されたオブジェクトや暗号化されたオブジェクトに対して部分リストア/リトリートを実行すると、予測不能な結果になります。

注: 部分オブジェクト・リストア/リトリートを使用するアプリケーションをコーディングする場合は、データの送信中にデータを圧縮できません。これを強制的に行うには、*ObjAttr.objCompressed* を *bTrue* に設定します。

部分オブジェクト・リストア/リトリートを実行するには、次の2つのデータ・フィールドをそれぞれのオブジェクトの **GetList** 項目に関連付けます。

offset

データの戻りを開始するオブジェクト内のバイト・オフセット

length

戻すオブジェクトのバイト数。

特定の **dsmBeginGetData** リストについて、部分オブジェクトのリストアまたはリトリートを実行できるオブジェクトの最大数を判別するには、**DSM_MAX_PARTIAL_GET_OBJ** を使用してください。

dsmBeginGetData 呼び出しで使用する以下のデータ・フィールドによって、オブジェクトのどの部分がリストアまたはリトリートされるかが決まります。

- offset と length の両方がゼロであれば、オブジェクト全体が IBM Spectrum Protect ストレージからリストアまたはリトリートされます。
- offset がゼロより大きく、length がゼロの場合は、オブジェクトは offset から終わりまでがリストアまたはリトリートされます。
- length がゼロより大きい場合は、offset から、length で指定した長さのオブジェクト部分のみが、リストアまたはリトリートされます。

データのリストア/リトリート

IBM Spectrum Protect サーバーに照会を行い、サーバーとのセッションが確立された後で、データのリストアまたはリトリートを行う手順を実行できます。

手順

データをリストアまたはリトリートするには、以下のステップを実行します。

1. IBM Spectrum Protect サーバーに、バックアップまたはアーカイブ・データのどちらかについて照会します。
2. サーバーからリストアまたはリトリートすべきオブジェクトを判別します。
3. **リストア順序** フィールドでオブジェクトをソートします。
4. アクセスしたいオブジェクトのリストを指定して **dsmBeginGetData** 呼び出しを送信します。
5. **dsmGetObj** 呼び出しを送信して、システムから各オブジェクトを取得します。すべての関連したオブジェクト・データを取得するために、オブジェクトごとに複数の **dsmGetData** 呼び出しが必要な場合があります。1つのオブジェクトについてすべてのデータを取得した後で、**dsmEndGetObj** 呼び出しを送信します。
6. すべてのオブジェクトについてすべてのデータを受信した後、または受信操作を終了するために、**dsmEndGetData** 呼び出しを送信します。

サーバーへの照会

リストア操作またはリトリブ操作を開始する前に、IBM Spectrum Protect サーバーに照会して、ストレージからのオブジェクトを受信できるかを判別する必要があります。

照会を送信するには、アプリケーションは **dsmBeginQuery** 呼び出しのパラメーター・リストおよび構造を入力する必要があります。構造には、照会で調べるファイル・スペースと、高位名および低位名のフィールドのパターン照合項目が含まれている必要があります。セッションが NULL 所有者名を指定して初期設定された場合は、所有者フィールドを指定する必要はありません。しかし、セッションが明示的に所有者名を指定して初期設定された場合は、その所有者名に関連付けられているオブジェクトだけが返されます。

時刻指定 **BackupQuery** 照会によって、特定の時刻におけるシステムのスナップショットが提供されます。有効な日付を指定することによって、その時刻までにバックアップがとられたすべてのファイルを照会できます。もっと新しい日付の活動バックアップがオブジェクトにある場合でも、特定時点がオブジェクトの状態を指定変更するので、以前の非活動のコピーが戻されます。詳しくは、[pitDate](#) の例を参照してください。

照会を行うと、次の表に示した情報に加えて、オブジェクトとともに保管されたすべての情報が返されます。

表 17. サーバーへの照会で返される情報

フィールド	説明
copyId	copyIdHi および copyIdLo の値は、IBM Spectrum Protect ストレージのこのノードに、当該オブジェクトを固有に識別する 8 バイトの数値を提供します。この ID を使用して、リストアまたはリトリブ 処理のためにストレージから特定のオブジェクトを要求してください。
restoreOrderExt	restoreOrderExt 値は、実現可能な最も効率のよい方法で IBM Spectrum Protect ストレージからオブジェクトを受信するためのメカニズムを提供します。リストアするオブジェクトをこの値でソートして、テープが確実に 1 回だけマウントされ、初めから終わりまで読み取られるようにします。

後で処理するために、照会情報の一部または全部を保存しておく必要があります。実際のリストア操作に必要なになるので、copyId および restoreOrderExt フィールドを保存しておきます。データ・ファイルをオープンするため、あるいは宛先を識別するために必要な他のすべての情報も保存する必要があります。

照会操作を終了するときは、**dsmEndQuery** を呼び出してください。

リストア順序によるオブジェクトの選択およびソート

バックアップまたはアーカイブ照会が実行されたら、アプリケーション・クライアントは、どのオブジェクト (ある場合) をリストアまたはリトリブすべきかを判別する必要があります。

さらに、オブジェクトを昇順で (低位から高位に) ソートします。このソートは、リストア操作のパフォーマンス上、非常に重要です。**restoreOrderExt** フィールドでオブジェクトをソートすることによって、サーバーから最も効率的な順序でデータが読み取られることが保証されます。

最初にディスク上のすべてのデータがリストアされ、次にボリュームのマウントを必要とするメディア・クラス (テープなど) 上のデータがリストアされます。**restoreOrderExt** フィールドを使うことによって、テープ上のデータの読み取りが、テープの頭で処理を開始し、テープの終わりに向かって処理を整然と進めることも保証されます。

restoreOrderExt フィールドでのソートを正しく行えば、テープ・マウントの重複やテープの不要な巻き戻しは起こりません。

restoreOrderExt.top フィールドに指定したゼロ以外の値が IBM Spectrum Protect サーバー上の固有のシリアル・アクセス・デバイスに関連付けられます。シリアル・アクセス・デバイスは、一度に 1 つのセッション/マウント・ポイントでしか使用できないため、アプリケーションで複数のセッションを使用する場合は、同じ **restoreOrderExt.top** 値での同時リストアが存在しないことをアプリケーションで確認

する必要があります。そうしないと、最初のセッションではオブジェクトにアクセスできますが、他のセッションは、最初のセッションが終了してデバイスが使用可能になるまで待機することになります。

次の例は、「**リストア順序**」フィールドを使用してオブジェクトをソートする方法を示しています。

図 16. リストア順序フィールドによるオブジェクトのソート

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

} SortOrder;          /* struct used for sorting */

=====
/* the code for sorting starts from here */
dsmQueryType      queryType;
qryBackupData      queryBuffer;
DataBlk           qDataBlkArea;
qryRespBackupData  qbDataArea;
dsInt16_t          rc;
dsBool_t          done = bFalse;
int i = 0;
int qry_item;
SortOrder sortorder[100]; /* sorting can be done up to 100 items
                           only right now. Set appropriate
                           array size to fit your needs */

/*-----+
| NOTE: Make sure that proper initializations have been done to
|   queryType,
|   queryBuffer, qDataBlkArea, and qbDataArea.
|-----*/

qDataBlkArea.bufferPtr = (char*) &qbDataArea;

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

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

        } /* if ((rc == DSM_RC_MORE_DATA) || (rc == DSM_RC_FINISHED)) */
        else
        {
            done = bTrue;
            /*-----+
            /* take appropriate action. */
            /*-----+
        }

        i++;
        qry_item++;

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

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

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

```

/*-----+
int SortRestoreOrder(SortOrder *a, SortOrder *b)
|
| This function compares restoreOrder fields from two structures.
| if (a > b)
|     return(GREATERTHAN);
| if (a < b)
|     return(LESSTHAN);
| if (a == b)
|     return(EQUAL);
|-----*/
int SortRestoreOrder(SortOrder *a, SortOrder *b)
{
    if (a->restoreOrderExt.top > b->restoreOrderExt.top)
        return(GREATERTHAN);
    else if (a->restoreOrderExt.top < b->restoreOrderExt.top)
        return(LESSTHAN);
    else if (a->restoreOrderExt.hi_hi > b->restoreOrderExt.hi_hi)
        return(GREATERTHAN);
    else if (a->restoreOrderExt.hi_hi < b->restoreOrderExt.hi_hi)
        return(LESSTHAN);
    else if (a->restoreOrderExt.hi_lo > b->restoreOrderExt.hi_lo)
        return(GREATERTHAN);
    else if (a->restoreOrderExt.hi_lo < b->restoreOrderExt.hi_lo)
        return(LESSTHAN);
    else if (a->restoreOrderExt.lo_hi > b->restoreOrderExt.lo_hi)
        return(GREATERTHAN);
    else if (a->restoreOrderExt.lo_hi < b->restoreOrderExt.lo_hi)
        return(LESSTHAN);
    else if (a->restoreOrderExt.lo_lo > b->restoreOrderExt.lo_lo)
        return(GREATERTHAN);
    else if (a->restoreOrderExt.lo_lo < b->restoreOrderExt.lo_lo)
        return(LESSTHAN);
    else
        return(EQUAL);
}

```

dsmBeginGetData 呼び出しの開始

受信するオブジェクトを選択してソートした後、オブジェクトをリストアまたはリトリーブ操作するために、IBM Spectrum Protect に実行依頼します。**dsmBeginGetData** 呼び出しはリストアまたはリトリーブ操作を開始します。オブジェクトは要求した順序でアプリケーション・クライアントに戻されます。

これらの呼び出しの次の 2 つのパラメーターに情報を指定します。

mountWait

このパラメーターは、アプリケーション・クライアントがオブジェクトのデータを取得するためにオフライン・メディアがマウントされるまで待機するのか、あるいは、リストアまたはリトリーブ操作の処理中に当該オブジェクトをスキップすべきかを、サーバーに伝えます。

dsmGetObjListP

このパラメーターは、リストアまたはリトリーブされるすべてのオブジェクト ID のリストである **objId** フィールドを含むデータ構造です。各 **objId** は、**partialObjData** 構造に関連付けられており、これが、**objId** 全体の検索であるか、またはオブジェクトの特定のセクションのみの検索であるかを示します。

各 **objId** の長さは 8 バイトなので、単一のリストアまたはリトリーブ要求に何千ものオブジェクトを持つことができます。単一の呼び出しで要求するオブジェクトの数は、DSM_MAX_GET_OBJ または DSM_MAX_PARTIAL_GET_OBJ に制限されます。

リストアまたはリトリーブするための各オブジェクトの受信

dsmBeginGetData 呼び出しを送信した後で、サーバーから送信される各オブジェクトを受信するための手順を実行することができます。

このタスクについて

DSM_RC_MORE_DATA 戻りコードは、バッファーが返されたので、再度 **dsmGetData** を呼び出す必要があることを意味します。**DataBlk.numBytes** を調べ、返された実際のバイト数を確認してください。

オブジェクトに関するすべてのデータを取得する場合、**dsmEndGetObj** 呼び出しを送信する必要があります。オブジェクトをさらに受信する場合には、**dsmGetObj** を再送信してください。

この処理を停止する場合 (例えば、まだ受信していないすべてのオブジェクトについて、リストア・ストリーム内に残っているデータをすべて廃棄するなどの場合) には、**dsmEndGetData** 呼び出しを送信してください。この呼び出しで、サーバーからクライアントへのデータがフラッシュされます。ただし、この方法は完了するまでに時間がかかります。リストア操作を終了するためには、**dsmTerminate** を使用してセッションをクローズしてください。

手順

1. データ・ストリームから要求されるオブジェクトを識別するため、またそのオブジェクトに関連したデータの最初のブロックを取得するために **dsmGetObj** 呼び出しを送信してください。
2. 必要に応じて、さらに **dsmGetData** 呼び出しを送信して残りのオブジェクト・データを取得します。

リストアおよびリトリブのフロー・ダイアグラムの例

リストア操作およびリトリブ操作の実行方法を示す例として、状態遷移図およびフローチャートを使用できます。

"オブジェクト取得中" のブロックから **dsmEndGetData** に向かう矢印は、**dsmGetObj** または **dsmGetData** の呼び出しの後で **dsmEndGetData** 呼び出しを開始できることを示します。これは、オブジェクトを IBM Spectrum Protect ストレージから取得しているときにエラー状態が起きて、操作を停止したい場合に必要です。しかし、通常は、最初に **dsmEndGetObj** を呼び出してください。

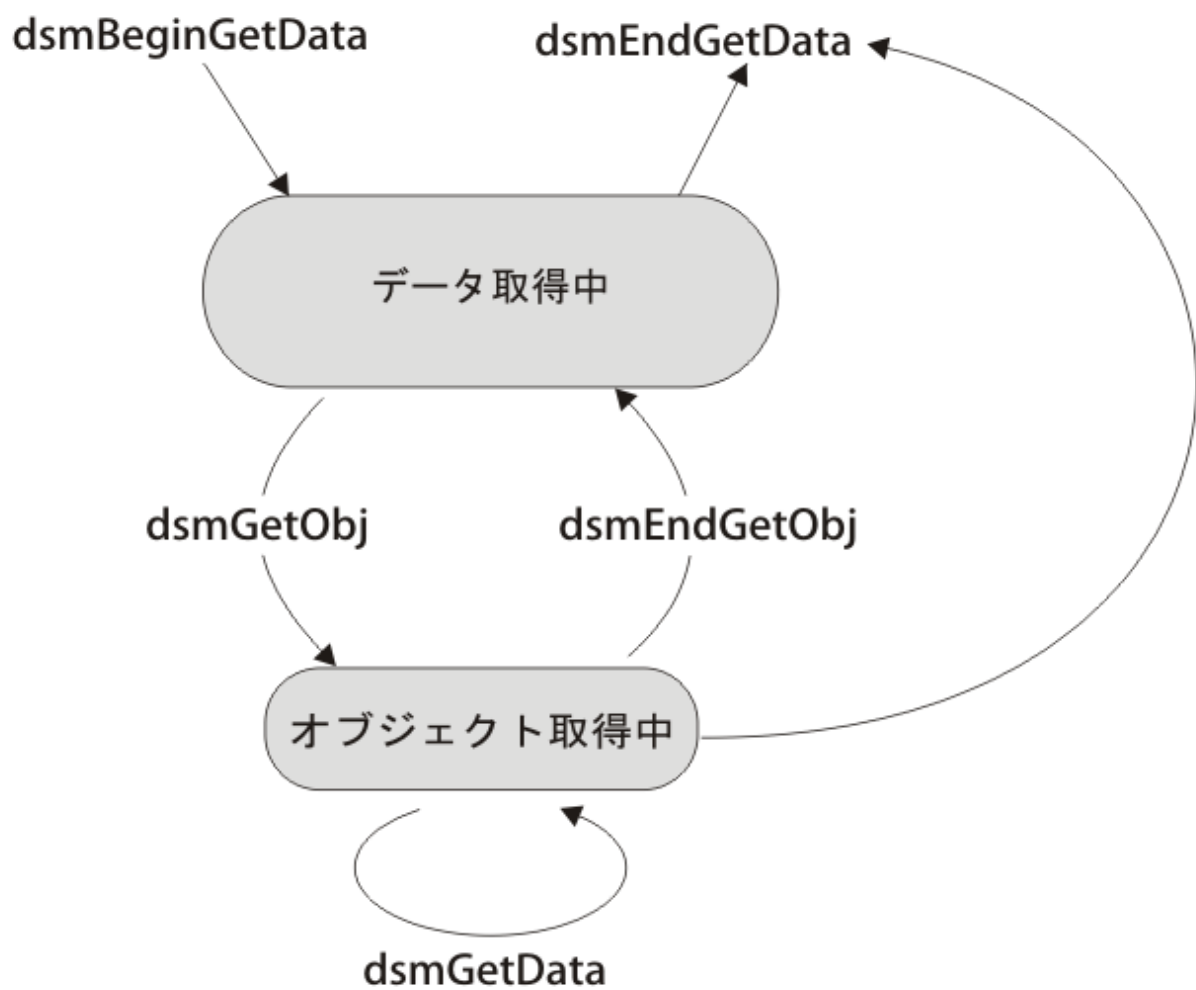


図 17. リストアおよびリトリブ操作の状態遷移

70 ページの図 18 に示すのは、リストア操作またはリトリブ操作を実行する場合のフローチャートです。

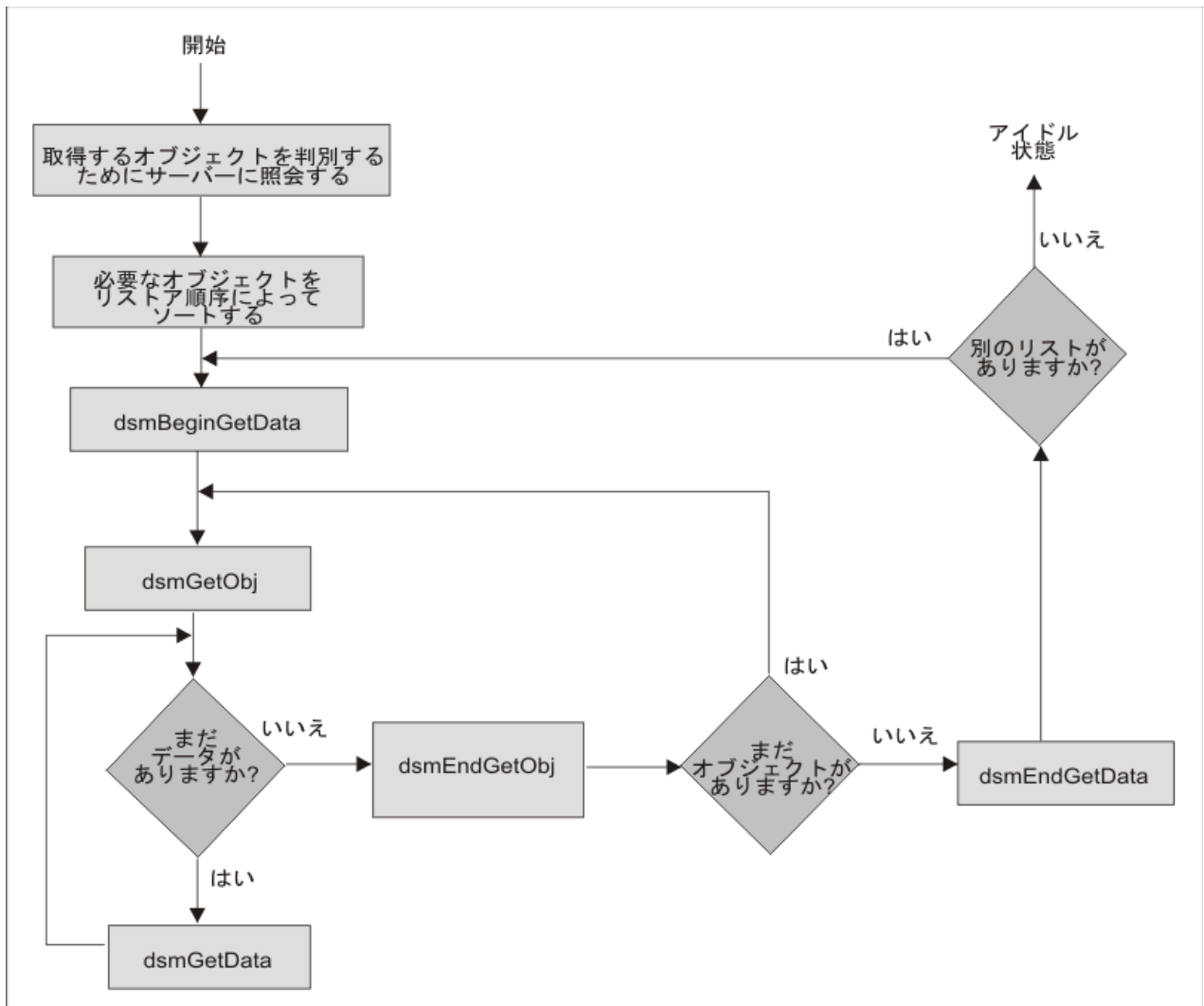


図 18. リストアおよびリトリブ操作のフローチャート

サーバーからデータを受信するコーディング例

この例は、IBM Spectrum Protect ストレージからデータをリトリブする API 機能の使用法を示しています。

リストア操作またはリトリブ操作のどちらが実行されているかに応じて、別のパラメーターを呼び出すことができるように、switch ステートメント内に **dsmBeginGetData** 呼び出しが示されています。

dsmGetData 関数呼び出しはプログラム実行のループを終了するフラグが設定されるまでデータの取得を繰り返すループの内側から呼び出されます。

図 19. サーバーからのデータの受信の例

```

/* Call dsmBeginQuery and create a linked list of objects to restore. */
/* Process this list to create the proper list for the GetData calls. */
/* Set up the getList structure to point to this list. */
/* This example is set up to perform a partial object retrieve. To */
/* retrieve only complete objects, set up: */
/*      getList.stVersion = dsmGetListVersion; */
/*      getList.partialObjData = NULL; */
dsmGetList getList;
getList.stVersion = dsmGetListPORVersion; /* structure version */
getList.numObjId = items; /* number of items in list */
getList.objId = (ObjID *)rest_ibuff;
/* list of object IDs to restore */
getList.partialObjData = (PartialObjData *) part_ibuff;
/* list of partial object data */
switch(get_type)

```

```

{
    case (Restore_Get) :
        rc = dsmBeginGetData
(dsmHandle,bFalse,gtBackup,&getList);
        break;
    case (Retrieve_Get) :
        rc = dsmBeginGetData
(dsmHandle,bFalse,gtArchive,&getList);
        break;
    default : ;
}
if (rc)
{
    printf("*** dsmBeginGetData
failed: ");
    rcApiOut(dsmHandle, rc);
    return rc;
}
/* Get each object from the list and verify whether it is on the */
/* server. If so, initialize structures with object attributes for */
/* data validation checks. When done, call dsmGetObj */
/* */
rc = dsmGetObj
(dsmHandle,objId,&dataBlk);
done = bFalse;
while(!done)
{
    if ( (rc == DSM_RC_MORE_DATA)
        || (rc == DSM_RC_FINISHED))
    {
        if (rc == DSM_RC_MORE_DATA)
        {
            dataBlk.numBytes = 0;
            rc = dsmGetData
(dsmHandle,&dataBlk);
        }
        else
            done = bTrue;
    }
    else
    {
        printf("*** dsmGetObj
or dsmGetData
failed: ");
        rcApiOut(dsmHandle, rc);
        done = bTrue;
    }
}
/* while */
rc = dsmEndGetObj
(dsmHandle);
/* check rc from dsmEndGetObj */
/* check rc from dsmEndGetData */
rc = dsmEndGetData (dsmHandle);
return 0;

```

サーバー上のオブジェクトの更新および削除

API アプリケーションは、**dsmUpdateObj** または **dsmUpdateObjEx** 関数呼び出しを使用して、アーカイブ済みまたはバックアップ済みオブジェクトを更新することができます。いずれかの呼び出しは、このセッション状態でのみ使用し、一度に1つのオブジェクトを更新してください。同じ名前を含む複数のアーカイブ・オブジェクトのいずれかを更新するには、**dsmUpdateObjEx** を使用します。

アーカイブ・オブジェクトを選択するには、**stArchive** に対する **dsmSendType** 関数呼び出しを設定してください。

- **dsmUpdateObj** では、割り当て名を持つ最新のアーカイブ・オブジェクトのみが更新されます。
- **dsmUpdateObjEx** では、適切なオブジェクト ID を指定して任意のアーカイブ・オブジェクトを更新することができます。

アーカイブ・オブジェクトの場合、アプリケーションは次のフィールドを更新することができます。

- 説明
- オブジェクト情報

- 所有者

バックアップ・オブジェクトを選択するには、**stBackup** に対して **dsmSendType** を設定します。バックアップ・オブジェクトの場合、アクティブ・コピーのみが更新されます。

バックアップ・オブジェクトの場合、アプリケーションは次のフィールドを更新することができます。

- 管理クラス (Management class)
- オブジェクト情報
- 所有者

サーバーからのオブジェクトの削除

API アプリケーションは、アーカイブされたオブジェクトを削除するか、またはバックアップがとられたオブジェクトをオフにする呼び出しを行うことができます。アーカイブ・オブジェクトの削除は、管理者によりノードの登録時に与えられたノード権限に依存します。管理者は、ノードがアーカイブ・オブジェクトを削除できることを指定することができます。

dsmDeleteObj 関数呼び出しを使用して、アーカイブ済みオブジェクトを削除し、バックアップ・オブジェクトをオフにします。この **delType** を使用すると、バックアップ・オブジェクトがサーバーから除去されます。これは、**objID** に基づいて、サーバー・データベースからオブジェクトを削除します。オブジェクトを削除できるのは、そのオブジェクトの所有者のみです。オブジェクトの任意のバージョン (活動または非活動) を削除できます。サーバーがバージョンの調整を行います。オブジェクトの活動バージョンを削除した場合、最初の非活動バージョンが活動状態になります。オブジェクトの非アクティブ・バージョンを削除すると、それより古いすべてのバージョンの順位が前に進みます。ノードは、**backDel** 許可を指定して登録されている必要があります。

アーカイブ・オブジェクトは、システムが次のオブジェクト期限切れサイクルを実行する時に、ストレージ内で削除のマークが付けられます。アーカイブ・オブジェクトはサーバーから削除された後は、リトリートできません。

サーバーでバックアップ・オブジェクトを非活動にすると、オブジェクトの状態は活動から非活動に移ります。これらの状態には、割り当てられた管理クラスに基づいて、それぞれに別の保持ポリシーが関連付けられています。

dsmSendObj 呼び出しと同様に、**dsmDeleteObj** の呼び出しは、トランザクションの範囲内で送信されます。57 ページの図 12 の状態遷移は、**dsmDeleteObj** 呼び出しが、その前に **dsmBeginTxn** の呼び出しを持ち、その後に **dsmEndTxn** の呼び出しが続くことを示しています。

イベント・ロギング

API アプリケーションは、イベント・メッセージを中央設置場所に記録できます。アプリケーションは、ログを IBM Spectrum Protect サーバー、ローカル・マシン、またはその両方に送信できます。

dsmLogEventEx 関数呼び出しは、セッション内で実行されます。サーバー上に記録されたメッセージを表示するには、管理可能クライアントで **query actlog** コマンドを使用します。

アプリケーションが、非常に多数のクライアント・メッセージをクライアント・ログ **dsmLogType** (**logLocal** または **logBoth** のいずれか) に書き込む場合は、IBM Spectrum Protect クライアント・オプション **errorlogretention** を使用して、クライアント・エラー・ログ・ファイルを除去してください。

IBM Spectrum Protect ログについて詳しくは、IBM Spectrum Protect サーバーの資料を参照してください。

IBM Spectrum Protect API の状態遷移図の要約

IBM Spectrum Protect API を使用して所有アプリケーションを作成する際のすべての考慮事項を検討した後、アプリケーション全体についての以下の状態遷移の要約を検討してください。

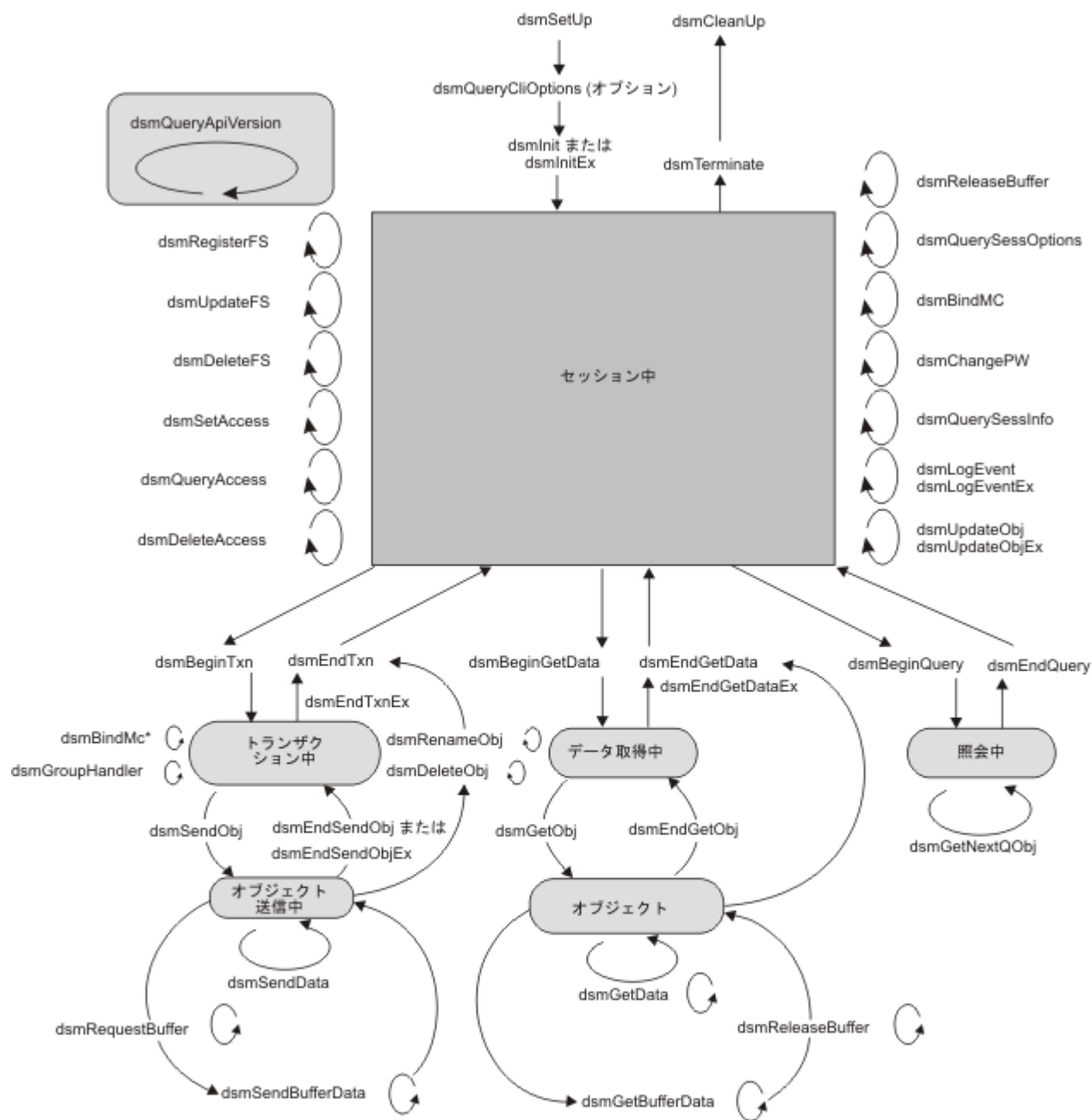
74 ページの図 20 は API の状態遷移を示しています。この図には、前出のすべての状態遷移と、まだ示していない他のいくつかの呼び出しが含まれています。

この図の重要なポイントは、次のとおりです。

- **dsmQueryApiVersionEx** は、いつでも呼び出せます。関連する状態はありません。例については、[16 ページの図 1](#) を参照してください。
- **dsmQueryCliOptions** は、**dsmInitEx** 呼び出しの前でのみ呼び出します。
- ファイル・スペースを管理するには、**dsmRegisterFS**、**dsmUpdateFS**、および **dsmDeleteFS** を使用します。これらの呼び出しは、アイドル・セッション状態の中で行われます。ファイル・スペースについて照会するには、**dsmBeginQuery** 呼び出しを使用します。ファイル・スペース呼び出しの詳細は、[25 ページの『ファイル・スペースの管理』](#)を参照してください。
- **dsmBindMC** 呼び出しは、アイドル・セッション状態の中から、あるいはオブジェクト送信トランザクション状態の中から送信してください。[30 ページの図 8](#) の例を参照してください。
- **dsmChangePW** 呼び出しは、アイドル・セッション状態の中から送信してください。

注：**dsmInitEx** 呼び出しがパスワード期限切れ戻りコードを出して戻る場合、有効なセッションを開始するには、その前に **dsmChangePW** 呼び出しを行う必要があります。**dsmChangePW** を使用する例については、[21 ページの図 4](#) を参照してください。

- 呼び出しがエラーで戻る場合には、状態は以前のままです。例えば、**dsmGetObj** がエラーで戻る場合には、状態はデータ取得中のままとなり、**dsmEndGetObj** への呼び出しは、呼び出しシーケンス・エラーとなります。



* トランザクションの内側または外側場合があります。

図 20. API の状態遷移の要約

第4章 インターオペラビリティについて

API には、2つのタイプのインターオペラビリティがあります。つまり、バックアップ/アーカイブ・クライアントと API アプリケーションとの間のインターオペラビリティと、異なるオペレーティング・システム間のインターオペラビリティです。

バックアップ/アーカイブ・クライアントのインターオペラビリティ

バックアップ/アーカイブのコマンド・ラインによって、限定されたインターオペラビリティを提供する API オブジェクトにアクセスできます。API オブジェクトを表示してアクセスできるのは、バックアップ・アーカイブ・コマンド・ライン・クライアントからのみであり、どのグラフィカル・インターフェースからも、表示やアクセスができません。バックアップ・アーカイブ・コマンド・ライン・クライアントは、ファイルの内容をリストアップできるだけで、他の操作はできません。したがって、修復タイプの操作にのみ、これを使用してください。

次のような コマンド・ライン・アクションが提供されています。

- Delete archive
- Delete file space
- Query
- Restore
- Retrieve
- Set access

パス情報は、バックアップ/アーカイブ・クライアント・オブジェクトのための 実ディレクトリーです。これに対し、API オブジェクト・パス情報は既存のディレクトリーとは関係がない場合もあります。すなわち、このパスは完全に作り出されることがあります。インターオペラビリティは、これらのオブジェクト・タイプのこの局面を変更することはありません。この機能を正常に使用するには、その制限と規則に従ってください。

注：

1. バックアップ/アーカイブ・クライアントと、保存保護サーバーに保管されている API オブジェクトの間には、インターオペラビリティはありません。
2. API クライアントを使用して保管されたファイルを、バックアップ/アーカイブ・クライアント GUI を使用してアクセスすることはできません。これらのファイルをアクセスするには、コマンド・ラインを使用するしかありません。

API オブジェクトの命名

API オブジェクト名に一貫性のある命名規則を設定してください。命名規則は、ファイル・スペース、高位修飾子、および低位修飾子に対応できるものであることが必要です。ファイル・スペース名と高位修飾子は、実ディレクトリー名を参照できます。各オブジェクト名は、複数の、低位修飾子に適用されるディレクトリー名から構成することができます。

便宜上、ディレクトリー情報が接頭部として付いていないオブジェクト名を低位修飾子として使用してください。詳しくは、[23 ページの『オブジェクト名とオブジェクト ID』](#)を参照してください。

ファイル・スペース名は、API またはバックアップ/アーカイブ・コマンド・ラインのどちらから参照される場合も、完全修飾名であることが必要です。例えば、UNIX または Linux オペレーティング・システムで、以下のファイル・スペースを登録したとします。

- /a
- /a/b

/a を参照した場合は、ファイル・スペース /a だけに関連したオブジェクトが表示されます。/a/b に関連したオブジェクトを表示するには、/a/b をファイル・スペース名として指定する必要があります。

両方のファイル・スペースを登録した後で、ファイル・スペース /a にオブジェクト b をバックアップしてから、/a/b に対する照会を行うと、ファイル・スペース /a/b のみに関連するオブジェクトが引き続き表示されます。

上記の制限に対する例外は、API を使用してファイル・スペースの照会または削除を試みている場合に、ファイル・スペースの参照で起こります。いずれの場合も、ワイルドカード文字を使用する場合には、ファイル・スペース名を完全に修飾する必要はありません。例えば、/a* は /a と /a/b の両方を参照します。

ヒント：インターオペラビリティが重要である場合は、オーバーラップするファイル・スペース名を使用しないでください。

Windows システムでは、バックアップ/アーカイブのコマンド・ライン・インターフェースからオブジェクトにアクセスするときは、API オブジェクトのファイル・スペース名を中括弧 { } で囲みます。Windows オペレーティング・システムでは、ファイル・スペース名は、登録または参照されるときに自動的に英大文字にされます。ただし、この自動機能は、その他のオブジェクト名指定では実行されません。完全なインターオペラビリティが必要な場合は、API オブジェクトをバックアップするときに、アプリケーションにおける高位修飾子および低位修飾子を英大文字にしてください。アプリケーションからサーバーにオブジェクトを送信する前に、そのオブジェクトの高位修飾子(ディレクトリー名)と低位修飾子(ファイル名)を大文字に変更していない場合、バックアップ/アーカイブ・クライアントを介して、直接名前オブジェクトにアクセスすることができなくなります。

例えば、オブジェクトがサーバーに {"FileSpaceName"}¥TEST¥MYDIRNAME¥file.txt で保存されている場合、file.txt オブジェクトを直接リストアしたり、照会したりすることはできません。これは、ファイルがサーバーへコピーされる前に、アプリケーションでファイル名を大文字に変更しなかったためです。これらのオブジェクトを処理する唯一の方法は、ワイルドカード文字を使用する方法です。例えば、¥TEST¥MYDIRNAME¥file.txt を照会するには、バックアップ/アーカイブ・クライアント・ユーザーは、オブジェクトがサーバーに送信される前に大文字に変更されていないオブジェクト名のすべての部分に、ワイルドカード文字を使用する必要があります。この file.txt ファイルを照会するためには、次のコマンドを使用する必要があります。

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

他の修飾子の他の部分も小文字で保存されている場合には、それらの修飾子もワイルドカードを使用して照会する必要があります。例えば、{"FileSpaceName"}¥TEST¥mydirname¥file.txt として保管されているオブジェクトを照会するには、次のコマンドを使用します。

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

次の例は、これらの概念を示すものです。Windows と UNIX または Linux のどちらの環境でも、完全な高位修飾子や低位修飾子を指定する必要はありません。ただし、完全な修飾子を指定しない場合は、ワイルドカード文字を使用する必要があります。

プラットフォーム	例
Windows	<p>ファイル・スペース MYFS 内のすべてのバックアップ・ファイルを照会するには、次のストリングを入力します。</p> <pre>dsmc q ba "{MYFS}¥*¥*"</pre> <p>それぞれの高位修飾子および低位修飾子ごとに、少なくとも 1 つのアスタリスク (*) を使用する必要があります。</p>

プラットフォーム	例
UNIX または Linux	<p>ファイル・スペース /A にあるすべてのバックアップ・ファイルを照会するには、次のストリングを入力します。</p> <pre>dsmc q ba "/A/*/*"</pre> <p>それぞれの高位修飾子および低位修飾子ごとに、少なくとも 1 つのアスタリスク (*) を使用する必要があります。</p>

API で使用できるバックアップ/アーカイブ・クライアント・コマンド

アプリケーション内では、バックアップ/アーカイブ・クライアント・コマンドのサブセットを使用できます。例えば、同じノード上または別のノード上で、他のユーザーが所有しているオブジェクトを表示し管理することができます。

同じノードまたは別のノードの他のユーザーが所有しているオブジェクトを表示および管理するには、以下のステップを行います。

1. **set access** コマンドによってアクセス権を与えます。
2. 所有者とノードを指定します。バックアップ/アーカイブ・コマンド・ラインから **fromowner** と **fromnode** オプションを使用して、所有者とノードを指定します。例えば次のとおりです。

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

77 ページの表 18 は、API オブジェクトに使用できる コマンドを示しています。

表 18. API オブジェクトで使用できるバックアップ/アーカイブ・クライアント・コマンド

コマンド	説明
Delete Archive	現行ユーザーが所有するアーカイブ・ファイルを削除することができます。set access コマンドの設定は、このコマンドには影響しません。
Delete Filespace	delete filespace コマンドは、API オブジェクトに 影響を及ぼします。
Query	<p>バックアップおよびアーカイブした API オブジェクト、または、他のユーザーが所有するオブジェクトまたは他のノードにあるオブジェクトを、バックアップ/アーカイブ・コマンド・ラインから照会できます。API オブジェクトの 照会に関しては、75 ページの『API オブジェクトの命名』を参照してください。</p> <p>set access 許可が与えられている、別のユーザーが所有するオブジェクトを照会するには、既存の -fromowner オプションを使用します。set access 許可が与えられている、別のノードに存在するオブジェクトを照会するには、既存の -fromnode オプションを使用します。詳しくは、115 ページの『dsmInitEx』を参照してください。</p>
Restore Retrieve	<p>注：これらのコマンドは、例外的に必要な場合にのみ使用してください。アプリケーション管理の鍵を使用して暗号化されている API オブジェクトは、暗号鍵がわかっているか、パスワード・ファイルに暗号鍵が保管されていれば、リストアまたはリトリブできます。透過的な暗号化を使用して暗号化された API オブジェクトは、バックアップ/アーカイブ・クライアントを使用してリストアまたはリトリブすることができません。</p> <p>これらのコマンドは、デフォルト・ファイル属性を使用して作成されるビット・ファイルとして、データを戻します。他のユーザーが所有する API オブジェクトまたは別のノードの API オブジェクトを、リストアまたはリトリブすることができます。set access コマンドによって、どのオブジェクトに限定するかが決まります。</p>

表 18. API オブジェクトで利用できるバックアップ/アーカイブ・クライアント・コマンド (続き)

コマンド	説明
Set Access	set access コマンドによって、別のユーザーが所有する API オブジェクトまたは別のノードの API オブジェクトを管理できます。

例

```
dsmc query backup f:¥*¥*
```

	Size		Backup Date	Mgmt Class	A/I File
	----		-----	-----	-----
API	1 B		11/14/2018 08:22:24	DEFAULT	A \\viola\f\$\dir1\test
	1 B		11/14/2018 08:21:41	DEFAULT	A \\viola\f\$\dir1\test

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

	Size		Backup Date	Mgmt Class	A/I File
	----		-----	-----	-----
API	1 B		11/15/2018 08:22:24	DEFAULT	A /home/user1/test
	1 B		11/15/2018 08:21:41	DEFAULT	A /home/user1/test

オペレーティング・システムのインターオペラビリティ

IBM Spectrum Protect API は、クロスプラットフォーム・インターオペラビリティをサポートします。UNIX システムまたは Linux システム上のアプリケーションは、Windows システムからバックアップされたファイル・スペースやオブジェクトを操作できます。同様に、Windows システムは、UNIX または Linux システムからバックアップされたファイル・スペースおよびオブジェクトを操作できます。

このタスクについて

デフォルトで、ある UNIX システムのオブジェクトの名前には、他の UNIX システムのオブジェクトの名前と互換性があります。デフォルトで、Windows システムのオブジェクトの名前には、UNIX システムのオブジェクトの名前と互換性がありません。いくつかのパラメーターが IBM Spectrum Protect ファイル・スペースのオブジェクトの命名を制御します。アプリケーションを適切にセットアップすると、Windows システムと UNIX システムの両方で実行されるアプリケーションでオブジェクトの名前を使用できます。オブジェクトのバックアップとリストアには、同じパラメーターを使用してください。

制約事項: ユニコードを使用する Windows アプリケーションは、UNIX システムで実行されるアプリケーションと互換性がないファイル・スペースを作成します。

手順

インターオペラビリティを使用するには、次のようにセットアップを完了してください。

1. 一貫性のある命名規則を設定してください。スラッシュ (/) や円記号 (¥) などの文字を **dir** 区切り文字として選択してください。ディレクトリー区切り文字をファイル・スペース名、高位修飾子、および低位修飾子の前に置きます。
2. **dsmInitEx** の呼び出し時に、**dirDelimiter** フィールドの値を、選択したディレクトリー区切り文字に設定し、**bCrossPlatform** を **bTrue** に設定します。
3. IBM Spectrum Protect インターフェースを使用するときは、**useUnicode** フラグを **bFalse** に設定してください。ユニコードのファイル名は、非ユニコードのファイル名と互換性がありません。

クライアント・ノード・プロキシ・サポートを使用した複数ノードのバックアップ

ストレージを共有している複数のノードのバックアップを、IBM Spectrum Protect サーバー上で 1 つの共通ターゲット・ノード名に統合できます。この方式は、バックアップを実行するシステムが時間の経過とともに変更される場合 (例えば、クラスターを使用している場合など) に便利です。また、**asnodename** オ

プッシュンを使用すると、バックアップを実行したシステムとは異なるシステムからデータをリストアすることもできます。

このタスクについて

dsmInitEx のオプション・ストリングで **asnodename** オプションを使用して、IBM Spectrum Protect サーバー上の特定のターゲット・ノード名の下で、データのバックアップ、アーカイブ、リストア、リトリブ、照会、または削除を実行します。**asnodename** オプションは、**dsm.opt** ファイルまたは **dsm.sys** ファイルの中で指定することもできます。

制約事項: ファイルを暗号化してからサーバーにバックアップする場合は特に、ターゲット・ノードを通常のノードとして使用しないでください。

手順

このオプションを使用可能にするには、以下の手順を実行します。

1. 共有データ環境内のすべてのノードに API クライアントをインストールします。
2. まだ各ノードが IBM Spectrum Protect サーバーに登録されていない場合は、登録します。共有データ環境内で使用されている各エージェント・ノードが共有する共通の「ターゲット」ノード名を登録します。
3. 共有データ環境内の各エージェント・ノードをサーバーに登録します。エージェント・ノード名は認証に使用されます。**asnodename** オプションが使用されている場合は、データの保管にエージェント・ノード名は使用されません。
4. IBM Spectrum Protect サーバー上のターゲット・ノード名にアクセスするために、**grant proxynode** コマンドを使用して共有環境内のすべてのノードにプロキシ権限を付与するよう管理者に依頼します。
5. 別のノードに代わってクライアント操作を実行する権限を付与されているクライアント・ノードを表示するために、**query proxynode** 管理クライアント・コマンドを使用します。この権限は、**grant proxynode** コマンドにより付与されます。または、このノードが他のどのノードのプロキシとして機能できるかを確認するために、**dsmQuery** コマンドを照会タイプ **qtProxyNodeAuth** と一緒に使用します。
6. アプリケーションで (TSMENCRKEY ではなく) ユーザー暗号化を使用してデータを暗号化している場合は、すべてのノードが同じ暗号鍵を使用していることを確認します。共有ノード環境内のすべてのバックアップ対象ファイルに、同じ暗号鍵を使用する必要があります。

関連タスク

[クライアント・ノード・プロキシ・サポートによるデータのバックアップ \(UNIX および Linux システム\)](#)

[クライアント・ノード・プロキシ・サポートによるデータのバックアップ \(Windows システム\)](#)

第 5 章 ユニコードでの API の使用

IBM Spectrum Protect API はユニコード UCS2、固定長、2 バイトのコード・ページをサポートします。このコード・ページには、日本語、中国語、ドイツ語などのすべての既知のコード・ページのためのコード・ポイントが入っています。これは 65,535 の固有なコード・ポイントをサポートします。

制約事項: この機能は Windows でのみ使用可能です。

ユニコードを使用すると、アプリケーションでは、同じマシンからどのような文字セットのファイル名でもバックアップし、リストアすることができます。例えば、英語のマシンを使用して、任意の言語コード・ページでファイル名をバックアップし、リストアできます。

ユニコードを使用する場合

ユニコード・アプリケーションを作成し、IBM Spectrum Protect ユニコード・インターフェースを利用することにより、複数の言語をサポートするご使用のアプリケーションを単純化することができます。

以下のいずれかの場合は、IBM Spectrum Protect ユニコード・インターフェースを使用してください。

- アプリケーションが既にユニコード用にコンパイルされており、マルチバイト文字セット (mbcs) に変換されてから IBM Spectrum Protect API を呼び出す場合。
- 新規アプリケーションを作成していて、アプリケーションをユニコード・サポート可能にする場合。
- アプリケーションが使用するストリングが、ユニコードを使用しているオペレーティング・システムまたは別のアプリケーションから渡される場合。

ユニコードが必要でなくなったときに、アプリケーションをコンパイルし直す必要はありません。

API は、dsm インターフェースのサポートを続行します。API SDK には、ユニコード API を使用する方法を示す `callmtu1.c` と `callmtu2.c` のサンプル・プログラムが入っています。これらのプログラムをコンパイルするには、**makemtu** を使用します。

ユニコードのセットアップ

ユニコードをセットアップして使用するには、特定の手順を実行する必要があります。それをすれば、API はサーバーにユニコードのファイル・スペースを登録し、そのファイル・スペース内のファイル名はすべてユニコード・ストリングになります。

制約事項: 同じファイル・スペースに、ユニコードと非ユニコードのファイル名を保管することはできません。

1. **-DUNICODE** フラグを指定してコードをコンパイルします。
2. アプリケーション内のすべてのストリングは `wchar` ストリングでなければなりません。
3. API 呼び出しについては、`tsmapitd.h` ファイル内の構造と `tsmapifp.h` ファイル内の関数定義に従ってください。
4. **tsmInitEx** 関数呼び出しで `useUnicode` フラグを `bTrue` に設定します。新規ファイル・スペースはどれも、ユニコード・ファイル・スペースとして登録されます。

既に登録済みの非ユニコードのファイル・スペースにデータを送信するときは、API はファイル名を今までどおり非ユニコードとして送信します。サーバー上の古いファイル・スペースの名前を `fsname_old` に変更して、新規データのための新しいユニコード・ファイル・スペースを開始してください。API は古いファイル・スペースから非ユニコード・データをリストアします。ファイル・スペースがユニコードであるかどうかを判別するには、ファイル・スペース照会で見られる **tsmQryRespFSDData** 構造内の **bIsUnicode** フィールドを使用してください。

各 **dsmXXX** 関数呼び出しに対応して **tsmXXX** 関数呼び出しがあります。この 2 つは、使用する構造が違います。UNICODE フラグを指定してコンパイルすると、すべての **tsmXXX** 関数呼び出し構造のストリング値は `dsChar_t` タイプになります。 `dsChar_r` は `wchar` にマップされます。これらのインターフェースには、他に違いはありません。

制約事項: どちらかの一方のインターフェースを使用してください。**dsmXXX** 関数呼び出しと **tsmXXX** 関数呼び出しのインターフェースを混用しないでください。必ず、IBM Spectrum Protect 構造および IBM Spectrum Protect バージョンの定義を使用してください。

定数の中には、引き続き **dsmapi**.h ファイルで定義されるものがあります。したがって、コンパイル時には、**dsmapi**.h ファイルと **tsmapi**.h ファイルの両方を使用する必要があります。

IBM Spectrum Protect インターフェースは、他のオペレーティング・システム、例えば、UNIX または Linux でも使用できますが、これらのオペレーティング・システムでは、**dsChar_t** タイプが **char** にマップされます。ユニコードは Windows オペレーティング・システムのみでサポートされるからです。アプリケーションの変種を 1 つだけ作成して、IBM Spectrum Protect インターフェースを使用して、複数のオペレーティング・システムでコンパイルすることができます。新しいアプリケーションを作成するときは、IBM Spectrum Protect インターフェースを使用してください。

既存のアプリケーションをアップグレードする場合は、次のようにします。

1. **dsmXXX** 関数呼び出し構造と呼び出しを、IBM Spectrum Protect インターフェース用に変換します。
2. 既存のファイル・スペースをマイグレーションします。
3. **useUnicode** フラグを **true** に設定して、新しいファイル・スペースをバックアップします。

注: ノードにアクセスするためにユニコード使用可能クライアントを使用した後は、API の古いバージョンを使用して、あるいは別のオペレーティング・システムからの API で、その同じノード名に接続することはできません。アプリケーションがプラットフォーム 相互間機能を使用する場合は、**Unicode** フラグは使用しないでください。ユニコードと非ユニコードのオペレーティング・システム間には、プラットフォーム相互間サポートはありません。

useUnicode フラグを使用可能にすると、すべてのストリング構造はユニコード・ストリングとして処理されます。サーバーでは、次のフィールドのみが真のユニコードとなります。

- ファイル・スペース名
- 高位
- 低位
- アーカイブ記述

残りのフィールドはすべて、ローカル・コード・ページの **mbcs** に変換されてから、サーバーに送信されます。ノード名など、フィールドは、**wchar** ストリングです。それらのストリングは、現在のロケールで有効でなければなりません。例えば、日本語のマシンで中国語名のファイルをバックアップできますが、ノード名は有効な日本語である必要があります。オプション・ファイルは現行のコード・ページのままです。ユニコードの **include-exclude** リストを作成する必要がある場合は、ファイル名とともに **incl excl** オプションを使用し、ファイル内でユニコード・パターンを使用するユニコード・ファイルを作成してください。

関連資料

[incl excl オプション](#)

第 6 章 API 関数呼び出し

83 ページの表 19 は、API 関数呼び出しのアルファベット順リストで、簡単な説明と、その 関数呼び出しについてさらに詳しい情報が記載されているページ位置を示しています。

エレメント	説明
目的	関数呼び出しについて説明します。
構文	<p>その関数呼び出しの実際の C コードです。このコードは、UNIX または Linux バージョンの <code>dsmapifp.h</code> ヘッダー・ファイルからコピーしたものです。191 ページの『付録 C API 関数定義ソース・ファイル』を参照してください。</p> <p>このファイルは、他のオペレーティング・システムでは少し異なります。他の オペレーティング・システムのアプリケーション・プログラマーは、API 定義が正確な構文になっているかどうか、それぞれのバージョンの ヘッダー・ファイル <code>dsmapifp.h</code> をチェックしてください。</p>
パラメーター	関数呼び出しの各パラメーターについて説明します。使用方法に基づいて、入力 (I) または出力 (O) として識別します。パラメーターのなかには、入出力の両方 (I/O) を指定するものもあります。ここで言及するデータ・タイプは、 <code>dsmapihd.h</code> ヘッダー・ファイルで定義されています。155 ページの『付録 B API タイプ定義ソース・ファイル』を参照してください。
戻りコード	その関数呼び出しに特定の戻りコードのリストです。通信エラー、サーバーの問題、あるいはユーザー・エラーのような一般的な システム・エラーもあります。これらのエラーはどの呼び出しでも起こりうるものであり、ここには示されていません。戻りコードは <code>dsmrc.h</code> ヘッダー・ファイルで定義されています。145 ページの『付録 A API 戻りコード・ソース・ファイル: <code>dsmrc.h</code> 』を参照してください。

表 19. API 関数呼び出し

関数呼び出しとページ位置	説明
85 ページの『 <code>dsmBeginGetData</code> 』	ストレージ内のオブジェクトのリストについて、リストアまたはリトリブ操作を開始します。
87 ページの『 <code>dsmBeginQuery</code> 』	情報を求めるために、IBM Spectrum Protect に対する照会要求を開始します。
91 ページの『 <code>dsmBeginTxn</code> 』	完全なアクションを開始する 1 つまたは複数のトランザクションを開始します。すべてのアクションが成功するか、どれも成功しないかのいずれになります。
92 ページの『 <code>dsmBindMC</code> 』	渡されたオブジェクトに管理クラスを 関連付け (バインド) ます。
93 ページの『 <code>dsmChangePW</code> 』	IBM Spectrum Protect パスワードを変更します。
94 ページの『 <code>dsmCleanUp</code> 』	これは、 <code>dsmSetUp</code> の 呼び出し時に使用されます。
95 ページの『 <code>dsmDeleteAccess</code> 』	オブジェクトのバックアップ・バージョンまたは アーカイブ・コピーのための現行の許可規則を削除します。
95 ページの『 <code>dsmDeleteFS</code> 』	ファイル・スペースをストレージから削除します。
96 ページの『 <code>dsmDeleteObj</code> 』	ストレージ内のバックアップ・オブジェクトをオフにしたり、アーカイブ・オブジェクトを削除したりします。
98 ページの『 <code>dsmEndGetData</code> 』	ストレージからオブジェクトを取得する <code>dsmBeginGetData</code> セッションを終了します。

表 19. API 関数呼び出し (続き)

関数呼び出しとページ位置	説明
98 ページの 『 dsmEndGetDataEx 』	送信された LAN フリーの合計バイト数を提供します。
99 ページの『 dsmEndGetObj 』	指定したオブジェクトに関するデータを取得 する dsmGetObj セッションを終了します。
99 ページの『 dsmEndQuery 』	dsmBeginQuery アクションの終了を知らせます。
100 ページの『 dsmEndSendObj 』	ストレージに送信される「データ終わり」を示します。
100 ページの 『 dsmEndSendObjEx 』	圧縮情報と、送信されたバイト数を提供します。
101 ページの『 dsmEndTxn 』	IBM Spectrum Protect トランザクションを終了します。
103 ページの『 dsmEndTxnEx 』	dsmGroupHandler 関数呼び出し で使用するグループ・リーダー・オブジェクト ID 情報を提供します。
104 ページの『 dsmGetData 』	IBM Spectrum Protect からバイト・ストリームのデータを取得し、それを呼び出し側のバッファーに入れます。
105 ページの『 dsmGetBufferData 』	IBM Spectrum Protect サーバーから、IBM Spectrum Protect によって割り振られたバッファーに入れられたデータを取得します。
106 ページの『 dsmGetNextQObj 』	直前の dsmBeginQuery 呼び出しから次の照会応答を取得し、それを呼び出し側の バッファーに入れます。
109 ページの『 dsmGetObj 』	データ・ストリームから要求されたオブジェクト・データ を取得し、それを呼び出し側のバッファーに入れます。
110 ページの『 dsmGroupHandler 』	与えられた入力に応じて、論理ファイル・グループにアクションを実行します。
111 ページの『 dsmInit 』	API セッションを開始し、クライアントをストレージに接続します。
115 ページの『 dsmInitEx 』	拡張検査を許可する追加のパラメーターを使用して、API セッションを開始します。
119 ページの『 dsmLogEvent 』	ユーザー・メッセージを、サーバー・ログ・ファイル、ローカル・エラー・ログ、またはその両方に記録します。
119 ページの『 dsmLogEventEx 』	ユーザー・メッセージを、サーバー・ログ・ファイル、ローカル・エラー・ログ、またはその両方に記録します。
121 ページの『 dsmQueryAccess 』	オブジェクトのバックアップ・バージョンまたはアーカイブ・コピーのためのすべてのアクセス許可 規則をサーバーに照会します。
121 ページの 『 dsmQueryApiVersion 』	アプリケーション・クライアントによってアクセス される API ライブラリー・バージョンについて照会要求を実行します。
122 ページの 『 dsmQueryApiVersionEx 』	アプリケーション・クライアントによってアクセス される API ライブラリー・バージョンについて照会要求を実行します。
122 ページの 『 dsmQueryCliOptions 』	ユーザーのオプション・ファイル内の重要なオプションの値を照会します。
123 ページの 『 dsmQuerySessInfo 』	dsmHandle に指定したセッションの操作に関連した情報について、IBM Spectrum Protect に照会要求を開始します。
124 ページの 『 dsmQuerySessOptions 』	dsmHandle の指定したセッション内での有効な 重要オプション値を照会します。

表 19. API 関数呼び出し (続き)

関数呼び出しとページ位置	説明
125 ページの『dsmRCMsg』	API 戻りコードに関連したメッセージ・テキストを取得します。
126 ページの『dsmRegisterFS』	サーバーに新規ファイル・スペースを登録します。
127 ページの『dsmReleaseBuffer』	IBM Spectrum Protect によって割り振られたバッファを返却します。
128 ページの『dsmRenameObj』	高位オブジェクト名または低位オブジェクト名を名前変更します。
129 ページの『dsmRequestBuffer』	バッファ・コピー除去のため、IBM Spectrum Protect によって割り振られたバッファを取得します。
130 ページの『dsmRetentionEvent』	オブジェクト ID のリストを、これらのオブジェクトに対して実行される保存イベント操作を指定して、サーバーに送信します。
131 ページの『dsmSendBufferData』	IBM Spectrum Protect によって割り振られたバッファのデータを送信します。
132 ページの『dsmSendData』	データのバイト・ストリームをバッファ経由で IBM Spectrum Protect に送信します。
134 ページの『dsmSendObj』	ストレージへの単一オブジェクトの送信要求を開始します。
137 ページの『dsmSetAccess』	他のユーザー、または他のノードに、オブジェクトのバックアップ・バージョンまたはアーカイブ・コピーへのアクセス、すべてのオブジェクトへのアクセス、または一部の選択したものへのアクセスを許可します。
138 ページの『dsmSetUp』	環境変数値を指定変更します。
140 ページの『dsmTerminate』	サーバーとのセッションを終了して、IBM Spectrum Protect 環境を終結処理します。
140 ページの『dsmUpdateFS』	ストレージ内のファイル・スペースを更新します。
141 ページの『dsmUpdateObj』	既にサーバー上にある活動バックアップ・オブジェクトに関連する objInfo 情報を更新します。または、アーカイブ済みオブジェクトを更新します。
142 ページの『dsmUpdateObjEx』	同じ名前を持つ複数のオブジェクトがある場合でも、特定のアーカイブ・オブジェクトに関連する objInfo 情報を更新します。または、活動バックアップ・オブジェクトを更新します。

関連資料

[API 戻りコード](#)

dsmBeginGetData

dsmBeginGetData 関数呼び出しは、ストレージ内のオブジェクトのリストについてリストアまたはリトリブ操作を開始します。このオブジェクトのリストは、**dsmGetList** 構造にあります。アプリケーションは、**dsmBeginGetData** への呼び出しの前に照会からの値を使用して、このリストを作成します。

呼び出し側は、オブジェクト照会から取得された restore order (リストア順序) フィールドを使用して、この呼び出しに含まれているリストをソートする必要があります。これによって、データ・テープの巻き戻しや再マウントを行う必要なしに、考えられる最も効率のよい方法でストレージからオブジェクトがリストアされることが保証されます。

オブジェクト全体を取得する場合、最大 **dsmGetList.numObjID** は **DSM_MAX_GET_OBJ** です。オブジェクトの一部を取得する場合、最大は **DSM_MAX_PARTIAL_GET_OBJ** です。

リスト内の各オブジェクトを取得するには、**dsmBeginGetData** の呼び出しに続いて1つ以上の **dsmGetObj** 呼び出しを行う必要があります。各オブジェクトを取得した後で、またはそのオブジェクトについてそれ以上データが必要でなくなったら、**dsmEndGetObj** 呼び出しを送信します。

すべてのオブジェクトを取得したとき、または **dsmEndGetObj** がキャンセルされた場合は、**dsmEndGetData** 呼び出しを送信します。この後、再びサイクルを開始できます。

構文

```
dsInt16_t dsmBeginGetData  
(dsUInt32_t dsmHandle,  
 dsBool_t mountWait,  
 dsmGetType_t getType,  
 dsmGetList *dsmGetObjListP);
```

パラメーター

dsUInt32_t dsmHandle (I)

この呼び出しを直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

dsBool_t mountWait (I)

必要なデータが現在オフラインの場合に、オフライン・メディアがマウントされるまでアプリケーション・クライアントが待機するかどうかを示す真または偽のブール値。mountWait が真の場合、アプリケーションはサーバーに必要なメディアがマウントされるのを待ちます。アプリケーションは、メディアがマウントされるか、または要求が取り消されるまで待ちます。

dsmGetType_t getType (I)

取得するオブジェクトがどのタイプであるかを示す、**gtBackup** および **gtArchive** からなるタイプのリスト。

dsmGetList *dsmGetObjListP (I)

リストアまたはリトリブするオブジェクトまたは部分オブジェクトについての情報が入っている構造。この構造はオブジェクト ID のリストを指し、部分オブジェクトのリトリブまたはリストアの場合は、関連したオフセットおよび長さのリストを指します。アプリケーションで部分オブジェクトのリストアまたはリトリブ機能を使用する場合は、**dsmGetList.stVersion** フィールドを **dsmGetListPORVersion** に設定してください。部分オブジェクトのリストアまたはリトリブでは、送信中にデータを圧縮することはできません。これを強制的に行うには、**ObjAttr.objCompressed** を **bTrue** に設定します。

この構造について詳しくは、70 ページの図 19 および 155 ページの『付録 B API タイプ定義ソース・ファイル』を参照してください。

部分オブジェクトのリストアまたはリトリブについて詳しくは、64 ページの『部分オブジェクト・リストア/リトリブ』を参照してください。

戻りコード

戻りコード番号を括弧 () 内に示してあります。

表 20. *dsmBeginGetData* の戻りコード

戻りコード	説明
DSM_RC_ABORT_INVALID_OFFSET (33)	部分オブジェクト・リトリブ中に指定されたオフセットが、オブジェクトの長さより大きい。
DSM_RC_ABORT_INVALID_LENGTH (34)	部分オブジェクト・リトリブ中に指定された長さがオブジェクトの長さより大きいか、またはオフセットにこの長さを加えると、オブジェクトの終わりを超える。
DSM_RC_NO_MEMORY (102)	要求を完了するための RAM が残っていない。

表 20. *dsmBeginGetData* の戻りコード (続き)

戻りコード	説明
DSM_RC_NUMOBJ_EXCEED (2029)	dsmGetList.numObjId が DSM_MAX_GET_OBJ より 大である。
DSM_RC_OBJID_NOTFOUND (2063)	オブジェクト ID が見つからなかった。このオブジェクトはリストアされなかった。
DSM_RC_WRONG_VERSION_PARM (2065)	アプリケーション・クライアントの API バージョンが IBM Spectrum Protect ライブラリー・バージョン と異なる。

dsmBeginQuery

dsmBeginQuery 関数呼び出しは、データ、ファイル・スペース、および管理クラス に関する情報を得るための照会要求を、サーバーに対して開始します。

具体的に、**dsmBeginQuery** では以下の情報を照会できます。

- アーカイブ・データ
- バックアップ・データ
- 活動バックアップ・データ
- ファイル・スペース
- 管理クラス

この呼び出しから戻される照会データは、1 つ以上の **dsmGetNextQObj** 呼び出しによって取得されます。照会が完了すると、**dsmEndQuery** 呼び出しが送信されます。

構文

```
dsInt16_t dsmBeginQuery
(dsUInt32_t dsmHandle,
 dsmQueryType queryType,
 dsmQueryBuff *queryBuffer);
```

パラメーター

dsUInt32_t dsmHandle (I)

この呼び出しを直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

dsmQueryType queryType (I)

実行する照会のタイプを識別します。以下のいずれかのオプションを割り当てます。

qtArchive

アーカイブ・オブジェクトを照会します。

qtBackup

バックアップ・オブジェクトを照会します。

qtBackupActive

アプリケーション・プログラマーが渡した全ファイル・スペース名に関する 活動バックアップ・オブジェクトのみを照会します。この照会は「高速パス」と呼ばれ、ストレージから活動オブジェクトを照会する効率的な方法です。

前提条件: UNIX または Linux オペレーティング・システムに root ユーザーとしてログオンしている必要があります。

qtFilespace

登録済みファイル・スペースを照会します。

qtMC

定義済み管理クラスを照会します。

qtBackupGroups

クローズされているグループを照会します。

qtOpenGroups

オープンしているグループを照会します。

qtProxyNodeAuth

このノードが他のどのノードのプロキシとして機能できるかを照会します。

qtProxyNodePeer

ターゲットが同じである対等ノードはどのノードかを照会します。

dsmQueryBuff *queryBuffer (I)

特定のデータ構造にマップされるバッファを指すポインターを識別します。この構造は、ユーザーが渡す照会タイプに関連付けられます。これらの構造は、各照会タイプごとに選択基準を含んでいます。実行する照会の有効範囲を指定するために、各構造内のフィールドに入力してください。各構造内の `stVersion` フィールドには、構造のバージョン番号が入っています。

データ構造およびその関連フィールドには、以下の項目があります。

qryArchiveData

objName

完全オブジェクト名。アスタリスク (*) または疑問符 (?) などのワイルドカード文字を、名前の高位と低位の部分で使用できます。アスタリスクはゼロ個以上の文字に一致し、疑問符は1文字に一致します。objName の objType フィールドには、以下のいずれかの値を指定できます。

- DSM_OBJ_FILE
- DSM_OBJ_DIRECTORY
- DSM_OBJ_ANY_TYPE

高位名および低位名の詳しくは、[24 ページの『高位名と低位名』](#)のトピックを参照してください。

owner

オブジェクトの所有者名。

insDateLowerBound

オブジェクトがアーカイブされた 挿入日の下限。年のコンポーネントを `DATE_MINUS_INFINITE` に設定すると、下限を無制限にします。

insDateUpperBound

オブジェクトがアーカイブされた 挿入日の上限。年のコンポーネントを `DATE_PLUS_INFINITE` に設定すると、上限を無制限にします。

expDateLowerBound

有効期限の下限。年のコンポーネントを `DATE_MINUS_INFINITE` に設定すると、下限を無制限にします。

expDateUpperBound

有効期限の上限。管理クラスの **RETVer** の `NOLIMIT` の設定に一致させるため、年のコンポーネントを `DATE_PLUS_INFINITE` に設定します。

descr

アーカイブ記述。すべての記述を検索するには、アスタリスク (*) を入力します。

qryBackupData

objName

完全オブジェクト名。アスタリスク (*) または疑問符 (?) などのワイルドカード文字を、名前の高位と低位の部分で使用できます。アスタリスクはゼロ個以上の文字に一致し、疑問符は1文字に一致します。objName の objType フィールドには、以下のいずれかの値を指定できます。

- DSM_OBJ_FILE
- DSM_OBJ_DIRECTORY
- DSM_OBJ_ANY_TYPE

高位名および低位名の詳しくは、[24 ページの『高位名と低位名』](#)のトピックを参照してください。

owner

オブジェクトの所有者名。

objState

以下のいずれかのオブジェクト状態を照会できます。

- DSM_ACTIVE
- DSM_INACTIVE
- DSM_ANY_MATCH

pitDate

時刻指定値。このフィールドを指定して照会を行うと、この日時より前にバックアップされた最新のオブジェクトが返されます。objState は、活動でも非活動でもかまいません。

pitDate より前に削除されたオブジェクトは返されません。以下に例を挙げます。

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

金曜日に、水曜日の 12:00:00 a.m. の時刻指定値を使用して、この照会を呼び出すと、次の情報が返されます。

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

この呼び出しでは、DEF は返されません。そのオブジェクトは時刻指定値より前に削除されているからです。

qryABackupData

objName

完全オブジェクト名。アスタリスク (*) または疑問符 (?) などのワイルドカード文字を、名前の高位と低位の部分で使用できます。アスタリスクはゼロ個以上の文字に一致し、疑問符は1文字に一致します。objName の objType フィールドには、以下のいずれかの値を指定できます。

- DSM_OBJ_FILE
- DSM_OBJ_DIRECTORY
- DSM_OBJ_ANY_TYPE

高位名および低位名の詳しくは、[24 ページの『高位名と低位名』](#)のトピックを参照してください。

qryFSData

fsName

このフィールドに特定のファイル・スペースの名前を入力するか、または、すべての登録済みファイル・スペースに関する情報をリトリブするときはアスタリスク (*) を入力してください。

qryMCData

mcName

特定の管理クラスの名前を入力するか、または、すべての管理クラスに関する情報を検索するときは空ストリング (『 』) を入力してください。

注: アスタリスク (*) は使用できません。

mcDetail

管理クラスのバックアップおよびアーカイブ・コピー・グループに関する情報が返されるかどうかを決定します。以下の値が有効です。

- bTrue
- bFalse

qryBackupGroup:

groupType

グループ・タイプは DSM_GROUPTYPE_PEER です。

fsName

ファイル・スペース名。

owner

所有者 ID。

groupLeaderObjId

グループ・リーダー・オブジェクト ID。

objType

オブジェクト・タイプ。

qryProxyNodeAuth:

targetNodeName

ターゲット・ノード名。

peerNodeName

ピア・ノード名。

hlAddress

高位名のピア・アドレス。

llAddress

低位名のピア・アドレス。

qryProxyNodePeer:

targetNodeName

ターゲット・ノード名。

peerNodeName

ピア・ノード名。

hlAddress

高位名のピア・アドレス。

llAddress

低位名のピア・アドレス。

戻りコード

次の表は、**dsmBeginQuery** 関数呼び出しの戻りコードの説明です。

表 21. *dsmBeginQuery* の戻りコード

戻りコード	戻りコード番号	説明
DSM_RC_NO_MEMORY	102	要求を完了するのに十分なメモリーがない。
DSM_RC_FILE_SPACE_NOT_FOUND	124	指定されたファイル・スペースが見つからなかった。
DSM_RC_NO_POLICY_BLK	2007	サーバー・ポリシー情報が使用できなかった。
DSM_RC_INVALID_OBJTYPE	2010	オブジェクト・タイプが無効。
DSM_RC_INVALID_OBJOWNER	2019	オブジェクト所有者名が無効。
DSM_RC_INVALID_OBJSTATE	2024	オブジェクトの条件が無効。
DSM_RC_WRONG_VERSION_PAIRM	2065	アプリケーション・クライアントの API バージョンが IBM Spectrum Protect ライブラリー・バージョンと異なる。

dsmBeginTxn

dsmBeginTxn 関数呼び出しは、完全なアクションを開始する、1 つ以上の IBM Spectrum Protect トランザクションを開始します。アクションはすべて成功するか、またはどれも成功しないかのいずれかです。アクションには、単一の呼び出しまたは一連の呼び出しを指定することができます。例えば、**dsmSendObj** 呼び出しの後にいくつかの **dsmSendData** 呼び出しが続く場合、これを単一のアクションと見なすことができます。同様に、バックアップするオブジェクトを含んでいるデータ域を示す **dataBlkPtr** を伴う **dsmSendObj** 呼び出しも単一のアクションと見なされます。

データ転送操作のためには、複数のオブジェクトを一緒にして単一トランザクションにグループ化することを試みてください。オブジェクトをグループ化することによって、IBM Spectrum Protect システムのパフォーマンスが顕著に向上します。クライアントおよびサーバーのどちらから見ても、トランザクションの開始および終了を行うたびに、一定のオーバーヘッドがかかります。

単一トランザクション内で実行できることには制限があります。このような制限には、次のものが含まれます。

- 単一トランザクションで送信または削除できるオブジェクトの最大数。この制限値は、**dsmQuerySessInfo** が *ApiSessInfo.maxObjPerTxn* フィールドに戻すデータに置かれます。これは、*TxnGroupMax* サーバー・オプションに対応します。
- サーバーに送信される単一トランザクション内のすべてのオブジェクト (バックアップまたはアーカイブ) はすべて、そのオブジェクトにバインドしている管理クラスに定義されているのと同じコピー宛先をもっていなければならない。この値は、**dsmBindMC** が **mcBindKey.backup_copy_dest** または **mcBindKey.archive_copy_dest** フィールドに戻すデータに置かれます。

API を使用すれば、アプリケーション・クライアントはこれらの制限値をモニターし制御することができるか、または API 自体がこれらの制限値をモニターし、そのうちの 1 つ以上の制限値に達したときに、API 呼び出しからの適切な戻りコードを介して、アプリケーション・クライアントにそれを通知することができます。

1 対の **dsmBeginTxn** 呼び出しと **dsmEndTxn** 呼び出しの間のアクションのセットを最適化するために、**dsmBeginTxn** 呼び出しと **dsmEndTxn** 呼び出しを必ず突き合わせてください。

構文

```
dsInt16_t dsmBeginTxn (dsUInt32_t dsmHandle);
```

パラメーター

dsUInt32_t dsmHandle (I)

この呼び出しを直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

戻りコード

戻りコード番号を括弧 () 内に示してあります。

表 22. *dsmBeginTxn* の戻りコード

戻りコード	説明
DSM_RC_ABORT_NODE_NOT_AUTHORIZED (36)	FROMNODE または FROMOWNER は TXN 操作では許可されません。

dsmBindMC

dsmBindMC 関数呼び出しは、渡されたオブジェクトを管理クラスに関連付け (バインド) ます。オブジェクトは、オプション・ファイル内に示されている include-exclude リストを介して渡されます。特定の管理クラスについて、包含リストに一致するものが見つからない場合は、デフォルト管理クラスが割り当てられます。除外リストは、オブジェクトのバックアップがとられないようにできますが、アーカイブされないようにはできません。

アプリケーション・クライアントは、**mcBindKey** 構造に戻されるパラメーターを使用して、当該オブジェクトをバックアップまたはアーカイブすべきかどうか、またはコピー先が異なるために新規トランザクションを開始する必要があるかどうかを判別することができます。詳しくは、**dsmBeginTxn** を参照してください。

すべてのオブジェクトには管理クラスが関連付けられていなければならないので、**dsmSendObj** を呼び出す前に **dsmBindMC** を呼び出してください。この呼び出しはトランザクションの内側でも外側でも実行することができます。例えば、複数オブジェクトのトランザクション内で、**dsmBindMC** が、オブジェクトのコピー先を直前のオブジェクトと異なるものであるとしている場合、トランザクションを終了し、新規トランザクションを開始する必要があります。この場合、**dsmBindMC** は、当該オブジェクトについては既に実行されているので、もう一度実行する必要はありません。

構文

```
dsInt16_t dsmBindMC  
(dsUInt32_t dsmHandle,  
 dsmObjName *objNameP,  
 dsmSendType sendType,  
 mcBindKey *mcBindKeyP);
```


パラメーター

dsUInt32_t dsmHandle (I)

この呼び出しを直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

dsmObjName *objNameP (I)

ファイル・スペース名、高位オブジェクト名、低位オブジェクト名、および オブジェクト・タイプを含む構造を指すポインター。

dsmSendType sendType (I)

この管理クラス・バインドがアーカイブ送信またはバックアップ送信の どちらのために行われるかを識別します。この呼び出しに指定 できる値は、次のとおりです。

名前	説明
stBackup	バックアップ・オブジェクト
stArchive	アーカイブ・オブジェクト
stBackupMountWait	バックアップ・オブジェクト
stArchiveMountWait	アーカイブ・オブジェクト

dsmBindMC 呼び出しの **stBackup** と **stBackupMountWait** は同等であり、**stArchive** と **stArchiveMountWait** も同等です。

mcBindKey *mcBindKeyP (O)

これは、管理クラス情報が戻される **mcBindKey** 構造のアドレスです。アプリケーション・クライアントは、ここに戻される情報を使用して、当該オブジェクトが複数オブジェクト・トランザクション内に収まるかどうか判別したり、そのオブジェクトにバインドされている管理クラスについて 管理クラス照会を行ったりすることができます。

戻りコード

戻りコード番号を括弧 () 内に示してあります。

表 23. *dsmBindMC* の戻りコード

戻りコード	説明
DSM_RC_NO_MEMORY (102)	要求を完了するための RAM が残っていない。
DSM_RC_INVALID_PARM (109)	渡されたパラメーターの 1 つに無効値がある。
DSM_RC_TL_EXCLUDED (185)	バックアップ・オブジェクトが除外されていて、送信できない。
DSM_RC_INVALID_OBJTYPE (2010)	オブジェクト・タイプが無効。
DSM_RC_INVALID_SENDDTYPE (2022)	送信タイプが無効。
DSM_RC_WRONG_VERSION_PARM (2065)	アプリケーション・クライアントの API バージョンが IBM Spectrum Protect ライブラリーの バージョンと異なる。

dsmChangePW

dsmChangePW 関数呼び出しは IBM Spectrum Protect のパスワードを変更します。UNIX または Linux のような複数ユーザーのオペレーティング・システムの場合、この呼び出しを使用できるのは root ユーザーまたは許可ユーザーのみです。

Windows オペレーティング・システムでは、そのパスワードを **dsm.opt** ファイル内に指定できます。この状態では、**dsmChangePW** は **dsm.opt** ファイルを更新しません。**dsmChangePW** の呼び出しを行った後で、**dsm.opt** ファイルを別個に更新する必要があります。

dsmInitEx が DSM_RC_VERIFIER_EXPIRED を戻した場合、この呼び出しは正常に処理されています。この状態でこの **dsmChangePW** 呼び出しが失敗した場合、このセッションは終了します。

別の理由で **dsmChangePW** が呼び出された場合には、戻りコードに関係なく、セッションはオープンのままです。

構文

```
dsInt16_t dsmChangePW
(dsUInt32_t dsmHandle,
char      *oldPW,
char      *newPW);
```

パラメーター

dsUInt32_t dsmHandle (I)

この呼び出しを直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

char *oldPW (I)

呼び出し側の旧パスワード。最大長は DSM_MAX_VERIFIER_LENGTH です。

char *newPW (I)

呼び出し側の新規パスワード。最大長は DSM_MAX_VERIFIER_LENGTH です。

戻りコード

戻りコード番号を括弧 () 内に示してあります。

表 24. *dsmChangePW* の戻りコード

戻りコード	説明
DSM_RC_ABORT_BAD_VERIFIER (6)	間違ったパスワードが入力されました。
DSM_RC_AUTH_FAILURE (137)	認証に失敗した。旧パスワードが正しくない。
DSM_RC_NEWPW_REQD (2030)	新規パスワードに値を入力する必要がある。
DSM_RC_OLDPW_REQD (2031)	旧パスワードに値を入力する必要がある。
DSM_RC_PASSWD_TOOLONG (2103)	指定されたパスワードが長すぎる。
DSM_RC_NEED_ROOT (2300)	API の呼び出し元は、root ユーザーまたは許可ユーザーでなければならない。

dsmCleanUp

dsmCleanUp 関数呼び出しは、**dsmSetUp** の呼び出し時に使用されます。**dsmCleanUp** 関数呼び出しは、**dsmTerminate** の後に呼び出さなければなりません。**dsmCleanUp** を呼び出した後は、他の呼び出しを使用することはできません。

この呼び出しに特定の戻りコードはありません。

構文

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

パラメーター

dsBool_t mtFlag (I)

このパラメーターは、API を単一スレッド・モードで使用したか、またはマルチスレッド・モードで使用したかを示します。指定できる値は、次のとおりです。

- DSM_SINGLETHREAD
- DSM_MULTITHREAD

dsmDeleteAccess

dsmDeleteAccess 関数呼び出しは、オブジェクトの バックアップ・バージョンまたはアーカイブ・コピーのための現行の許可規則を削除します。許可規則を削除すると、その規則によって指定されていた、ファイルに対してユーザーが持っていた アクセス許可は取り消されます。

dsmDeleteAccess を使用して削除できるのは、一度に 1 つの 規則のみです。**dsmQueryAccess** コマンドで 規則 ID を取得してください。

この呼び出しに特定の戻りコードはありません。

構文

```
dsInt16_t DSMLINKAGE dsmDeleteAccess
                (dsUInt32_t      dsmHandle,
                 dsUInt32_t      ruleNum) ;
```

パラメーター

dsUInt32_t dsmHandle (I)

この呼び出しを直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

dsUInt32_t ruleNum (I)

削除するアクセス規則の規則 ID。この値は、**dsmQueryAccess** 関数呼び出しで取得します。

dsmDeleteFS

dsmDeleteFS 関数呼び出しは、ストレージからファイル・スペースを削除します。ファイル・スペースを削除するには、IBM Spectrum Protect 管理者から適切な許可を得なければなりません。必要な許可を得ているかどうかは、**dsmQuerySessInfo** を呼び出せば分かります。この関数呼び出しは、*archDel* および *backDel* と呼ばれる 2 つのフィールドを含む、タイプ *ApiSessInfo* の データ構造を戻します。

注：

- UNIX または Linux オペレーティング・システムでは、root ユーザーまたは許可ユーザーのみがファイル・スペースを削除できます。
- 削除したいファイル・スペースにバックアップ・バージョンがある場合は、バックアップ削除権限 (*backDel* = BACKDEL_YES) が必要です。削除したいファイル・スペースに アーカイブ・コピーがある場合は、アーカイブ削除権限 (*archDel* = ARCHDEL_YES) が必要です。削除したいファイル・スペースにバックアップ・バージョンとアーカイブ・コピーの両方がある場合は、両方のタイプの削除権限が必要です。
- アーカイブ・マネージャー・サーバーを使用すると、ファイル・スペースは 実際には除去されません。この関数呼び出しでは、ファイル・スペースが実際に削除されなくても *rc*=0 が返されます。ファイル・スペースが削除されたことを検査する唯一の方法は、サーバーに対してファイル・スペース照会を発行することです。
- IBM Spectrum Protect サーバーのファイル・スペースの削除 (delete file-space) 関数は、バックグラウンド・プロセスです。戻りコードを渡す前に検出されたエラー以外のエラー は、IBM Spectrum Protect サーバー・ログに記録されます。

構文

```
dsInt16_t dsmDeleteFS
(dsUInt32_t dsmHandle,
char *fsName,
unsigned char repository);
```

パラメーター

dsUInt32_t dsmHandle (I)

この呼び出しを直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

char *fsName (I)

削除するファイル・スペース名を指すポインター。ワイルドカード文字は使用できません。

unsigned char repository (I)

このパラメーターは、削除するファイル・スペースがバックアップ・リポジトリか、アーカイブ・リポジトリか、あるいはその両方かを示します。このフィールドに指定できる値は、次のとおりです。

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

戻りコード

戻りコード番号を括弧 () 内に示してあります。

表 25. *dsmDeleteFS* の戻りコード

戻りコード	説明
DSM_RC_ABORT_NOT_AUTHORIZED (27)	ファイル・スペースを削除するために必要な権限をもっていない。
DSM_RC_INVALID_REPOS (2015)	リポジトリの値が無効。
DSM_RC_FSNAME_NOTFOUND (2060)	ファイル・スペース名が見つからない。
DSM_RC_NEED_ROOT (2300)	API の呼び出し側は root ユーザーでなければならない。

dsmDeleteObj

dsmDeleteObj 関数呼び出しは、バックアップ・オブジェクトの非活動化、バックアップ・オブジェクトの削除、またはストレージ内のアーカイブ・オブジェクトの削除を行います。**dtBackup** タイプは、現在の活動バックアップ・コピーのみを非活動化します。**dtBackupID** タイプは、どのオブジェクト ID が指定されていてもそれをサーバーから除きます。この関数は、トランザクション内から呼び出します。

詳しくは、**dsmBeginTxn** を参照してください。

制約事項: 保存セットに含まれているバックアップ・オブジェクトを削除できません。長期間のデータ保持要件を満たすために、これらのファイルはサーバー・ストレージに保持され、保存セットの独自の有効期限の日付 (この日付を過ぎると削除の対象) に従って有効期限が切れます。

dsmDeleteObj を送信する前に、**delInfo** に関する情報を取得するために [33 ページの『IBM Spectrum Protect システムの照会』](#)で説明した照会シーケンスを送信します。**dsmGetNextQObj** の呼び出しは、バックアップ照会の場合は **qryRespBackupData** という名前のデータ構造を返し、アーカイブ照会の場合は **qryRespArchiveData** という名前のデータ構造を返します。これらのデータ構造は、**delInfo** に必要な情報を含んでいます。

maxObjPerTxn の値は、単一トランザクションで削除できるオブジェクトの最大数を決定します。この値を取得するには、**dsmQuerySessInfo** を呼び出します。

ヒント: ユーザーのノードは、管理者によって設定された適切な許可を持っている必要があります。アーカイブ・オブジェクトを削除するには、アーカイブ削除権限が必要です。バックアップ・オブジェクトを非活動化するとき、バックアップ削除権限は必要ありません。

構文

```
dsInt16_t dsmDeleteObj
(dsUInt32_t dsmHandle,
 dsmDelType delType,
 dsmDelInfo delInfo)
```

パラメーター

dsUInt32_t dsmHandle (I)

この呼び出しを直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

dsmDelType delType (I)

どのタイプのオブジェクト (バックアップまたはアーカイブ) を削除すべきかを指示します。指定できる値は、次のとおりです。

名前	説明
----	----

dtArchive	削除するオブジェクトは、前にアーカイブされたものである。
------------------	------------------------------

dtBackup	非活動化するオブジェクトは、前にバックアップされたものである。
-----------------	---------------------------------

dtBackupID	削除するオブジェクトは、前にバックアップされたものである。
-------------------	-------------------------------

制約事項: この **delType** を **objID** とともに使用すると、バックアップ・オブジェクトがサーバーから除去されます。オブジェクトを削除できるのは、そのオブジェクトの所有者のみです。

オブジェクトの任意のバージョン (活動または非活動) を削除できます。サーバーがバージョンの調整を行います。オブジェクトの活動バージョンを削除した場合、最初の非活動バージョンが活動状態になります。オブジェクトの非活動バージョンを削除すると、それより古いすべてのバージョンの順位が前に進みます。ノードは、**backDel** 許可を指定して登録されている必要があります。

dsmDelInfo delInfo (I)

オブジェクトを識別するフィールドをもつ構造。このフィールドは、オブジェクトがバックアップ・オブジェクトかまたはアーカイブ・オブジェクトかによって異なります。バックアップ・オブジェクトを非活動化するための構造は、**delBack** と呼ばれ、オブジェクト名とオブジェクトのコピー・グループが含まれます。アーカイブ・オブジェクトのための構造は、**delArch** と呼ばれ、オブジェクト ID が含まれます。

バックアップ・オブジェクトを除去するための構造 **delBackID** には、オブジェクト ID が含まれます。

戻りコード

戻りコード番号を括弧 () 内に示してあります。

表 26. *dsmDeleteObj* の戻りコード

戻りコード	説明
DSM_RC_FS_NOT_REGISTERED (2061)	ファイル・スペース名が登録されていない。

表 26. *dsmDeleteObj* の戻りコード (続き)

戻りコード	説明
DSM_RC_WRONG_VERSION_PARM (2065)	アプリケーション・クライアントの API バージョンが IBM Spectrum Protect ライブラリーのバージョンと異なる。

dsmEndGetData

dsmEndGetData 関数呼び出しは、ストレージからオブジェクトを取得する **dsmBeginGetData** セッションを終了します。

dsmEndGetData 関数呼び出しは、リストアする必要があるすべてのオブジェクトが処理された後に開始され、取得処理を早期に終了します。**dsmEndGetData** を呼び出して、**dsmBeginGetData** セッションを終了してからでないと、他の処理を続けることはできません。

dsmEndGetData がいつ呼び出されたかによって、API は、処理を停止する前に、一部のデータ・ストリームの処理を完了させる必要があることがあります。したがって、呼び出し側は、この呼び出しからの即時の戻りを期待することはできません。アプリケーションが即時にセッションをクローズし、リストアを終了する必要がある場合は、**dsmTerminate** を使用してください。

この呼び出しに特定の戻りコードはありません。

構文

```
dsInt16_t dsmEndGetData (dsUInt32_t dsmHandle);
```

パラメーター

dsUInt32_t dsmHandle (I)

この呼び出しを直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

dsmEndGetDataEx

dsmEndGetDataEx 関数呼び出しは、送信された LAN フリー 合計バイト数を提供します。これは **dsmEndGetData** 関数呼び出しの拡張です。

構文

この呼び出しに特定の戻りコードはありません。

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

パラメーター

dsmEndGetDataExIn_t *dsmEndGetDataExInP (I)

セッションを識別し、そのセッションを後続の呼び出しに関連付ける「オブジェクト取得終了の dsmHandle」を渡します。

dsmEndGetDataExOut_t *dsmEndGetDataExOutP (O)

この構造には入力パラメーターが含まれます。

totalLFBytesRecv

受信した LAN フリーの合計バイト数。

dsmEndGetObj

dsmEndGetObj 関数呼び出しは、指定したオブジェクトに関するデータを取得 する **dsmGetObj** セッションを終了します。

dsmEndGetObj は、オブジェクトに関するデータ終わりが受信された後に開始 します。これは、このオブジェクトに関するすべてのデータ が受信されたこと、またはこのオブジェクトに関してこれ以上データを 受信しないことを示します。別の **dsmGetObj** 呼び出しを開始するときは、その前に **dsmEndGetObj** を呼び出す 必要があります。

dsmEndGetObj がいつ呼び出されるかによって、API は、処理を停止する前に、一部のデータ・ストリームの処理を完了させる必要があることがあります。したがって、呼び出し側は、この呼び出しからの即時の戻りは期待できません。

構文

```
dsInt16_t dsmEndGetObj  
(dsUInt32_t dsmHandle);
```

パラメーター

dsUInt32_t dsmHandle (I)

この呼び出しを直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

戻りコード

戻りコード番号を括弧 () 内に示してあります。

表 27. *dsmEndGetObj* の戻りコード

戻りコード	説明
DSM_RC_NO_MEMORY (102)	要求を完了するための RAM が残っていない。

dsmEndQuery

dsmEndQuery 関数呼び出しは **dsmBeginQuery** の終わりを知らせます。アプリケーション・クライアントは、照会を完了するために **dsmEndQuery** を送信 します。この呼び出しを送信するのは、**dsmGetNextQObj** を介してすべての照会応答を取得した後であるか、あるいはすべてのデータが返される前に照会を終了する場合です。

ヒント：この場合、IBM Spectrum Protect は、引き続き照会データをサーバーからクライアントに送信しますが、API は残りのデータをすべて廃棄します。

dsmBeginQuery を送信した後は、**dsmEndQuery** を送信しなければ、他の活動を開始できません。

この呼び出しに特定の戻りコードはありません。

構文

```
dsInt16_t dsmEndQuery  
(dsUInt32_t dsmHandle);
```

パラメーター

dsUint32_t dsmHandle (I)

この呼び出しを直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

dsmEndSendObj

dsmEndSendObj 関数呼び出しは、ストレージに送信されるデータの終わりを示します。

dsmSendObj および **dsmSendData** 呼び出しからのデータ終わりを示すには、**dsmEndSendObj** 関数呼び出しを入力する必要があります。これを実行しない場合にはプロトコル違反が発生します。

構文

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

パラメーター

dsUint32_t dsmHandle (I)

この呼び出しを直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

戻りコード

戻りコード番号を括弧 () 内に示してあります。

表 28. *dsmEndSendObj* の戻りコード

戻りコード	説明
DSM_RC_NO_MEMORY (102)	この要求を完了するための RAM が残っていない。

dsmEndSendObjEx

dsmEndSendObjEx 関数呼び出しは、処理されたバイト数に関する追加情報を提供します。この情報には、合計送信バイト数、圧縮情報、LAN フリー・バイト数、および重複排除情報が含まれます。

dsmEndSendObjEx 関数呼び出しは、**dsmEndSendObj** 関数呼び出しの拡張です。

構文

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

パラメーター

dsmEndSendObjExIn_t *dsmEndSendObjExInP (I)

このパラメーターは、セッションを識別し、そのセッションを後続の呼び出しに関連付ける「オブジェクト送信終了の dsmHandle」を渡します。

dsmEndSendObjExOut_t *dsmEndSendObjExOutP (O)

以下のオブジェクト送信終了情報を渡す。

名前	説明
totalBytesSent	アプリケーションから読み取った合計バイト数

名前	説明
objCompressed	オブジェクトが圧縮されたかどうかを示すフラグ
totalCompressedSize	圧縮後の合計バイト・サイズ
totalLFBytesSent	送信された LAN フリーの合計バイト数
objDeduplicated	API によってオブジェクトが重複排除されたかどうかを示すフラグ
totalDedupSize	重複排除後に送信された合計バイト数

戻りコード

戻りコード番号を括弧 () 内に示してあります。

表 29. *dsmEndSendObjEx* の戻りコード

戻りコード	説明
DSM_RC_NO_MEMORY (102)	この要求を完了するための RAM が残っていない。

dsmEndTxn

dsmEndTxn 関数呼び出しは、IBM Spectrum Protect トランザクションを終了します。

dsmEndTxn 関数呼び出しと **dsmBeginTxn** を 1 対にすることによって、1 つのトランザクション と見なされる呼び出しまたは呼び出しのセットが識別されます。

アプリケーション・クライアント は **dsmEndTxn** 呼び出しで、トランザクションのコミットと終了のどちらを実行する必要があるかを指定できます。

次の呼び出しのすべてを、トランザクションの範囲内で実行する必要があります。

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

構文

```
dsInt16_t dsmEndTxn
(dsUInt32_t dsmHandle,
 dsUInt8_t vote,
 dsUInt16_t *reason);
```

パラメーター

dsUInt32_t dsmHandle (I)

この呼び出しを直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

dsUInt8_t vote (I)

アプリケーション・クライアントが、前回の **dsmBeginTxn** 呼び出しから今回の呼び出しまでの間に行われたすべてのアクションをコミットするかどうかを示します。以下の値を指定できます。

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


DSM_VOTE_ABORT は、アプリケーションでトランザクションを停止する理由が検出された場合にのみ使用します。

dsUint16_t *reason (0)

dsmEndTxn の呼び出しがエラーで終了するか、**vote** の値が一致しない場合、このパラメーターには **vote** が失敗した理由を示す理由コードが入ります。この呼び出しの戻りコードがゼロで、理由コードが非ゼロの場合があります。したがってアプリケーション・クライアントは、戻りコードとその理由の両方について (**if (rc || reason)**) を使い必ずエラーの検査をしないと、正常終了を想定することはできません。

アプリケーションが **DSM_VOTE_ABORT** の **vote** を指定すると、理由コードは **DSM_RS_ABORT_BY_CLIENT** (3) になります。考えられる理由コードのリストについては、[145 ページの『付録 A API 戻りコード・ソース・ファイル: dsmrc.h』](#) を参照してください。戻りコードのリストの 1 から 50 は、理由コード用に予約されています。サーバーがトランザクションを終了した場合、戻りコードは **DSM_RC_CHECK_REASON_CODE** になります。この場合、理由値は、打ち切りの原因についてさらに多くの情報を含んでいます。

戻りコード

戻りコード番号を括弧 () 内に示してあります。

表 30. *dsmEndTxn* の戻りコード

戻りコード	説明
DSM_RC_ABORT_CRC_FAILED (236)	サーバーから受信した CRC が、クライアントの計算による CRC と一致しない。
DSM_RC_INVALID_VOTE (2011)	vote に指定された値が無効。
DSM_RC_CHECK_REASON_CODE (2302)	トランザクションが打ち切られた。理由フィールドをチェックしてください。
DSM_RC_ABORT_STGPOOL_COPY_CONT_NO (241)	いずれかのコピー・ストレージ・プールへの書き込みが失敗し、しかも、IBM Spectrum Protect のストレージ・プール・オプション COPYCONTINUE が NO に設定されている。トランザクションは終了する。
DSM_RC_ABORT_RETRY_SINGLE_TXN (242)	<p>この打ち切りコードは、保管操作時の問題のために現行トランザクションが打ち切られたことを示します。この問題は、個別トランザクション内の各ファイルを送信することによって解決する場合があります。このエラーは、以下の環境での典型的なエラーです。</p> <ul style="list-style-type: none">• 次のストレージ・プールに異なるコピー・ストレージ・プール・リストがある。• トランザクションの最中に、操作がこのプールに切り替えられた。•

dsmEndTxnEx

dsmEndTxnEx 関数呼び出しは、**dsmGroupHandler** 関数呼び出しとともに使用する グループ・リーダー・オブジェクト ID 情報を提供します。これは **dsmEndTxn** 関数 呼び出しの拡張です。

構文

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

パラメーター

dsmEndTxnExIn_t *dsmEndTxnExInP (I)

この構造には、以下のパラメーターが含まれます。

dsmHandle

セッションを識別し、そのセッションを後続の IBM Spectrum Protect 呼び出しに関連付ける ハンドル。

dsUInt8_t vote (I)

アプリケーション・クライアントが、直前の **dsmBeginTxn** 呼び出しとこの呼び出しとの間に行われたすべてのアクションをコミットするかどうか を示します。指定できる値は、次のとおりです。

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

アプリケーションでトランザクションを 停止する理由を検出した場合にのみ、DSM_VOTE_ABORT を使用してください。

dsmEndTxnExOut_t *dsmEndTxnExOutP (O)

この構造には、以下のパラメーターが含まれます。

dsUInt16_t *reason (O)

dsmEndTxnEx への呼び出しがエラーで終了するか、*vote* の値が一致しない場合、このパラメーターには *vote* が失敗した理由を示す理由コードが入ります。

ヒント：この呼び出しの戻りコードがゼロで、理由コードが非ゼロの場合があります。したがってアプリケーション・クライアントは、戻りコードとその理由の両方について (**if (rc || reason)**) を使い必ずエラーの検査をしないと、正常終了を想定することはできません。

アプリケーションが DSM_VOTE_ABORT の *vote* を指定すると、理由コードは DSM_RS_ABORT_BY_CLIENT (3) になります。考えられる理由コードのリストについては、[145 ページの『付録 A API 戻りコード・ソース・ファイル: dsmrc.h』](#)を参照してください。戻りコードのリストの 1 から 50 は、理由コード用に予約されています。サーバーが トランザクションを終了した場合、戻りコードは DSM_RC_CHECK_REASON_CODE になります。この場合、理由値は、打ち切りの原因についてさらに 多くの情報を含んでいます。

groupLeaderObjId

dsmGroupHandler 呼び出しで DSM_ACTION_OPEN フラグが使用される場合に返されるグループ・リーダー・オブジェクト ID。

戻りコード

戻りコード番号を括弧 () 内に示してあります。

表 31. dsmEndTxnEx の戻りコード

戻りコード	説明
DSM_RC_INVALID_VOTE (2011)	<i>vote</i> に指定された値が無効。

表 31. dsmEndTxnEx の戻りコード (続き)

戻りコード	説明
DSM_RC_CHECK_REASON_CODE (2302)	トランザクションが打ち切られた。理由フィールドをチェックしてください。
DSM_RC_ABORT_STGPOOL_COPY_CONT_NO (241)	いずれかのコピー・ストレージ・プールへの書き込みが失敗し、しかも、IBM Spectrum Protect のストレージ・プール・オプション COPYCONTINUE が NO に設定されている。トランザクションは終了する。
DSM_RC_ABORT_RETRY_SINGLE_TXN (242)	同時書き込み操作中に、トランザクションに含まれるオブジェクトが、目的のコピー・ストレージ・プールと異なるコピー・ストレージ・プールを持つ宛先に送られました。現在のトランザクションを終了し、各オブジェクトをそれぞれ別個のトランザクションで再送してください。

dsmGetData

dsmGetData 関数呼び出しは、IBM Spectrum Protect からバイト・ストリームのデータを取得し、それを呼び出し元のバッファに入れます。直前の **dsmGetObj** または **dsmGetData** 呼び出しからさらに受信するデータがある場合アプリケーション・クライアントは **dsmGetData** を呼び出します。

構文

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

パラメーター

dsUint32_t dsmHandle (I)

この呼び出しを直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

DataBlk *dataBlkPtr (I/O)

受信するデータ用のバッファを指すポインターと、そのバッファのサイズの 両方を含む構造を指します。戻りの際、この構造には、実際に転送されたバイト数が記録されます。タイプ定義については、[155 ページの『付録 B API タイプ定義ソース・ファイル』](#)を参照してください。

戻りコード

戻りコード番号を括弧 () 内に示してあります。

表 32. dsmGetData の戻りコード

戻りコード	説明
DSM_RC_ABORT_INVALID_OFFSET (33)	部分オブジェクト・リトリブ中に指定されたオフセットが、オブジェクトの長さより大きい。
DSM_RC_ABORT_INVALID_LENGTH (34)	部分オブジェクト・リトリブ中に指定された長さがオブジェクトの長さより 大きいか、またはオフセットにこの長さを加えると、オブジェクトの終わりを 超える。
DSM_RC_FINISHED (121)	処理は終了した。最後のバッファが受信された。データ量を知るために numBytes をチェックして、IBM Spectrum Protect dsmEndGetObj を呼び出す。

表 32. *dsmGetData* の戻りコード (続き)

戻りコード	説明
DSM_RC_NULL_DATA_BLKPTR (2001)	データ・ブロック・ポインターがヌル。
DSM_RC_ZERO_BUFLLEN (2008)	データ・ブロック・ポインターのバッファ長がゼロ。
DSM_RC_NULL_BUF_PTR (2009)	データ・ブロック・ポインターのバッファ・ポインターがヌル。
DSM_RC_WRONG_VERSION_PARM (2065)	アプリケーション・クライアントの API バージョンが IBM Spectrum Protect ライブラリーのバージョンと異なる。
DSM_RC_MORE_DATA (2200)	取得するデータがまだある。

dsmGetBufferData

dsmGetBufferData 関数呼び出しは、バッファを介して IBM Spectrum Protect からバイト・ストリームのデータを受け取ります。アプリケーションでこの関数呼び出しを使用するときは、使用するたびに、データのコピーと、**dsmReleaseBuffer** 呼び出しによるバッファの解放が必要になります。アプリケーションが保留しているバッファの数が **dsmInitEx** 呼び出しに指定されている numTsmBuffers の値と等しくなると、**dsmGetBufferData** 関数は、**dsmReleaseBuffer** が呼び出されるまでブロックされます。

構文

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

パラメーター

getDataExIn_t * dsmGetBufferDataExInP (I)

この構造には、以下の入力パラメーターが含まれます。

dsUInt32_t dsmHandle

セッションを識別し、直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

getDataExOut_t * dsmGetBufferDataExOutP (O)

この構造には、以下の出力パラメーターが含まれます。

dsUInt8_t tsmBufferHandle(O)

受け取ったバッファを識別するハンドル。

char *dataPtr(O)

データが書き込まれたアドレス。

dsUInt32_t numBytes(O)

実際に IBM Spectrum Protect によって書き込まれたバイト数。

戻りコード

戻りコード番号を括弧 () 内に示してあります。

表 33. *dsmGetBufferData* の戻りコード

戻りコード	説明
DSM_RC_BAD_CALL_SEQUENCE (2041)	呼び出しが適切な状態で発行されなかった。
DSM_RC_OBJ_ENCRYPTED (2049)	この関数は、暗号化されたオブジェクトには使用できない。
DSM_RC_OBJ_COMPRESSED (2048)	この関数は、圧縮されたオブジェクトには使用できない。

表 33. *dsmGetBufferData* の戻りコード (続き)

戻りコード	説明
DSM_RC_BUFF_ARRAY_ERROR (2045)	バッファ配列エラーが発生した。

dsmGetNextQObj

dsmGetNextQObj 関数呼び出しは、直前の **dsmBeginQuery** 呼び出しから次の照会応答を取得し、その応答を呼び出し元のバッファに入れます。

dsmGetNextQObj 呼び出しは 1 回以上呼び出されます。この関数が呼び出されるたびに、単一の照会レコードがリトリブされるか、エラーまたは DSM_RC_FINISHED の理由コードが返されます。

DSM_RC_FINISHED が返された場合、処理するデータはもうありません。すべての照会データがリトリブされた場合、またはそれ以上の照会データが必要ない場合には、**dsmEndQuery** 呼び出しを送信して照会プロセスを終了してください。

dataBlkPtr パラメーターは、**qryResp*Data** 構造タイプで定義されたバッファを指している必要があります。**dsmGetNextQObj** が呼び出されるコンテキストによって、照会応答で入力される構造のタイプが決まります。

構文

```
dsInt16_t dsmGetNextQObj (dsUInt32_t    dsmHandle,
                          DataBlk *dataBlkPtr);
```

パラメーター

dsUInt32_t dsmHandle (I)

この呼び出しを直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

DataBlk *dataBlkPtr (I/O)

受信するデータ用のバッファを指すポインターと、そのバッファのサイズの両方を含む構造を指します。このバッファは、**qryResp*Data** 応答構造です。戻りの際、この構造には転送されたバイト数が入っています。次の表で、各照会タイプに関連付けられている構造について説明します。

DataBlk のタイプ定義の詳細は、155 ページの『付録 B API タイプ定義ソース・ファイル』のトピックを参照してください。

表 34. *DataBlk* ポインター構造

照会	応答の構造	注目するフィールド
qtArchive	qryRespArchiveData	sizeEstimate 直前の dsmSendObj 呼び出しで渡された値が入っています。 mediaClass 値は、オブジェクトがディスク上にある場合は MEDIA_FIXED、オブジェクトが磁気テープ上にある場合は MEDIA_LIBRARY となります。 clientDeduplicated このオブジェクトがクライアントによって重複排除されたかどうかを示します。

表 34. DataBlk ポインター構造 (続き)

照会	応答の構造	注目するフィールド
qtBackup	qryRespBackupData	restoreOrderExt タイプは dsUInt16_t です。 dsmBeginGetData 呼び出しで複数のオブジェクトをリストアップするときは、このフィールドでソートします。この呼び出しのためのソートのコーディング例は、API サンプル dapiqry.c にあります。ソート例については、 66 ページの図 16 を参照してください。
		sizeEstimate 直前の dsmSendObj 呼び出しで渡された値が入っています。
		mediaClass 値は、オブジェクトがディスク上にある場合は MEDIA_FIXED、オブジェクトが磁気テープ上にある場合は MEDIA_LIBRARY となります。
		clientDeduplicated このオブジェクトがクライアントによって重複排除されたかどうかを示します。
qtBackupActive	qryARespBackupData	
qtBackupGroups	qryRespBackupData	dsBool_t isGroupLeader 真の場合、このオブジェクトはグループ・リーダーです。
qtOpenGroups	qryRespBackupData	dsBool_t isOpenGroup; 真の場合、このグループはオープン状態であり、未完了です。

表 34. DataBlk ポインター構造 (続き)

照会	応答の構造	注目するフィールド
qtFilespace	qryRespFSData	backStartDate ファイル・スペースが backStartDate アクションで更新されたときのサーバーのタイム・スタンプが入っています。 backCompleteDate ファイル・スペースが backCompleteDate アクションで更新されたときのサーバーのタイム・スタンプが入っています。 lastReplStartDate サーバーで複製が開始された最終時刻のタイム・スタンプが入っています。 lastReplCmpltDate 複製が完了した (失敗した場合も含む) 最終時刻のタイム・スタンプが入っています。 lastBackOpDateFromServer サーバーに保管された最後の保管タイム・スタンプが入っています。 lastBackOpDateFromLocal クライアントに保管された最後の保管タイム・スタンプが入っています。
qtMC	qryRespMCData qryRespMCDetailData	
qtProxyNodeAuth	qryRespProxyNodeData targetNodeName peerNodeName hlAddress llAddress	
qtProxyNodePeer	qryRespProaxyNodeData targetNodeName peerNodeName hlAddress llAddress	

戻りコード

次の表は、**dsmGetNextQObj** 関数呼び出しの戻りコードの説明です。

表 35. **dsmGetNextQObj** 関数呼び出しの戻りコード

戻りコード	戻りコード番号	説明
DSM_RC_ABORT_NO_MATCH	2	一致するものがない照会が要求された。
DSM_RC_FINISHED	121	処理が終了している (dsmEndQuery を開始する)。処理するデータはもうありません。

表 35. **dsmGetNextQObj** 関数呼び出しの戻りコード (続き)

戻りコード	戻りコード番号	説明
DSM_RC_UNKNOWN_FORMAT	122	IBM Spectrum Protect がリストアまたはリトリーブを試みたファイルのフォーマットが明らかでない。
DSM_RC_COMM_PROTOCOL_ERROR	136	通信プロトコル・エラー。
DSM_RC_NULL_DATABLKPTR	2001	ポインターがデータ・ブロックを指していない。
DSM_RC_INVALID_MCNAME	2025	管理クラス名が無効。
DSM_RC_BAD_CALL_SEQUENCE	2041	呼び出しの順序が無効。
DSM_RC_WRONG_VERSION_PARM	2065	アプリケーション・クライアントの API バージョンが IBM Spectrum Protect ライブラリー・バージョン と異なる。
DSM_RC_MORE_DATA	2200	取得するデータがまだある。
DSM_RC_BUFF_TOO_SMALL	2210	バッファが小さすぎる。

dsmGetObj

dsmGetObj 関数呼び出しは、IBM Spectrum Protect データ・ストリームから要求されたオブジェクト・データを取得し、それを呼び出し側のバッファに入れます。**dsmGetObj** 呼び出しは、データ・ストリームから次のオブジェクトまたは部分オブジェクトを取得するために、オブジェクト ID を使用します。

指示されたオブジェクトについてのデータは、**DataBlk** が指すバッファに入れます。さらに多くのデータが使用可能であれば、**dsmGetData** へ 1 回または複数回呼び出しを行い、DSM_RC_FINISHED の戻りコードが戻されるまで残りのオブジェクト・データを受信する必要があります。**DataBlk** の numBytes フィールドを検査して、バッファにデータが残っているかどうかを調べてください。

オブジェクトは、**dsmBeginGetData** 呼び出しで **dsmGetList** パラメーターに示された順序で要求する必要があります。例外は、アプリケーション・クライアント がリストの後ろの方のオブジェクトに到達するために、データ・ストリーム内のオブジェクトを省略したい場合です。オブジェクト ID で示されるオブジェクトがストリームの次のオブジェクトではない場合には、オブジェクトが検出されるか、ストリームが空になるまで、データ・ストリームが処理されます。この機能は気を付けて使用してください。これは、要求されたオブジェクトを検出するために、大量のデータを処理し、廃棄しなければならない場合があるからです。

要件: **dsmGetObj** が失敗コード (NOT FINISHED または MORE_DATA) を戻した場合は、セッションを終了してリストア操作を停止する必要があります。これは、暗号化を使用している場合、および RC_ENC_WRONG_KEY を受け取った場合には特に重要です。この場合、正しい鍵を使用して新しいセッションを開始する必要があります。

構文

```
dsInt16_t dsmGetObj
(dsUInt32_t dsmHandle,
 ObjID     *objIdP,
 DataBlk   *dataBlkPtr);
```

パラメーター

dsUInt32_t dsmHandle (I)

この呼び出しを直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

ObjID *objIdP (I)

リストアすべきオブジェクトの ID を指すポインター。

DataBlk *dataBlkPtr (I/O)

リストアされるデータが入っているバッファを指すポインター。

戻りコード

戻りコード番号を括弧 () 内に示してあります。

表 36. *dsmGetObj* の戻りコード

戻りコード	説明
DSM_RC_ABORT_INVALID_OFFSET (33)	部分オブジェクト・リトリブ中に指定されたオフセットが、オブジェクトの長さより大きい。
DSM_RC_ABORT_INVALID_LENGTH (34)	部分オブジェクト・リトリブ中に指定された長さがオブジェクトの長さより大きいか、またはオフセットにこの長さを加えると、オブジェクトの終わりを越える。
DSM_RC_FINISHED (121)	処理が終了している (dsmEndGetObj を開始する)。
DSM_RC_WRONG_VERSION_PARM (2065)	アプリケーション・クライアントの API バージョンが IBM Spectrum Protect ライブラリーのバージョンと異なる。
DSM_RC_MORE_DATA (2200)	取得するデータがまだある。
RC_ENC_WRONG_KEY (4580)	dsmInitEx 呼び出しに指定されている鍵または保管されている鍵が、このオブジェクトを暗号化するとき使用された鍵と一致しない。セッションを終了して、正しい鍵を指定する必要がある。

dsmGroupHandler

dsmGroupHandler 関数呼び出しは、与えられた入力に従って、論理ファイル・グループについてアクションを実行します。クライアントは、多数の個別オブジェクトを関連付け、IBM Spectrum Protect サーバーで 1 つの論理グループとして参照し、管理できるようにします。

詳しくは、[61 ページの『ファイルのグループ化』](#)を参照してください。

構文

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

パラメーター

dsmGroupHandlerIn_t *dsmGroupHandlerInP (I)

グループ属性を API に渡します。

groupType

グループのタイプ。値は、次のとおりです。

- DSM_GROUPTYPE_PEER - 対等グループ

actionType

実行するアクション。値は、次のとおりです。

- DSM_GROUP_ACTION_OPEN - 新規グループの作成
- DSM_GROUP_ACTION_CLOSE - オープン・グループのコミットと保管

- DSM_GROUP_ACTION_ADD - グループへの追加
- DSM_GROUP_ACTION_ASSIGNTO - 別のグループへの割り当て
- DSM_GROUP_ACTION_REMOVE - グループからのメンバーの除去

memberType.

オブジェクトのグループ・タイプ値は、次のとおりです。

- DSM_MEMBERTYPE_LEADER - グループ・リーダー
- DSM_MEMBERTYPE_MEMBER - グループ・メンバー

*uniqueGroupTagP

グループに関連付けられる固有のストリング ID。

leaderObjId

グループ・リーダーのオブジェクト ID。

*objNameP

グループ・リーダーのオブジェクト名を指すポインター。

memberObjList

除去または割り当てを行うオブジェクトのリスト。

dsmGroupHandlerOut_t *dsmGroupHandlerOutP (0)

API が完了させる構造のアドレスを渡します。構造のバージョン番号が戻されます。

戻りコード

戻りコード番号を括弧 () 内に示してあります。

表 37. dsmGroupHandler の戻りコード

戻りコード	説明
DSM_RC_ABORT_INVALID_GROUP_ACTION (237)	無効な操作がグループ・リーダーまたはメンバーに対して試みられた。

dsmInit

dsmInit 関数呼び出しは、API セッションを開始し、クライアントを IBM Spectrum Protect ストレージに接続します。アプリケーション・クライアントは、一度に 1 つ のみ活動セッションをオープンできます。異なるパラメーターを指定して別のセッションをオープンする場合は、まず **dsmTerminate** 呼び出しを使用して、現行セッションを終了する必要があります。

ノード間の照会、リストア、リトリブを許可するには、*-fromnode* および *-fromowner* ストリング・オプションを使用します。詳しくは、[25 ページの『ノード間および所有者間のオブジェクトのアクセス』](#)を参照してください。

構文

```
dsInt16_t dsmInit (dsUInt32_t      *dsmHandle,
                  dsmApiVersion *dsmApiVersionP,
                  char           *clientNodeNameP,
                  char           *clientOwnerNameP,
                  char           *clientPasswordP,
                  char           *applicationType,
                  char           *configfile,
                  char           *options);
```

パラメーター

dsUInt32_t *dsmHandle (O)

この初期設定セッションを識別し、それを後続の IBM Spectrum Protect 呼び出しに関連付けるハンドル。

dsmApiVersion *dsmApiVersionP (I)

アプリケーション・クライアントがこのセッションに使用している API のバージョンを識別するデータ構造を指すポインター。この構造には、`dsmapi.h` ファイル内に設定された 3 つの定数 (`DSM_API_VERSION`、`DSM_API_RELEASE`、および `DSM_API_LEVEL`) の値が入っています。ユーザーのワークステーションにインストールされている API ライブラリーのバージョンと、アプリケーション・クライアント API のバージョン との間の互換性を保証するために、事前に **dsmQueryApiVersion** への呼び出しを行っておく 必要があります。

char *clientNodeNameP (I)

このパラメーターは、IBM Spectrum Protect セッションのためのノードを指すポインター です。すべてのセッションには、それぞれ、ノード名が関連付けられていなければなりません。ファイル `dsmapi.h` 内の定数 `DSM_MAX_NODE_LENGTH` によって、ノード名に許可される最大サイズが設定されます。

ノード名では、大文字小文字が区別されません。

このパラメーターが両方とも NULL に設定されており、しかも *passwordaccess* が *prompt* に設定されている場合、API は、まず渡された オプション・ストリングからノード名を取得しようとします。ノード名がオプション・ストリングにない場合は、API は構成ファイルまたはオプション・ファイルからノード名を取得しようとします。ノード名検出のためのこれらの試みが失敗した場合、UNIX または Linux の API はシステム・ホスト名を使用しますが、その他のオペレーティング・システムの API は、コード `DSM_RC_REJECT_ID_UNKNOWN` を返します。

このパラメーターが NULL に設定されており、`dsm.opt` ファイルの (Windows API の場合) または `dsm.sys` ファイル (UNIX または Linux API の場合) の *passwordaccess* オプションが *generate* に設定されている場合、API は *nodename* オプション値またはシステム・ホスト名を使用します。

char *clientOwnerNameP (I)

このパラメーターは、IBM Spectrum Protect セッションの所有者を指すポインターです。セッションを開始するオペレーティング・システムが複数ユーザー・オペレーティング・システムの場合、NULL の所有者名 (root ユーザー) には、オブジェクトの所有者に関係なく、アプリケーションに所属するオブジェクトをバックアップ、アーカイブ、リストアまたはリトリブする権限があります。

所有者名では、大文字小文字が区別されます。

このパラメーターが NULL に設定されており、`dsm.sys` ファイルの *passwordaccess* オプションが *generate* に設定されている場合は API がログイン・ユーザー ID を使用します。

注: マルチユーザー・オペレーティング・システムでは、所有者名がアプリケーションを実行しているセッションの活動ユーザー ID と一致している必要はありません。

char *clientPasswordP (I)

このパラメーターは、IBM Spectrum Protect セッションが実行されるノードのパスワードを 指すポインターです。`dsmapi.h` ファイル内の定数 `DSM_MAX_VERIFIER_LENGTH` によって、このパスワードに許可される最大サイズが 設定されます。

パスワードでは大文字小文字が区別されません。

passwordaccess が *generate* に設定されている場合、パスワード・ファイルを最初に開始するときを除いて、このパラメーターの値は無視されます。

char *applicationType (I)

このパラメーターは、セッションを実行しているアプリケーションを識別します。アプリケーション・クライアントが値を定義します。

API アプリケーション・クライアントがサーバーとのセッションを開始するたびに、クライアントのアプリケーション・タイプ (またはプラットフォーム) がサーバー上で更新されます。アプリケーション・タイプ値は、サーバーの **platform** (プラットフォーム) フィールドに入力されるので、オペレー

ティング・システムの省略語を組み込むことをお勧めします。ストリングの最大長は DSM_MAX_PLATFORM_LENGTH です。

アプリケーション・タイプの現行値を調べるときは、**dsmQuerySessInfo** を呼び出してください。

char *configfile (I)

このパラメーターは、API 構成ファイルの完全修飾名を含む文字ストリングを指します。API 構成ファイルに指定したオプションによって、クライアント・オプション・ファイルの指定が指定変更されます。オプション・ファイルは、IBM Spectrum Protect (クライアントまたは API) のインストール時に定義されます。

char *options (I)

次のユーザー・オプションを含むことができる文字ストリングを指します。

- *Compressalways*
- *Servername* (UNIX または Linux のみ)
- *TCPServeraddr*
- *Fromnode*
- *Fromowner*
- *EnableClientEncryptKey*
- *Passwordaccess*

アプリケーション・クライアントは、オプション・リストを使用して、構成ファイルによって設定されたこれらのオプションの値を指定変更できます。

オプションの形式は、次のようになります。

1. オプション・リストに指定される各オプションはダッシュ (-) で始まりその後に直接オプション・キーワードが続きます。
2. このキーワードの後に、直接等号 (=) が続き、その後にオプション・パラメーターが続きます。
3. オプション・パラメーターにブランク・スペースがある場合は、このパラメーターを単一引用符または二重引用符で囲む必要があります。
4. 複数のオプションを指定する場合は、それらのオプションの間にブランクを入れます。

options が NULL の場合、すべてのオプションの値はユーザー・オプション・ファイルまたは API 構成ファイルから取られます。

戻りコード

戻りコード番号を括弧 () 内に示してあります。

表 38. dsmInit の戻りコード

戻りコード	説明
DSM_RC_ABORT_SYSTEM_ERROR (1)	サーバーはシステム・エラーを検出し、クライアントに通知した。
DSM_RC_REJECT_VERIFIER_EXPIRED (52)	パスワードが期限切れになっており、更新の必要がある。
DSM_RC_REJECT_ID_UNKNOWN (53)	ノード名が見つからなかった。
DSM_RC_AUTH_FAILURE (137)	認証できなかった。
DSM_RC_NO_STARTING_DELIMITER (148)	パターンに開始区切り文字がない。

表 38. *dsmInit* の戻りコード (続き)

戻りコード	説明
DSM_RC_NEEDED_DIR_DELIMITER (149)	『突き合わせディレクトリー』メタストリング(『...』)の直前と直後にディレクトリー区切り文字が必要だが、それがない。
DSM_RC_NO_PASS_FILE (168)	パスワード・ファイルは使用できません。
DSM_RC_UNMATCHED_QUOTE (177)	オプション・ストリングにアンマッチの引用符がある。
DSM_RC_NLS_CANT_OPEN_TXT (0610)	メッセージ・テキスト・ファイルをオープンできない。
DSM_RC_INVALID_OPT (400)	オプション・ストリング内の項目が無効。
DSM_RC_INVALID_DS_HANDLE (2014)	DSM ハンドルが無効。
DSM_RC_NO_OWNER_REQD (2032)	<i>passwordaccess</i> が <i>generate</i> の場合、所有者パラメーターは NULL でなければならない。
DSM_RC_NO_NODE_REQD (2033)	<i>passwordaccess</i> が <i>generate</i> の場合、ノード・パラメーターは NULL でなければならない。
DSM_RC_WRONG_VERSION (2064)	アプリケーション・クライアントの API バージョンが IBM Spectrum Protect バージョンより高い。
DSM_RC_PASSWD_TOOLONG (2103)	指定されたパスワードが長すぎる。
DSM_RC_NO_OPT_FILE (2220)	構成ファイルが見つけれなかった。
DSM_RC_INVALID_KEYWORD (2221)	オプション・ストリングに指定されたキーワードが無効。
DSM_RC_PATTERN_TOO_COMPLEX (2222)	包含/除外パターンが複雑すぎて、IBM Spectrum Protect で解釈できない。
DSM_RC_NO_CLOSING_BRACKET (2223)	パターンに右大括弧がない。
DSM_RC_INVALID_SERVER (2225)	複数ユーザー環境で、システム構成ファイル内にサーバーが見つからなかった。
DSM_RC_NO_HOST_ADDR (2226)	ホストに接続するための情報が不十分。
DSM_RC_MACHINE_SAME (2227)	オプション・ファイル内に定義するノード名は、システム・ホスト名と同じであってはならない。
DSM_RC_NO_API_CONFIGFILE (2228)	構成ファイルをオープンできない。
DSM_RC_NO_INCLEXCL_FILE (2229)	包含/除外ファイルが見つからなかった。
DSM_RC_NO_SYS_OR_INCLEXCL (2230)	<i>dsm.sys</i> ファイルまたは包含/除外ファイルが見つからなかった。

関連概念[クライアント・オプション・ファイルの概要](#)[処理オプション](#)

dsmInitEx

dsmInitEx 関数呼び出しは、拡張検査の追加のパラメーターを使用して API セッションを開始します。

構文

```
dsInt16_t dsmInitEx (dsUInt32_t *dsmHandleP,  
                    dsmInitExIn_t *dsmInitExInP,  
                    dsmInitExOut_t *dsmInitExOutP) ;
```

パラメーター

dsUInt32_t *dsmHandleP (O)

この初期設定セッションを識別し、それを後続の IBM Spectrum Protect 呼び出しに関連付けるハンドル。

dsmInitExIn_t *dsmInitExInP

この構造には、以下の入力パラメーターが含まれます。

dsmApiVersion *dsmApiVersionP (I)

このパラメーターは、アプリケーション・クライアントがこのセッションに 使用している API のバージョンを識別するデータ構造を指すポインターです。この 構造には、dsmapitd.h ファイル内に設定された 4 つの定数の値、DSM_API_VERSION、DSM_API_RELEASE、DSM_API_LEVEL、および DSM_API_SUBLEVEL が入っています。**.dsmQueryApiVersionEx** を呼び出し、アプリケーション・クライアントの API のバージョンとユーザーのワークステーションにインストール済みの API ライブラリーのバージョンとの間の互換性を確認します。

char *clientNodeNameP (I)

このパラメーターは、IBM Spectrum Protect セッションのためのノードを指すポインター です。すべてのセッションがノード名に関連付けられている必要があります。dsmapitd.h ファイル内の定数 DSM_MAX_NODE_LENGTH によって、ノード名の最大サイズが設定されます。

ノード名では、大文字小文字が区別されません。

このパラメーターが NULL の場合、および **passwordaccess** が prompt に設定されている場合は、API は、まず渡された オプション・ストリングからノード名を取得しようとします。ノード名がオプション・ストリングにない場合は、API は構成ファイルまたはオプション・ファイルからノード名を取得しようとします。ノード名検出のためのこれらの試みが失敗した場合、UNIX または Linux の API はシステム・ホスト名を使用しますが、その他のオペレーティング・システムの API は DSM_RC_REJECT_ID_UNKNOWN を返します。

このパラメーターが NULL に設定されており、dsm.opt ファイルの (Windows API の場合) または dsm.sys ファイル (UNIX または Linux API の場合) の **passwordaccess** オプションが generate に設定されている場合、API は nodename オプション値またはシステム・ホスト名を使用します。

char *clientOwnerNameP (I)

このパラメーターは、IBM Spectrum Protect セッションの所有者を指すポインターです。オペレーティング・システムが複数ユーザー・プラットフォームの場合は、NULL の所有者名 (root ユーザー) には、オブジェクトの所有者に関係なく、アプリケーションに所属するオブジェクトをバックアップ、アーカイブ、リストア、またはリトリブする権限があります。

所有者名では、大文字小文字が区別されます。

このパラメーターが NULL に設定されており、dsm.sys ファイルの **passwordaccess** オプションが generate に設定されている場合は API がログイン・ユーザー ID を使用します。

注: マルチユーザー・プラットフォームでは、所有者名がアプリケーションを実行しているセッションの活動ユーザー ID と一致している必要はありません。

char *clientPasswordP (I)

IBM Spectrum Protect セッションが実行されるノードのパスワードを指すポインター。
dsmapiitd.h ファイル内の定数 DSM_MAX_VERIFIER_LENGTH によって、このパスワードに許可される最大サイズが設定されます。

パスワードでは、大文字小文字が区別されません。

passwordaccess が **generate** に設定されている場合、パスワード・ファイルを最初に開始する
ときを除いて、このパラメーターの値は無視されます。

char *userNameP;

このノードに関するクライアント権限をもつ管理ユーザー名へのポインター。

char *userPasswordP;

値が提供されている場合、**userName** パラメーターのパスワードへのポインター。

char *applicationType (I)

IBM Spectrum Protect セッションを実行しているアプリケーションを識別します。アプリケーション・クライアントがこの値を識別します。

API アプリケーション・クライアントがサーバーとのセッションを開始するたびに、クライアントのアプリケーション・タイプ (またはオペレーティング・システム) が、サーバー上で更新されます。値は、サーバーの **platform** フィールドに入力されます。値にオペレーティング・システム ID をを使用することを検討してください。ストリングの最大長は DSM_MAX_PLATFORM_LENGTH 定数で定義されます。

アプリケーション・タイプの現行値を表示するには、**dsmQuerySessInfo** を呼び出してください。

char *configfile (I)

API 構成ファイルの完全修飾名を含む文字ストリングを指します。API 構成ファイルに指定したオプションによって、クライアント・オプション・ファイルの指定が指定変更されます。オプション・ファイルは、IBM Spectrum Protect (クライアントまたは API) のインストール時に定義されます。

char *options (I)

次のユーザー・オプションを含むことができる文字ストリングを指します。

- Compressalways
- Servername (UNIX および Linux システムのみ)
- TCPServeraddr (UNIX システム以外)
- Fromnode
- Fromowner
- Passwordaccess

アプリケーション・クライアントは、オプション・リストを使用して、構成ファイルによって設定されたこれらのオプションの値を指定変更できます。

オプションのフォーマットは、次のとおりです。

1. オプション・リストに指定される各オプションはダッシュ (-) で始まりその後に直接オプション・キーワードが続きます。
2. このキーワードの後に、直接等号 (=) が続き、その後にオプション・パラメーターが続きます。
3. オプション・パラメーターにブランク・スペースがある場合は、このパラメーターを単一引用符または二重引用符で囲む必要があります。
4. 複数のオプションを指定する場合は、それらのオプションの間にブランクを入れます。

options が NULL の場合、すべてのオプションの値はユーザー・オプション・ファイルまたは API 構成ファイルから取られます。

dirDelimiter

ファイル・スペース、高位名または低位名に 接頭語として付けられるディレクトリー区切り文字。
dirDelimiter パラメーターは、アプリケーションがシステム・デフォルト値を指定変更する場合のみ、指定する必要があります。UNIX または Linux の環境では、デフォルトはスラッシュ (/) です。Windows 環境では、デフォルトは逆スラッシュ (¥) です。

useUnicode

ユニコードが使用可能であるかどうかを示すブール・フラグ。UNIX システムと Windows システムの間でクロスプラットフォーム・インターオペラビリティを得るには、**useUnicode** フラグを **false** に設定する必要があります。

bCrossPlatform

UNIX システムと Windows システムの間でクロスプラットフォーム・インターオペラビリティを得るために (**bTrue** に) 設定する必要があるブール・フラグ。**bCrossPlatform** フラグが設定されている場合、API は、ファイル・スペースがユニコードではないこと、およびアプリケーションがユニコードを使用していないことを確認します。ユニコードを使用する Windows アプリケーションには、非ユニコードのエンコードを使用するアプリケーションとの互換性はありません。ユニコードを使用する Windows アプリケーションに対して **bCrossPlatform** フラグを設定してはなりません。

UseTsmBuffers

バッファ・コピー除去を使用するかどうかを示します。

numTsmBuffers

useTsmBuffers=bTrue の場合のバッファ数。

bEncryptKeyEnabled

アプリケーション管理鍵による暗号化が使用されるかどうかを示します。

encryptionPasswordP

暗号化パスワード。

制約事項: **encryptkey=save** の場合、暗号鍵を存在していると、**encryptionPasswordP** に指定されている値は無視されます。

dsmAppVersion *appVersionP (I)

このパラメーターは、API セッションを開始しているアプリケーションのバージョン情報を識別するデータ構造を指すポインターです。構造には 4 つの定数の値、**applicationVersion**、**applicationRelease**、**applicationLevel**、および **applicationSubLevel** が含まれています。これらは、**tsmapitd.h** ファイルに設定されています。

dsmInitExOut_t *dsmInitExOut P

この構造には出力パラメーターが含まれます。

dsUint32_t *dsmHandle (O)

この初期設定セッションを識別し、それを後続の API 呼び出しに関連付けるハンドル。

infoRC

戻りコードに関する追加情報。関数の戻りコードと **infoRC** の値の両方を確認してください。

infoRC の値が **DSM_RC_REJECT_LASTSESS_CANCELED** (69) である場合、IBM Spectrum Protect は、管理者が最終セッションを取り消したことを示します。

戻りコード

戻りコード番号を括弧 () 内に示してあります。

表 39. **dsmInitEx** の戻りコード

戻りコード	説明
DSM_RC_ABORT_SYSTEM_ERROR (1)	IBM Spectrum Protect サーバーは、システム・エラーを検出し、クライアントに通知した。

表 39. **dsmInitEx** の戻りコード (続き)

戻りコード	説明
DSM_RC_REJECT_VERIFIER_EXPIRE D (52)	パスワードが期限切れになっており、更新の必要がある。次の呼び出しは、この呼び出しで戻されたハンドルを指定した dsmChangePW とする必要がある。
DSM_RC_REJECT_ID_UNKNOWN (53)	ノード名が見つからない。
DSM_RC_TA_COMM_DOWN (103)	通信リンクがダウンしている。
DSM_RC_AUTH_FAILURE (137)	認証できなかった。
DSM_RC_NO_STARTING_DELIMITER (148)	パターンに開始区切り文字がない。
DSM_RC_NEEDED_DIR_DELIMITER (149)	"match directories" メタストリング ("...") の直前と直後にディレクトリー区切り文字が必要だが、それが無い。
DSM_RC_NO_PASS_FILE (168)	パスワード・ファイルは使用できません。
DSM_RC_UNMATCHED_QUOTE (177)	オプション・ストリングに一致していない引用符がある。
DSM_RC_NLS_CANT_OPEN_TXT (0610)	メッセージ・テキスト・ファイルをオープンできない。
DSM_RC_INVALID_OPT (2013)	オプション・ストリング内の項目が無効。
DSM_RC_INVALID_DS_HANDLE (2014)	DSM ハンドルが無効。
DSM_RC_NO_OWNER_REQD (2032)	passwordaccess が generate の場合、所有者パラメーターは NULL でなければならない。
DSM_RC_NO_NODE_REQD (2033)	passwordaccess が generate に設定されている場合、ノード・パラメーターは NULL でなければならない。
DSM_RC_WRONG_VERSION (2064)	アプリケーション・クライアントの API バージョンが IBM Spectrum Protect バージョンより高い。
DSM_RC_PASSWD_TOOLONG (2103)	指定されたパスワードが長すぎる。
DSM_RC_NO_OPT_FILE (2220)	構成ファイルが見つからない。
DSM_RC_INVALID_KEYWORD (2221)	オプション・ストリングに指定されたキーワードが無効。
DSM_RC_PATTERN_TOO_COMPLEX (2222)	包含/除外パターンが、IBM Spectrum Protect で解釈するには複雑すぎる。
DSM_RC_NO_CLOSING_BRACKET (2223)	パターンに右大括弧がない。
DSM_RC_INVALID_SERVER (2225)	複数ユーザー環境で、システム構成ファイル内にサーバーが見つからなかった。
DSM_RC_NO_HOST_ADDR (2226)	ホストに接続するための情報が不十分。
DSM_RC_MACHINE_SAME (2227)	オプション・ファイル内に定義するノード名は、システム・ホスト名と同じであってはならない。
DSM_RC_NO_API_CONFIGFILE (2228)	構成ファイルをオープンできない。
DSM_RC_NO_INCLEXCL_FILE (2229)	包含/除外ファイルが見つからなかった。
DSM_RC_NO_SYS_OR_INCLEXCL (2230)	dsm.sys ファイルまたは包含/除外ファイルが見つからなかった。

関連概念

[クライアント・オプション・ファイルの概要](#)

[処理オプション](#)

dsmLogEvent

dsmLogEvent 関数呼び出しは、ユーザー・メッセージ (ANE4991 I) をサーバー・ログ・ファイル、ローカル・エラー・ログ、またはその両方に記録します。タイプ **logInfo** の構造がこの呼び出しに渡されます。この呼び出しは、セッション内で **InSession** 状態にあるときに実行する必要があります。Send、Get、または query の状態にあるときには実行しないでください。サーバー上に記録されたメッセージを検索するには、管理可能クライアントから **query actlog** コマンドを使用します。

状態遷移の要約、[74 ページの図 20](#) を参照してください。

構文

```
dsInt16_t dsmLogEvent
(dsUInt32_t dsmHandle,
 logInfo *logInfoP);
```

パラメーター

dsUInt32_t dsmHandle(I)

この呼び出しを直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

logInfo *logInfoP (I)

メッセージと宛先を渡します。アプリケーション・クライアントには、この構造用のストレージを割り振る責任があります。

logInfo 構造内のフィールドは、次のとおりです。

message

記録するメッセージのテキスト。これは、ヌル文字で終了するストリングでなければなりません。最大長は DSM_MAX_RC_MSG_LENGTH です。

dsmLogtype

メッセージのログ先を指定する。指定可能な値は、**logServer**、**logLocal**、**logBoth** です。

戻りコード

戻りコード番号を括弧 () 内に示してあります。

表 40. dsmLogEvent の戻りコード

戻りコード	説明
DSM_RC_STRING_TOO_LONG (2120)	メッセージ・ストリングが長すぎる。

dsmLogEventEx

dsmLogEventEx 関数呼び出しは、ユーザー・メッセージをサーバー・ログ・ファイル、ローカル・エラー・ログ、またはその両方に記録します。この呼び出しは、セッション内で **InSession** 状態にあるときに実行する必要があります。この呼び出しを送信、取得、または照会の呼び出しの中で呼び出すことはできません。

状態遷移図の要約: セッションの相互作用の概要については、次のトピックで状態遷移図の要約を参照してください。

[74 ページの図 20](#)

重大度によって IBM Spectrum Protect メッセージ番号が決まります。サーバーに記録されているメッセージを表示するには、管理可能クライアントから **query actlog** コマンドを使用します。アプリケーションが、クライアント・ログ dsmLogType (logLocal または logBoth のいずれか) に書き込まれる非常に多数のクライアント・メッセージを生成する場合は、IBM Spectrum Protect クライアント・オプション errorlogretention を使用して、クライアント・エラー・ログ・ファイルを除去してください。詳しくは、IBM Spectrum Protect サーバー資料を参照してください。

構文

```
extern dsInt16_t DSMLINKAGE dsmLogEventEx(  
    dsUInt32_t dsmHandle,  
    dsmLogExIn_t *dsmLogExInP,  
    dsmLogExOut_t *dsmLogExOutP  
);
```

パラメーター

dsUInt32_t dsmHandle(I)

この呼び出しを直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

dsmLogExIn_t *dsmLogExInP

この構造には入力パラメーターが含まれます。

dsmLogSeverity severity;

このパラメーターはイベントの重大度です。指定できる値は、次のとおりです。

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

char appMsgID[8];

このパラメーターは、特定のアプリケーション・メッセージを識別するための文字列です。適切な形式は、4桁の数字の前に3文字を指定したものです。例: DSM0250。

dsmLogType logType;

このパラメーターは、イベントの送信先を指定します。このパラメーターには、以下の値を指定できます。

- logServer
- logLocal
- logBoth

char *message;

このパラメーターは、記録するイベント・メッセージのテキストです。このテキストは、ヌルで終了する文字列であることが必要です。最大長は DSM_MAX_RC_MSG_LENGTH です。

制約事項: サーバーに送られるメッセージは英語であることが必要です。英語以外のメッセージは正しく表示されません。

dsmLogExOut_t *dsmLogExOutP

この構造には出力パラメーターが含まれます。現在は、出力パラメーターはありません。

戻りコード

戻りコード番号を括弧 () 内に示してあります。

表 41. dsmLogEventEx の戻りコード

戻りコード	説明
DSM_RC_STRING_TOO_LONG (2120)	メッセージ・ストリングが長すぎる。

dsmQueryAccess

dsmQueryAccess 関数呼び出しは、オブジェクトの バックアップ・バージョンまたはアーカイブ・コピーのためのすべてのアクセス許可 規則を、サーバーに照会します。アクセス規則の配列を指すポインターが呼び出し に渡されて、完全な配列が戻されます。配列の中に規則がいくつあるかを 示すために、規則の数を指すポインターが渡されます。

この呼び出しに特定の戻りコードはありません。

構文

```
dsInt16_t DSMLINKAGE dsmQueryAccess
                (dsUInt32_t      dsmHandle),
                qryRespAccessData **accessListP,
                dsUInt16_t      *numberOfRules) ;
```

パラメーター

dsUInt32_t dsmHandle (I)

この呼び出しを直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

qryRespAccessData **accessListP (O)

API ライブラリーが割り振る qryRespAccessData エレメント の配列を指すポインター。各エレメントが、1つのアクセス規則に対応します。配列内のエレメントの 数が、**numberOfRules** パラメーターに戻されます。各 qryRespAccessData エレメントで 戻される情報には、次のものがあります。

名前	説明
ruleNumber	アクセス規則の ID。これは、削除すべき規則を示すために使用できます。
AccessType	バックアップまたはアーカイブのタイプ。
Node	アクセスを許可したノード。
Owner	アクセスを許可したユーザー。
objName	高位または低位のファイル・スペース記述子。

dsUInt32_t *numberOfRules (O)

accessList 配列内の規則の数を戻します。

dsmQueryApiVersion

dsmQueryApiVersion 関数呼び出しは、アプリケーション・クライアントによってアクセス される API ライブラリー・バージョンについて照会要求を実行します。

API に対する更新はすべて、上方互換性形式で行われます。エンド・ユーザーの ワークステーション上の API ライブラリー以前の API バージョンまたはリリースを もつアプリケーション・クライアントはすべて、変更なく作動します。**dsmQueryApiVersion** 呼び出しが、アプリケーション・クライアントのバージョンまたはバージョンとリリースよりも 古いバージョンまたはバージョンとリリースを戻す場合は、先に進む前に、一部の API 呼び出しが、エンド・ユーザーのより古いバージョンの API ではサポートされていない方法で 拡張されている可能性があることに注意してください。

アプリケーションの API バージョン番号は、定数 DSM_API_VERSION、DSM_API_RELEASE、および DSM_API_LEVEL として dsmapi.h ヘッダー・ファイルに保管されます。

この呼び出しに特定の戻りコードはありません。

構文

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

パラメーター

dsmApiVersion *apiVersionP (O)

このパラメーターは、API ライブラリーのバージョン、リリース、およびレベルのコンポーネントを含む構造を指すポインターです。例えば、ライブラリーがバージョン 1.1.0 の場合は、この呼び出しからの戻り後、構造の各フィールドには、次の値が入ります。

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

dsmQueryApiVersionEx

dsmQueryApiVersionEx 関数呼び出しは、アプリケーション・クライアントによってアクセス される API ライブラリー・バージョンについて照会要求を実行します。

API に対する更新はすべて、上方互換性形式で行われます。エンド・ユーザーのワークステーション上の API ライブラリー以前の API バージョンまたはリリースをもつアプリケーション・クライアントはすべて、変更なく作動します。上位互換性の例外については、README_api_enu ファイルの中の「コード変更の要約 (Summary of Code Changes)」を参照してください。**dsmQueryApiVersionEx** 呼び出しが、アプリケーション・クライアントと異なるバージョンとリリースを戻す場合は、先に進む前に一部の API 呼び出しが、エンド・ユーザーのより古いバージョンの API ではサポートされていない方法で拡張されている可能性があることに注意してください。

アプリケーションの API バージョン番号は、定数 DSM_API_VERSION、DSM_API_RELEASE、DSM_API_LEVEL、および DSM_API_SUBLEVEL として dsmapi.h ヘッダー・ファイルに保管されます。

この呼び出しに特定の戻りコードはありません。

構文

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

パラメーター

dsmApiVersionEx *apiVersionP (O)

このパラメーターは、API ライブラリーのバージョン、リリース、レベル、およびサブレベルのコンポーネントを含む構造を指すポインターです。例えば、ライブラリーがバージョン 5.5.0.0 の場合は、この呼び出しからの戻り後、構造の各フィールドには、次の値が入ります。

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

dsmQueryCliOptions

dsmQueryCliOptions 関数呼び出しは、ユーザーのオプション・ファイル内の重要なオプションの値を照会します。この呼び出しでは、タイプが構造の **optStruct** が渡され、これに情報が入ります。この呼び出しは、**dsmInitEx** 呼び出しの前に実行され、セッション前のセットアップを決定します。

この呼び出しに特定の戻りコードはありません。

構文

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

パラメーター

optStruct *optstructP (I/O)

このパラメーターは、API が埋める構造のアドレスを渡します。アプリケーション・クライアントには、この構造用のストレージを割り振る責任があります。正常に戻れば、この構造の各フィールドには適切な情報が入っています。

以下の情報が **optStruct** 構造に返されます。

名前	説明
dsmiDir	環境変数 DSMI_DIR の値。
dsmiConfig	環境変数 DSMI_CONFIG によって指定される クライアント・オプション・ファイル。
serverName	IBM Spectrum Protect サーバーの名前。
commMethod	選択した通信方式。dsmapitd.h ファイルの中の DSM_COMM_* のための #define を参照してください。
serverAddress	通信方式に基づくサーバーのアドレス。
nodeName	クライアント・ノード (マシン) 名。
compression	このフィールドは、圧縮 (compression) オプションに関する情報を提供します。
passwordAccess	値は、generate の場合は <i>bTrue</i> 、prompt の場合は <i>bFalse</i> です。

関連概念

[処理オプション](#)

dsmQuerySessInfo

dsmQuerySessInfo 関数呼び出しは、**dsmHandle** に指定したセッションの 操作に関連する情報を求める IBM Spectrum Protect への照会要求を開始します。この呼び出しでタイプ **ApiSessInfo** の構造が、入力されたすべての使用可能な セッション関連情報とともに渡されます。この呼び出しは正常な **dsmInitEx** 呼び出しの後で開始されます。

ApiSessInfo 構造に返される情報には、次のものがあります。

- サーバー情報: ポート番号、日付と時刻、およびタイプ。
- クライアント・デフォルト: アプリケーション・タイプ、削除許可、区切り文字、およびトランザクション限界。
- セッション情報: ログイン ID および所有者。
- ポリシー・データ: ドメイン、アクティブ・ポリシー・セット、および保存猶予期間。

渡される構造およびその中の各フィールドの内容についての詳細は、[155 ページの『付録 B API タイプ定義ソース・ファイル』](#)を参照してください。

構文

```
dsInt16_t dsmQuerySessInfo (dsUInt32_t dsmHandle,  
    ApiSessInfo *SessInfoP);
```

パラメーター

dsUInt32_t dsmHandle (I)

この呼び出しを直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

ApiSessInfo *SessInfoP (I/O)

このパラメーターは、API が入力する構造のアドレスを渡します。アプリケーション・クライアントは、構造用のストレージを割り振り、使用する構造のバージョンを示すフィールド項目を埋める責任があります。正常に戻れば、この構造の各フィールドには適切な情報が入っています。

adsmServerName は、IBM Spectrum Protect サーバー上で **define server** コマンドによって与えられた名前です。archiveRetentionProtection フィールドが true の場合、サーバーは保存保護を使用可能です。

戻りコード

戻りコード番号を括弧 () 内に示してあります。

表 42. *dsmQuerySessInfo* の戻りコード

戻りコード	説明
DSM_RC_NO_SESS_BLK (2006)	サーバーのセッション・ブロック情報がない。
DSM_RC_NO_POLICY_BLK (2007)	使用できるサーバー・ポリシー情報がない。
DSM_RC_WRONG_VERSION_PARM (2065)	アプリケーション・クライアントの API バージョンが IBM Spectrum Protect ライブラリーのバージョンと異なる。

dsmQuerySessOptions

dsmQuerySessOptions 関数呼び出しは、**dsmHandle** の指定したセッション内での有効な重要オプション値を照会します。この呼び出しでは、タイプが構造の **optStruct** が渡され、これに情報が入ります。

この呼び出しは、正常な **dsmInitEx** 呼び出しの後で開始されます。返される値は、**dsmInitEx** 呼び出しに渡される値 (主として **optString** および **optFile**) によっては、**dsmQueryCliOptions** 呼び出しで返される値とは異なる可能性があります。オプションの優先順位については、1 ページの『構成ファイルとオプション・ファイルの理解』を参照してください。

この呼び出しに特定の戻りコードはありません。

構文

```
dsInt16_t dsmQuerySessOptions  
    (dsUInt32_t dsmHandle,  
    optStruct *optstructP);
```

パラメーター

dsUInt32_t dsmhandle(I)

この呼び出しを直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

optStruct *optstructP (I/O)

このパラメーターは、API が埋める構造のアドレスを渡します。アプリケーション・クライアントには、この構造用のストレージを割り振る責任があります。正常に戻れば、この構造の各フィールドには適切な情報が入っています。

optStruct 構造に返される情報は、次のとおりです。

名前	説明
dsmiDir	DSMI_DIR 環境変数の値。
dsmiConfig	DSMI_CONFIG 環境変数が指定する dsm.opt ファイル。
serverName	オプション・ファイルの IBM Spectrum Protect サーバー・スタンプの名前。
commMethod	選択した通信方式。dsmapitd.h ファイルの中の DSM_COMM_* のための #define を参照してください。
serverAddress	通信方式に基づくサーバーのアドレス。
nodeName	クライアント・ノード (マシン) の名前。
compression	圧縮オプションの値 (bTrue=on および bFalse=off)。
compressAlways	compressalways オプションの値 (bTrue=on および bFalse=off)。
passwordAccess	値は、generate の場合は bTrue、prompt の場合は bFalse です。

関連概念

[処理オプション](#)

dsmRCMsg

dsmRCMsg 関数呼び出しは、API 戻りコードに関連したメッセージ・テキストを取得します。

重要: **dsmRCMsg** が **dsmInitEx** の失敗後に呼びだされた場合、Servername オプションは無視され、エラーはログ・ファイルに出力されます。これは、デフォルトでサーバーに定義されています。

msg パラメーターは、括弧で囲まれたメッセージ接頭語戻りコードと、その後にメッセージ・テキストを表示します。例えば、**dsmRCMsg** の呼び出しにより、次のメッセージが戻されます。

```
ANS0264E (RC2300) Only root user can execute dsmChangePW
or dsmDeleteFS
.
(dsmChangePW
または dsmDeleteFS
を実行できるのは、root ユーザーのみです。)
```

ANSII コード・ページと OEM コード・ページで文字が異なる一部の言語では、印刷する前にストリングを ANSI から OEM に変換することが必要な場合があります (例えば、東ヨーロッパの 1 バイト文字セット)。次に一例を示します。

```
dsmRCMsg(dsmHandle, rc, msgBuf);
#ifdef WIN32
#ifdef WIN64
CharToOemBuff(msgBuf, msgBuf, strlen(msgBuf));
#endif
#endif
printf("
```

構文

```
dsInt16_t dsmRCMsg (dsUInt32_t dsmHandle,  
    dsInt16_t dsmRC,  
    char *msg);
```

パラメーター

dsUInt32_t dsmHandle (I)

この呼び出しを直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

dsInt16_t dsmRC (I)

関連メッセージ・テキストの API 戻りコード。API 戻りコードは `dsmrc.h` ファイルで定義されています。詳しくは、[145 ページの『付録 A API 戻りコード・ソース・ファイル: dsmrc.h』](#)を参照してください。

char *msg (O)

このパラメーターは、戻りコード **dsmRC** に関連付けられたメッセージ・テキストです。呼び出し側は、メッセージ・テキスト用に十分なスペースを割り振る責任があります。

msg の最大長は `DSM_MAX_RC_MSG_LENGTH` として定義されます。

各国語サポートがあり言語メッセージ・ファイルを選択できるプラットフォーム では、API からは各国語でのメッセージ・ストリングが戻されます。

戻りコード

戻りコード番号を括弧 () 内に示してあります。

表 43. `dsmRCMsg` の戻りコード

戻りコード	説明
DSM_RC_NULL_MSG (2002)	<code>dsmRCMsg</code> 呼び出しの msg パラメーターが NULL ポインターである。
DSM_RC_INVALID_RETCODE (2021)	dsmRCMsg 呼び出しに渡された戻りコードが無効である。
DSM_RC_NLS_CANT_OPEN_TXT (0610)	メッセージ・テキスト・ファイルをオープンできない。

dsmRegisterFS

dsmRegisterFS 関数呼び出しは、IBM Spectrum Protect サーバーに新規ファイル・スペースを登録します。最初にファイル・スペースを登録してからでないと、そこヘデータをバックアップすることはできません。

アプリケーション・クライアントは、バックアップ/アーカイブ・クライアントが使用するのと同じファイル・スペース名を使用してはなりません。

- UNIX または Linux では、これらの名前を調べるために **df** コマンドを実行します。
- Windows では、これらの名前は通常、システム上の別のドライブに関連付けられたボリューム・ラベルです。

構文

```
dsInt16_t dsmRegisterFS  
    (dsUInt32_t dsmHandle,  
    regFSData *regFilespaceP);
```

パラメーター

dsUInt32_t dsmHandle (I)

この呼び出しを直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

regFSDData *regFilespaceP (I)

このパラメーターは、IBM Spectrum Protect サーバーに登録すべきファイル・スペースの 名前および関連した情報を渡します。

ヒント: *fstype* フィールドには接頭部の「**API:**」が含まれます。ファイル・スペースの照会のすべてに、このストリングが表示されます。例えば、ユーザーが **dsmRegisterFS** で *fstype* として *myfstype* を渡した場合、サーバー上の実際の値ストリングは、照会時に **API:myfstype** として返されます。この接頭部は、API オブジェクトをバックアップ/アーカイブ・オブジェクトと 区別するためのものです。

fsInfo での使用可能域は、DSM_MAX_USER_FSINFO_LENGTH となりました。

戻りコード

戻りコード番号を括弧 () 内に示してあります。

表 44. *dsmRegisterFS* の戻りコード

戻りコード	説明
DSM_RC_INVALID_FSNAME (2016)	ファイル・スペース名が無効。
DSM_RC_INVALID_DRIVE_CHAR (2026)	ドライブ名が英字でない。
DSM_RC_NULL_FSNAME (2027)	ファイル・スペース名がヌル。
DSM_RC_FS_ALREADY_REGED (2062)	ファイル・スペースが既に登録済み。
DSM_RC_WRONG_VERSION_PARM (2065)	アプリケーション・クライアントの API バージョンが IBM Spectrum Protect ライブラリーのバージョンと異なる。
DSM_RC_FSINFO_TOOLONG (2106)	ファイル・スペース情報が長すぎる。

dsmReleaseBuffer

dsmReleaseBuffer 関数は、IBM Spectrum Protect にバッファを返します。アプリケーションが **dsmReleaseBuffer** を呼び出すのは、**dsmGetDataEx** を呼び出して、バッファ内のすべてのデータの移動を完了してバッファを解放できるようになったときです。**dsmReleaseBuffer** を呼び出すには、あらかじめ **dsmInitEx** を (*UseTsmBuffers* には *btrue* を指定し、*numTsmBuffers* にはゼロ以外の値を指定して) 呼び出しておく必要があります。また、**dsmReleaseBuffer** は、**dsmTerminate** を呼び出す直前になってもアプリケーションが保留しているデータ・バッファが残っている場合にも、呼び出す必要があります。

dsmReleaseBuffer の構文

```
dsInt16_t dsmReleaseBuffer (releaseBufferIn_t *dsmReleaseBufferInP,  
                             releaseBufferOut_t *dsmReleaseBufferOutP) ;
```

パラメーター

releaseBufferIn_t * dsmReleaseBufferInP (I)

この構造には、以下の入力パラメーターが含まれます。

dsUInt32_t dsmHandle (I)

この呼び出しを直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

dsUInt8_t tsmBufferHandle(I)

バッファを識別するハンドル。

char *dataPtr(I)

アプリケーションが書き込まれたアドレス。

戻りコード

戻りコード番号を括弧 () 内に示してあります。

表 45. *dsmReleaseBuffer* の戻りコード

戻りコード	説明
DSM_RC_BAD_CALL_SEQUENCE	呼び出しが適切な状態で発行されなかった。
DSM_RC_INVALID_TSMBUFFER	ハンドル、または dataPtr の値が無効。
DSM_RC_BUFF_ARRAY_ERROR	バッファ配列エラーが発生した。

dsmRenameObj

dsmRenameObj 関数呼び出しは、高位オブジェクト名または低位オブジェクト名を名前変更します。バックアップ・オブジェクトの場合は、現行のオブジェクト名を渡し、高位オブジェクト名または低位オブジェクト名のいずれか用に変更します。アーカイブ・オブジェクトの場合は、現行のオブジェクト・ファイル・スペース名およびオブジェクト ID を渡し、高位オブジェクト名または低位オブジェクト名のいずれか用に変更します。この関数呼び出しは、**dsmBeginTxn** 呼び出しおよび **dsmEndTxn** 呼び出し内で使用してください。

重複するバックアップ・オブジェクト名が既存のバックアップとマージされるかどうかを 判別するために、マージ・フラグが使用されます。新しい名前が既存のオブジェクト に対応し、マージが真の場合、現行のオブジェクトは新しい名前に変換され、そのオブジェクトが新しい名前の活動バージョンになります。一方、その名前を持つ 既存の活動オブジェクトは、そのオブジェクトの 最高位の非活動コピーになります。新しい名前が既存のオブジェクトに対応し、マージが偽の場合、関数は、戻りコード DSM_RC_ABORT_DUPLICATE_OBJECT を戻します。

制約事項：

- オブジェクトの所有者のみがそのオブジェクトの名前変更ができます。
- **dsmRenameObj** 関数は、データ保存保護が IBM Spectrum Protect サーバーで有効な場合、または IBM Spectrum Protect for Data Retention サーバーに接続している場合はサポートされません。

dsmRenameObj 関数呼び出しは、次の 3 つのマージの条件を テストします。

- 現行の **dsmObjName** オブジェクトと新規高位オブジェクト または新規低位オブジェクトが、所有者、コピー・グループ、および管理クラスに関して一致する必要がある。
- 現行の **dsmObjName** は、新しい名前 の現行活動オブジェクトよりも後で、バックアップされている必要がある。
- 現行の **dsmObjName** の活動コピーのみが存在し、非活動コピーは存在しない必要がある。

構文

```
dsInt16_t dsmRenameObj (dsmRenameIn_t      *dsmRenameInP,  
                        dsmRenameOut_t     *dsmRenameOutP);
```

パラメーター

dsUInt32_t dsmHandle (I)

この呼び出しを直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

dsmRenameIn_t *dsmRenameInP

この構造には入力パラメーターが含まれます。

dsUInt8_t repository (I);

このパラメーターは、削除するファイル・スペースがバックアップ・リポジトリ内にあるかアーカイブ・リポジトリ内にあるかを示しています。

dsmObjName *objNameP (I);

このパラメーターは、現行のファイル・スペース名、高位オブジェクト名、低位オブジェクト名、およびオブジェクト・タイプを含む構造を指すポインターです。

char newHl [DSM_MAX_HL_LENGTH + 1];

このパラメーターは新規の高位名を指定します。

char newLl [DSM_MAX_LL_LENGTH + 1];

このパラメーターは新規の低位名を指定します。

dsBool_t merge;

このパラメーターは、バックアップ・オブジェクトが重複する名前のオブジェクトと組み合わせられるかどうかを決めます。値は、真または偽です。

ObjID;

アーカイブ・オブジェクトのオブジェクト ID。

dsmRenameOut_t *dsmRnameOutP

この構造には出力パラメーターが含まれます。

注：出力パラメーターはありません。

戻りコード

戻りコード番号を括弧 () 内に示してあります。

表 46. *dsmRenameObj* の戻りコード

戻りコード	説明
DSM_RC_ABORT_MERGE_ERROR (45)	サーバーがマージ・エラーを見つけた。
DSM_RC_ABORT_DUPLICATE_OBJECT (32)	オブジェクトが既に存在し、マージは偽。
DSM_RC_ABORT_NO_MATCH (2)	オブジェクトが見つからない。
DSM_RC_REJECT_SERVER_DOWNLEVEL (58)	この関数を使用するには、IBM Spectrum Protect サーバーのレベルが V3.7.4.0 以降でなければならない。

dsmRequestBuffer

dsmRequestBuffer 関数は、IBM Spectrum Protect にバッファーを返します。**dsmRequestBuffer** は、アプリケーションから **dsmGetDataEx** を呼び出した後、バッファー内のすべてのデータの移動を完了してバッファーを解放できるようになったときに呼び出します。

dsmReleaseBuffer を呼び出すには、あらかじめ **dsmInitEx** を (*UseTsmBuffers* には *btrue* を指定し、*numTsmBuffers* にはゼロ以外の値を指定して) 呼び出しておく必要があります。また、**dsmReleaseBuffer** は、**dsmTerminate** を呼び出す直前になってもアプリケーションが保留している IBM Spectrum Protect バッファーが残っている場合にも、呼び出す必要があります。

構文

```
dsInt16_t dsmRequestBuffer (getBufferIn_t *dsmRequestBufferInP,  
                             getBufferOut_t *dsmRequestBufferOutP) ;
```

パラメーター

getBufferIn_t * dsmRequestBufferInP (I)

この構造には、以下の入力パラメーターが含まれます。

dsUInt32_t dsmHandle

セッションを識別し、直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

getBufferOut_t *dsmRequestBufferOut P (O)

この構造には出力パラメーターが含まれます。

dsUInt8_t tsmBufferHandle(O)

バッファーを識別するハンドル。

char *dataPtr(O)

アプリケーションが書き込まれるアドレス。

dsUInt32_t *bufferLen(O)

このバッファーに書き込めるバイト数の最大値。

戻りコード

戻りコード番号を括弧 () 内に示してあります。

表 47. *dsmRequestBuffer* の戻りコード

戻りコード	説明
DSM_RC_BAD_CALL_SEQUENCE (33)	呼び出しが適切な状態で発行されなかった。
DSM_RC_SENDDATA_WITH_ZERO_SIZE (34)	送信されるオブジェクトの長さが 0 の場合、 dsmReleaseBuffer の呼び出しは許可されない。
DSM_RC_BUFF_ARRAY_ERROR (121)	有効なバッファーを取得できなかった。

dsmRetentionEvent

dsmRetentionEvent 関数呼び出しは、オブジェクト ID のリストを、これらのオブジェクトに対して実行される保存イベント操作を指定して、IBM Spectrum Protect サーバーに送信します。この関数呼び出しは、**dsmBeginTxn** 呼び出しおよび **dsmEndTxn** 呼び出し内で使用してください。

注：この関数を使用するには、サーバーがバージョン 5.2.2.0 以降でなければなりません。

1 つの呼び出し内のオブジェクトの最大数は、**dsmQuerySessInfo** 呼び出しから *ApisessInfo* 構造内に返された *maxObjPerTxn* の値までに制限されています。

オブジェクト上のイベントを送信できるのは、そのオブジェクトの所有者のみです。

次のイベントが可能です。

eventRetentionActivate

イベント・ベースの管理クラスにバインドされているオブジェクトに対してのみ出すことができます。このイベントを送信すると、このオブジェクトに対するイベントが活動化され、このオブジェクトの保存の状態は DSM_ARCH_RETINIT_PENDING から DSM_ARCH_RETINIT_STARTED に変更されます。

eventHoldObj

このイベントは、オブジェクトに対して保存または削除保留を出し、保留解除が出されるまでオブジェクトが有効期限切れになったり削除されたりしないようにします。

eventReleaseObj

このイベントは、**objectHeld** フィールドの値が DSM_ARCH_HELD_TRUE であるオブジェクトに対してのみ出され、元の保存ポリシーを再開するオブジェクトの保留を除去します。

dsmRetentionEvent を送信する前に、オブジェクトに関する情報を取得するために 33 ページの『IBM Spectrum Protect システムの照会』で説明した照会シーケンスを送信します。**dsmGetNextQObj** の呼び出しは、アーカイブ照会の場合は **qryRespArchiveData** という名前のデータ構造を戻します。このデータ構造には、**dsmRetentionEvent** に必要な情報が含まれています。

構文

```
extern dsInt16_t DSMLINKAGE dsmRetentionEvent(  
    dsmRetentionEventIn_t      *ddsmRetentionEventInP,  
    dsmRetentionEventOut_t     *dsmRetentionEventOutP  
);
```

パラメーター

dsmRetentionEventIn_t *dsmRetentionEventP

この構造には、以下の入力パラメーターが含まれます。

dsUInt16_t stVersion;

このパラメーターは構造のバージョンを示します。

dsUInt32_t dsmHandle (I)

この呼び出しを直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

dsmEventType_t eventType (I);

このパラメーターはイベント・タイプを示します。指定可能な値 **eventRetentionActivate**、**eventHoldObj**、**eventReleaseObj** の意味については、このセクションの初めの説明を参照してください。

dsmObjList_t objList;

このパラメーターは、信号を送るオブジェクト ID のリストを示します。

戻りコード

戻りコード番号を括弧 () 内に示してあります。

表 48. **dsmRetentionEvent** の戻りコード

戻りコード	説明
DSM_RC_ABORT_NODE_NOT_AUTHORIZED (36)	ノードまたはユーザーに適正な権限がない。
DSM_RC_ABORT_TXN_LIMIT_EXCEEDED (249)	トランザクション内のオブジェクト数が多すぎる。
DSM_RC_ABORT_OBJECT_ALREADY_HELD (250)	オブジェクトは既に保留にされているため、さらに保留を出すことはできない。
DSM_RC_REJECT_SERVER_DOWNLEVEL (58)	この関数を使用するには、サーバーが V5.2.2.0 以降でなければなりません。

dsmSendBufferData

dsmSendBufferData 関数呼び出しは、直前の **dsmReleaseBuffer** 呼び出しの結果提供されたバッファを介して、バイト・ストリームのデータを IBM Spectrum Protect に送信します。アプリケーション・クライアントは、サーバーのストレージ用に、任意のタイプのデータを渡すことができます。通常、このデ

ータはファイル・データですが、ファイル・データ のみに限定されているわけではありません。送信するバイト・ストリームのデータが大容量である場合は、**dsmSendBufferData** を複数回呼び出すことができます。呼び出しが成功したか失敗したかに関係なく、バッファは 解放されます。

制約事項: *useTsmBuffers* オプションを使用している場合、圧縮 対象となっているオブジェクトがあっても、そのオブジェクトは圧縮されません。

構文

```
dsInt16_t dsmSendBufferData (sendBufferDataIn_t sendBufferDataOut_t, *dsmSendBufferDataExInP, *dsmSendBufferDataOutP);
```

パラメーター

sendBufferDataIn_t * dsmSendBufferDataInP (I)

この構造には、以下の入力パラメーターが含まれます。

dsUInt32_t dsmHandle (I)

この呼び出しを直前の **dsmInitEx** 呼び出しに 関連付けるハンドル。

dsUInt8_t tsmBufferHandle(I)

送信するバッファを識別するハンドル。

char *dataPtr(I)

アプリケーション・データが書き込まれたアドレス。

dsUInt32_t numBytes(I)

実際にアプリケーションによって書き込まれたバイト数 (**dsmReleaseBuffer** に指定された値よりも常に小さくなければなりません)。

戻りコード

戻りコード番号を括弧 () 内に示してあります。

表 49. *dsmSendBufferData* の戻りコード

戻りコード	説明
DSM_RC_BAD_CALL_SEQUENCE (2041)	呼び出しが適切な状態で発行されなかった。
DSM_RC_INVALID_TSMBUFFER (2042)	ハンドル、または dataPtr の値が無効。
DSM_RC_BUFF_ARRAY_ERROR (2045)	バッファ配列エラーが発生した。
DSM_RC_TOO_MANY_BYTES (2043)	numBytes の値が、 dsmReleaseBuffer 呼び出しに指定されたバッファのサイズよりも大きい。

dsmSendData

dsmSendData 関数呼び出しは、データのバイト・ストリームを、バッファを介して IBM Spectrum Protect に送信します。アプリケーション・クライアントは、サーバーのストレージ用に、任意のタイプのデータを渡すことができます。通常、これらのデータはファイル・データですが、それのみに限定されているわけではありません。**dsmSendData** は、送信する必要があるデータのバイト・ストリームが大きい場合、複数回呼び出すことができます。

制約事項: アプリケーション・クライアントは、**dsmSendData** 呼び出しから戻るまで、**dsmSendData** に指定したバッファを再利用できません。

ヒント: IBM Spectrum Protect がコード 157 (DSM_RC_WILL_ABORT) を戻す場合は、**dsmEndSendObj** の呼び出しを開始してから DSM_VOTE_COMMIT に vote を指定して、**dsmEndTxn** の呼び出しを開始してください。次にアプリケーションは戻りコード 2302 (DSM_RC_CHECK_REASON_CODE) を受け取り、理由

コードをアプリケーション・ユーザーに戻します。これは、サーバーがトランザクションを終了する理由をユーザーに知らせます。

構文

```
dsInt16_t dsmSendData  
(dsUInt32_t dsmHandle,  
 DataBlk *dataBlkPtr);
```

パラメーター

dsUInt32_t dsmHandle (I)

この呼び出しを直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

DataBlk *dataBlkPtr (I/O)

このパラメーターは、データの送信元となるバッファーを指すポインターと、そのバッファーのサイズの両方を含む構造を指します。戻りの際、この構造には、実際に転送されたバイト数が記録されます。タイプ定義については、[155 ページの『付録 B API タイプ定義ソース・ファイル』](#)を参照してください。

戻りコード

戻りコード番号を括弧 () 内に示してあります。

表 50. *dsmSendData* の戻りコード

戻りコード	説明
DSM_RC_NO_COMPRESS_MEMORY (154)	データの圧縮または解凍を行うために使用できるメモリーが不足している。
DSM_RC_COMPRESS_GREW (155)	圧縮中に、圧縮されたデータのサイズが元のデータと比較して、大きくなった。
DSM_RC_WILL_ABORT (157)	不明の予期しないエラーが発生し、トランザクションが停止した。
DSM_RC_WRONG_VERSION_PARM (2065)	アプリケーション・クライアントの API バージョンが IBM Spectrum Protect ライブラリーのバージョンと異なる。
DSM_RC_NEEDTO_ENDTXN (2070)	トランザクションを終了する必要がある。
DSM_RC_OBJ_EXCLUDED (2080)	包含/除外リストがオブジェクトを除外している。
DSM_RC_OBJ_NOBCG (2081)	オブジェクトがバックアップ・コピー・グループをもっておらず、サーバーに送信されない。
DSM_RC_OBJ_NOACG (2082)	オブジェクトにはアーカイブ・コピー・グループがなく、しかもオブジェクトはサーバーに送信されない。
DSM_RC_SENDDATA_WITH_ZERO_SIZE (2107)	オブジェクトは、 <i>sizeEstimate</i> がゼロ・バイトのデータを送信できない。

dsmSendObj

dsmSendObj 関数呼び出しは、ストレージへの単一オブジェクトの送信要求を開始します。性能上の理由から、複数の **dsmSendObj** 呼び出しおよび関連した **dsmSendData** 呼び出しを あるトランザクションの範囲内で行うことができます。

dsmSendObj 呼び出しは、オブジェクトのデータを、メモリー・バッファに渡されるバイト・ストリームとして処理します。**dsmSendObj** 呼び出しの **dataBlkPtr** パラメーターを使用すれば、アプリケーション・クライアントは、次のいずれかを行うことができます。

- 単一の呼び出しで、オブジェクトのデータおよび属性を渡す (属性は **objAttrPtr** を介して渡される)。
- **dsmSendObj** 呼び出しによってオブジェクト・データの一部を指定し、1 つ以上の **dsmSendData** 呼び出しによってそのデータの残りの部分を指定する。

アプリケーション・クライアントが使用できる代案として、**dsmSendObj** 呼び出しによって属性のみを指定し、1 つ以上の **dsmSendData** 呼び出しによってオブジェクト・データを指定するという方法もあります。この方法の場合、**dsmSendObj** 呼び出しで、**dataBlkPtr** を NULL に設定してください。

ヒント : 特定のオブジェクト・タイプでは、バイト・ストリーム・データがオブジェクト・データに関連していない場合があります。例えば、拡張属性を持たないディレクトリー項目の場合です。

管理クラスをバックアップまたはアーカイブしたいオブジェクトに正しくバインドするには、**dsmSendObj** を呼び出す前に、前述の **dsmBindMC** 呼び出しを行う必要があります。API は、オブジェクトがサーバーに送信されるとき、適切な管理クラスをそのオブジェクトに関連付けることができるように、このバインドを保持します。**dsmSendObj** 呼び出しでディレクトリーのオブジェクト・タイプのためのデフォルト値 (DSM_OBJ_DIRECTORY) にバインドされた管理クラスを許可する場合は、そのデフォルト値はデフォルト管理クラスであってはなりません。代わりに、最大の保存時間をもつ管理クラスが使用されます。この保存時間をもつ管理クラスが複数存在する場合は、最初に検出されたものが使用されます。

dsmEndSendObj 呼び出しを使用して、ストレージに送信したすべてのオブジェクト・データをたどってください。サーバーに送信するオブジェクト・データがない場合か、またはすべてのデータが **dsmSendObj** 呼び出し内に含まれている場合は、**dsmEndSendObj** 呼び出しを開始してからでなければ、別の **dsmSendObj** 呼び出しを開始することはできません。**dsmSendData** 呼び出しを通じて複数のデータの送信が要求された場合、**dsmEndSendObj** が最後の送信の後に続き、状態変更を示します。

ヒント : IBM Spectrum Protect がコード 157 (DSM_RC_WILL_ABORT) を戻す場合は、DSM_VOTE_COMMIT に vote を指定して **dsmEndTxn** の呼び出しを開始します。アプリケーションは戻りコード 2302 (DSM_RC_CHECK_REASON_CODE) を受け取り、理由コードをアプリケーション・ユーザーに戻します。これは、サーバーがトランザクションを終了する理由をユーザーに知らせます。

理由コードが 11 (DSM_RS_ABORT_NO_REPOSIT_SPACE) の場合は、*sizeEstimate* が実際のデータ量に対して小さ過ぎる可能性があります。アプリケーションは、より正確な *sizeEstimate* を決定して、そのデータを再度送信する必要があります。

構文

```
dsInt16_t dsmSendObj
(dsUint32_t dsmHandle,
 dsmSendType sendType,
 void *sendBuff,
 dsmObjName *objNameP,
 ObjAttr *objAttrPtr,
 DataBlk *dataBlkPtr);
```

パラメーター

dsUint32_t dsmHandle (I)

この呼び出しを直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

dsmSendType sendType (I)

このパラメーターは、実行される送信のタイプを指定します。指定できる値は、次のとおりです。

名前	説明
stBackup	サーバーに送信されるバックアップ・オブジェクト。
stArchive	サーバーに送信されるアーカイブ・オブジェクト。
stBackupMountWait	必要な装置 (テープなど) がマウントされるまでサーバーに 待機させた いバックアップ・オブジェクト。
stArchiveMountWait	必要な装置 (テープなど) がマウントされるまでサーバーに 待機させた いアーカイブ・オブジェクト。

注: アプリケーション・ユーザーがデータをテープに送信する可能性が少しでもある 場合は、**MountWait** タイプを使用してください。

void *sendBuff (I)

このパラメーターは、この呼び出しの **sendType** に固有のその他の情報を含む構造を指すポインターです。現在、**stArchive** の **sendType** のみに、構造が関連付けられています。この構造は **sndArchiveData** と呼ばれ、アーカイブ記述が含まれています。

dsmObjName *objNameP (I)

このパラメーターは、ファイル・スペース名、高位オブジェクト名、低位オブジェクト名、およびオブジェクト・タイプを含む構造を指すポインターです。詳しくは、[23 ページの『オブジェクト名とオブジェクト ID』](#)を参照してください。

ObjAttr *objAttrPtr (I)

このパラメーターは、関心のあるオブジェクト属性をアプリケーションに 渡します。タイプ定義については、[155 ページの『付録 B API タイプ定義ソース・ファイル』](#)を参照してください。

属性には、次のものがあります。

- **owner** は、オブジェクトの所有者を表します。所有者が特定の名前であることが 宣言されるか、または空ストリングであるかは、オブジェクトを IBM Spectrum Protect ストレージ から戻す際に重要です。詳しくは [24 ページの『セッション所有者としてのオブジェクトのアクセス』](#)を参照してください。
- **sizeEstimate** は、サーバーに 送信されるデータ・オブジェクトの合計サイズの最善の統計的予測値です。このサイズについては、可能な限り正確な見積もりを 出すよう努めてください。これは、サーバーがそのストレージ・リソース内でのスペース割り振りおよびオブジェクトの配置を効率よく 行うために、この属性を使用するからです。

指定したサイズ見積もりが、送信された実際のバイト数に比べて極端に 小さい場合には、サーバーが十分なスペースを割り振ることができなくなって、理由コード 11 (DSM_RS_ABORT_NO_REPOSIT_SPACE) でトランザクションを終了します。

注: このサイズ見積もりは、データ・オブジェクトの合計サイズについてバイト単位で行います。

DSM_MIN_COMPRESS_SIZE より小さいサイズを指定したオブジェクトは圧縮されません。

ユーザーのオブジェクトにビット・データがない (この呼び出しからの属性情報 のみ) 場合には、**sizeEstimate** をゼロにする必要があります。

注: バージョン 5.1.0 以降では、トランザクション内のコピー宛先は長さゼロのオブジェクトについては整合性検査が行われません。

- **objCompressed** は、オブジェクト・データが 既に圧縮されているかどうかを示すブール値です。

オブジェクトが圧縮されている場合 (object compressed=bTrue)、IBM Spectrum Protect が再度その圧縮を試みることはありません。圧縮されていない場合、IBM Spectrum Protect は、管理者によって設定され、API 構成ソース内で設定された compression オプションの値に基づいて、オブジェクトを圧縮するかどうかを決定します。

アプリケーションで部分オブジェクト・リストア/リトリブを計画している場合には、データの送信中にデータを圧縮することはできません。これを強制的に行うには、*ObjAttr.objCompressed* を *bTrue* に設定します。

- **objInfo** は、特定のオブジェクトについての情報を保管します。

制約事項: ここで、情報は自動的に保管されません。この属性を使用する場合は、*objInfo* の長さを示すために属性 *objInfoLength* を設定する必要があります。

- **mcNameP** には、**dsmBindMC** から取得した管理クラスを指定変更する管理クラスの名前が入っています。
- **disableDeduplication** はブール値です。この値が **true** に設定されると、このオブジェクトはクライアントによって重複排除されません。

DataBlk *dataBlkPtr (I/O)

このパラメーターは、バックアップまたはアーカイブされるデータの バッファーを指すポインターと、そのバッファーのサイズの両方を含む構造を 指します。このパラメーターは **dsmSendObj** にのみ適用されます。**dsmSendObj** 呼び出しではなく後続の **dsmSendData** 呼び出しでデータの送信を開始したい場合は、DataBlk 構造内のバッファー・ポインターを NULL に設定してください。戻りの際、この構造には、実際に転送されたバイト数が記録されます。タイプ定義については、[155 ページの『付録 B API タイプ定義ソース・ファイル』](#)を参照してください。

戻りコード

戻りコード番号を括弧 () 内に示してあります。

表 51. *dsmSendObj* の戻りコード

戻りコード	説明
DSM_RC_NO_COMPRESS_MEMORY (154)	データの圧縮または解凍を行うために使用できるメモリーが不足している。
DSM_RC_COMPRESS_GREW (155)	圧縮中に、圧縮されたデータのサイズが元のデータと比較して、大きくなった。
DSM_RC_WILL_ABORT (157)	不明の予期しないエラーが発生し、トランザクションが停止した。
DSM_RC_TL_NOACG (186)	このファイルの管理クラスが、送信タイプとして有効なコピー・グループをもっていない。
DSM_RC_NULL_OBJNAME (2000)	オブジェクト名がヌル。
DSM_RC_NULL_OBJATTRPTR (2004)	オブジェクト属性ポインターがヌル。
DSM_RC_INVALID_OBJTYPE (2010)	オブジェクト・タイプが無効。
DSM_RC_INVALID_OBJOWNER (2019)	オブジェクト所有者が無効。
DSM_RC_INVALID_SENDTYPE (2022)	送信タイプが無効。
DSM_RC_WILDCHAR_NOTALLOWED (2050)	ワイルドカード文字は許可されない。
DSM_RC_FS_NOT_REGISTERED (2061)	ファイル・スペースが登録されていない。
DSM_RC_WRONG_VERSION_PARM (2065)	アプリケーション・クライアントの API バージョンが IBM Spectrum Protect ライブラリーのバージョンと異なる。
DSM_RC_NEEDTO_ENDTXN (2070)	トランザクションを終了する必要がある。
DSM_RC_OBJ_EXCLUDED (2080)	包含/除外リストがオブジェクトを除外している。

表 51. *dsmSendObj* の戻りコード (続き)

戻りコード	説明
DSM_RC_OBJ_NOBCG (2081)	オブジェクトにはバックアップ・コピー・グループがなく、しかもオブジェクトはサーバーに送信されない。
DSM_RC_OBJ_NOACG (2082)	オブジェクトにはアーカイブ・コピー・グループがなく、しかもオブジェクトはサーバーに送信されない。
DSM_RC_DESC_TOOLONG (2100)	記述が長すぎる。
DSM_RC_OBJINFO_TOOLONG (2101)	オブジェクト情報が長すぎる。
DSM_RC_HL_TOOLONG (2102)	高位修飾子が長すぎる。
DSM_RC_FILESPACE_TOOLONG (2104)	ファイル・スペース名が長すぎる。
DSM_RC_LL_TOOLONG (2105)	低位修飾子が長すぎる。
DSM_RC_NEEDTO_CALL_BINDMC (2301)	dsmBindMC を最初に呼び出す必要がある。

dsmSetAccess

dsmSetAccess 関数呼び出しは、他のユーザーまたは他のノードに、オブジェクトのバックアップ・バージョンまたはアーカイブ・コピーへのアクセス、すべてのオブジェクトへのアクセス、または一部の選択したもののへのアクセスを許可します。別のユーザーにアクセスを許可すると、そのユーザーはそのファイルの照会、リストアまたはリトリートができるようになります。このコマンドは、*fs*、*hl*、*ll*、*node*、および *owner* のフィールドでワイルドカードをサポートします。

注: 1つのコマンドを使用して、バックアップ・バージョンとアーカイブ・コピーの両方へのアクセスを許可することはできません。backup または archive を指定しなければなりません。

構文

```
dsInt16_t DSMLINKAGE dsmSetAccess
(
    dsUInt32_t dsmHandle,
    dsmSetAccessType accessType,
    dsmObjName *objNameP,
    char *node,
    char *owner);
```

パラメーター

dsUInt32_t dsmHandle (I)

この呼び出しを直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

dsmAccessType accessType (I)

このパラメーターは、アクセスを許可したいオブジェクトのタイプを指定します。指定できる値は、次のとおりです。

名前	説明
atBackup	バックアップ・オブジェクトへのアクセスを設定することを指定します。
atArchive	アーカイブ・オブジェクトへのアクセスを設定することを指定します。

dsmObjName *objNameP (I)

このパラメーターは、ファイル・スペース名、高位オブジェクト名、および低位オブジェクト名を含む構造を指すポインターです。

注: すべてのファイル・スペースを指定するには、ファイル・スペース名にアスタリスク (*) を使用します。

char *node (I)

このパラメーターは、アクセスを許可するノード名を指すポインターです。すべてのノードの場合は、(*) を指定します。

char *owner (I)

このパラメーターは、アクセスを許可するユーザー名を指すポインターです。すべてのユーザーの場合は、(*) を指定します。

戻りコード

戻りコード番号を括弧 () 内に示してあります。

表 52. *dsmSetAccess* の戻りコード

戻りコード	説明
DSM_RC_INVALID_ACCESS_TYPE (2110)	無効なアクセス・タイプが指定された。
DSM_RC_FILE_SPACE_NOT_FOUND (124)	指定されたファイル・スペースがサーバー上で見つからない。
DSM_RC_QUERY_COMM_FAILURE (2111)	サーバーの照会中に通信エラーが発生した。
DSM_RC_NO_FILES_BACKUP (2112)	このファイル・スペースでバックアップがとられたファイルはない。
DSM_RC_NO_FILES_ARCHIVE (2113)	このファイル・スペースでアーカイブされたファイルはない。
DSM_RC_INVALID_SETACCESS (2114)	アクセス設定の記述が無効。

dsmSetUp

dsmSetUp 関数呼び出しは、環境変数値を指定変更します。**dsmSetUp** を呼び出してから、**dsmInitEx** を呼び出してください。**envSetUp** 構造で渡された値は、既存の環境変数またはデフォルトをすべて上書きします。フィールドに NULL を指定すると、値は環境から取られます。値を設定しないと、値はデフォルトから取られます。

要件:

1. **dsmSetUp** を使用する場合は、必ず **dsmTerminate** を呼び出してから **dsmCleanUp** を呼び出します。
2. API インストールメンテーションは、構成ファイル内に `testflag INSTRUMENT: API` が設定されており、アプリケーションで **dsmSetUp** 呼び出しまたは **dsmCleanUp** 呼び出しが使用されている場合にのみ、活性化されます。

構文

```
dsInt16_t DSMLINKAGE dsmSetUp
    (dsBool_t mtFlag,
     envSetUp *envSetUpP);
```

パラメーター

dsBool_t mtFlag (I)

このパラメーターは、API を単一スレッド・モードで使用するか、またはマルチスレッド・モードで使用するかを指定します。値は、次のとおりです。

```
DSM_SINGLETHREAD
DSM_MULTITHREAD
```

要件 : LAN フリー・データ転送を行うためには、マルチスレッド・フラグをオンにしておく必要があります。

envSetUp *envSetUpP(I)

このパラメーターは、指定変更値をもつ構造を指すポインターです。既存の環境変数を指定変更したくない場合は、NULL を指定します。**envSetUp** 構造内のフィールドには、次のものがあります。

名前	説明
dsmiDir	メッセージ・ファイルを含む、UNIX または Linux 上の完全修飾ディレクトリー・パス。これは、dsm.sys ディレクトリーも指定します。
dsmiConfig	クライアント・オプション・ファイルの完全修飾名。
dsmiLog	エラー・ログ・ディレクトリーの完全修飾パス。
argv	アプリケーションを許可ユーザー権限で実行する必要がある場合に、呼び出し側プログラムの argv[0] 名を渡します。詳しくは、 21 ページの『パスワード・ファイルへのアクセスの制御』 を参照してください。
logName	アプリケーションが dsiererror.log を使用しない場合の、エラー・ログのファイル名。
inclExclCaseSensitive	包含/除外ルールの大/小文字を区別するかしないかを示します。このパラメーターは、Windows でのみ使用できます。その他のシステムでは無視されます。

戻りコード

戻りコード番号を括弧 () 内に示してあります。

表 53. dsmSetUp の戻りコード

戻りコード	説明
DSM_RC_ACCESS_DENIED (106)	指定したファイルまたはディレクトリーへのアクセスがリジェクトされました。
DSM_RC_INVALID_OPT (0400)	無効なオプションが検出されました。
DSM_RC_NO_HOST_ADDR (0405)	システム・オプション・ファイルのサーバー名スタンザには、このサーバーについて TCPSERVERADDRESS が定義されていません。
DSM_RC_NO_OPT_FILE (0406)	filename で指定されたオプション・ファイルを検出できない。
DSM_RC_MACHINE_SAME (0408)	オプション・ファイルに定義される NODENAME (ノード名) をシステムの HostName (ホスト名) と同じにすることはできない。
DSM_RC_INVALID_SERVER (0409)	システム・オプション・ファイルに SERVERNAME オプションが含まれていません。
DSM_RC_INVALID_KEYWORD (0410)	dsmInitEx 構成ファイル、オプション・ストリング、dsm.sys、または dsm.opt で無効なオプション・キーワードが検出された。
DSM_RC_PATTERN_TOO_COMPLEX (0411)	発行された包含または除外パターンが複雑すぎて、IBM Spectrum Protect が正しく解釈できない。
DSM_RC_NO_CLOSING_BRACKET (0412)	包含または除外パターンが正しく構成されていない。右大括弧が脱落している。
DSM_RC-NLS_CANT_OPEN_TXT (0610)	システムがメッセージ・テキスト・ファイルをオープンできない。
DSM_RC-NLS_INVALID_CNTL_REC (0612)	システムがメッセージ・テキスト・ファイルを使用できない。

表 53. *dsmSetUp* の戻りコード (続き)

戻りコード	説明
DSM_RC_NOT_ADSM_AUTHORIZED (0927)	マルチスレッド化および <i>passwordaccess generate</i> を使用するには、許可ユーザーでなければならない。
DSM_RC_NO_INCLEXCL_FILE (2229)	包含/除外ファイルが見つからなかった。
DSM_RC_NO_SYS_OR_INCLEXCL (2230)	dsm.sys または包含/除外ファイルが見つからなかった。

dsmTerminate

dsmTerminate 関数呼び出しは、IBM Spectrum Protect サーバーとのセッションを終了して、IBM Spectrum Protect 環境を終結処理します。

構文

この呼び出しに特定の戻りコードはありません。

```
dsInt16_t dsmTerminate (dsUInt32_t dsmHandle);
```

パラメーター

dsUInt32_t dsmHandle (I)

この呼び出しを直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

dsmUpdateFS

dsmUpdateFS 関数呼び出しは、IBM Spectrum Protect ストレージ内のファイル・スペースを更新します。この更新により、管理者は確実に各ファイル・スペースの現行レコードを取得できます。

構文

```
dsInt16_t dsmUpdateFS
(dsUInt32_t dsmHandle,
 char *fs,
 dsmFSUpd *fsUpdP,
 dsUInt32_t fsUpdAct);
```

パラメーター

dsUInt32_t dsmHandle (I)

この呼び出しを直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

char *fs (I)

このパラメーターは、ファイル・スペース名を指すポインターです。

dsmFSUpd *fsUpdP (I)

このパラメーターは、必要な更新のための適切なフィールドを備えた構造を指すポインターです。更新を必要とするフィールドのみを埋め込みます。

dsUInt32_t fsUpdAct (I)

更新対象のフィールドを示す 2 バイトのビットマップ。ビット・マスクの値は、以下のとおりです。

- DSM_FSUPD_FSTYPE
- DSM_FSUPD_FSINFO

ヒント: Windows オペレーティング・システムでは、**FSINFO** が選択されると、**dsmDOSAttrib** からのドライブ文字値も更新されます。

- DSM_FSUPD_OCCUPANCY
- DSM_FSUPD_CAPACITY
- DSM_FSUPD_BACKSTARTDATE
- DSM_FSUPD_BACKCOMPLETEDATE

これらのビット・マスクの説明については、155 ページの『付録 B API タイプ定義ソース・ファイル』のトピックで DSM_FSUPD の定義を参照してください。

戻りコード

次の表は、**dsmUpdateFS** 関数呼び出しの戻りコードのリストです。

表 54. dsmUpdateFS の戻りコード		
戻りコード	戻りコード番号	説明
DSM_RC_FS_NOT_REGISTERED	2061	ファイル・スペース名が登録されていない。
DSM_RC_WRONG_VERSION_PARAMETER	2065	アプリケーション・クライアントの API バージョンが IBM Spectrum Protect ライブラリー・バージョン と異なる。
DSM_RC_FSINFO_TOOLONG	2106	ファイル・スペース情報が長すぎる。

dsmUpdateObj

dsmUpdateObj 関数呼び出しは、既にサーバー上にある 活動バックアップ・オブジェクトまたはアーカイブ・オブジェクトに 関連するメタ情報を更新します。アプリケーション・ビット・データは影響を受けません。オブジェクトを更新するには特定の非ワイルドカード名を付ける必要があります。アーカイブ済みオブジェクトを更新するには、**dsmSendType** を **stArchive** に設定します。最新の名前付きアーカイブ・オブジェクトのみが更新されます。

dsmUpdateObj 呼び出しを開始できるのは、セッション状態のときだけです。この呼び出しは、独自に所有するトランザクションを 実行するので、トランザクション内からこの呼び出しを呼び出すことはできません。また、一時点で更新できるのは、1 つのオブジェクトのみです。

制約事項: UNIX または Linux オペレーティング・システムでは、所有者 (owner) フィールドを変更した場合、root ユーザーでない限り、そのオブジェクトの照会やリストアができません。

構文

```
dsInt16_t dsmUpdateObj
(
    dsUInt32_t      dsmHandle,
    dsmSendType     sendType,
    void            *sendBuff,
    dsmObjName      *objNameP,
    ObjAttr         *objAttrPtr, /* objInfo */
    dsUInt16_t      objUpdAct); /* action bit vector */
```

パラメーター

フィールドの説明は、**dsmSendObj** と同じです。ただし、以下の違いがあります。

dsmObjName *objNameP (I)

ワイルドカードは、使用できません。

ObjAttr *objAttrPtr (I)

この呼び出しでは、**objCompressed** フィールドは無視されます。

その他の相違点は、次のとおりです。

- **owner (所有者)**。新しい **owner** フィールドを指定すると所有者が変わります。
- **sizeEstimate**。ゼロ以外の値を指定する場合は、送信される実際のデータ量をバイト単位で指定します。この値は、将来の利用に備えて IBM Spectrum Protect メタデータに保管されます。
- **objInfo (オブジェクト情報)**。この属性には、**objInfo** フィールドに入れられる新規の情報が入ります。**objInfoLength** を新規 **objInfo** の長さに設定してください。

dsUint16_t objUpdAct

objUpdAct のビット・マスクおよび可能なアクションは、次のとおりです。

DSM_BACKUPD_MC

オブジェクトの管理クラスを更新します。

DSM_BACKUPD_OBJINFO

objInfo、**objInfoLength**、および **sizeEstimate** を更新します。

DSM_BACKUPD_OWNER

オブジェクトの所有者を更新します。

DSM_ARCHUPD_DESCR

Description (説明) フィールドを更新します。新しい記述を **SendBuff** パラメーターを介して入力します。正しい使用法については、サンプル・プログラムを参照してください。

DSM_ARCHUPD_OBJINFO

objInfo、**objInfoLength**、および **sizeEstimate** を更新します。

DSM_ARCHUPD_OWNER

オブジェクトの所有者を更新します。

戻りコード

戻りコード番号を括弧 () 内に示してあります。

表 55. *dsmUpdateObj* の戻りコード

戻りコード	説明
DSM_RC_INVALID_ACTION (2232)	アクションが無効。
DSM_RC_FS_NOT_REGISTERED (2061)	ファイル・スペースが登録されていない。
DSM_RC_BAD_CALL_SEQUENCE (2041)	呼び出しの順序が無効。
DSM_RC_WILDCHAR_NOTALLOWED (2050)	ワイルドカード文字は使用できない。
DSM_RC_ABORT_NO_MATCH (2)	以前の照会と一致しない。

dsmUpdateObjEx

dsmUpdateObjEx 関数呼び出しは、サーバー上にある活動バックアップ・オブジェクトまたはアーカイブ・オブジェクトに関連するメタ情報を更新します。アプリケーション・ビット・データは影響を受けません。オブジェクトを更新するには、非ワイルドカード名を指定する必要があります。または、特定のアーカイブ・オブジェクトを更新するためのオブジェクト ID を指定することができます。この名前を指定す

るときに、ワイルドカード文字は使用できません。バックアップ・オブジェクトを更新するには、**stBackup** に対して **dsmSendType** パラメーターを設定します。アーカイブ・オブジェクトを更新するには、**dsmSendType** パラメーターを **stArchive** に設定します。

dsmUpdateObjEx 呼び出しを開始できるのは、セッション状態のときだけです。この呼び出しは、独自に所有するトランザクションを実行するので、トランザクション内からこの呼び出しを呼び出すことはできません。一時点で更新できるのは、1つのオブジェクトのみです。

制約事項: UNIX または Linux オペレーティング・システムでは、所有者 (owner) フィールドを変更した場合、root ユーザーでない限り、そのオブジェクトの照会やリストアができません。バックアップ・オブジェクトの現行アクティブ・バージョンのみを更新することができます。

構文

```
dsInt16_t dsmUpdateObjEx
(dsmUpdateObjExIn_t *dsmUpdateObjExInP,
 dsmUpdateObjExOut_t *dsmUpdateObjExOutP);
```

パラメーター

dsmUpdateObjExIn_t *dsmUpdateObjExInP

この構造には、以下の入力パラメーターが含まれます。

dsUInt16_t stVersion (I)

使用される構造の現行バージョン。

dsUInt32_t dsmHandle (I)

この呼び出しを直前の **dsmInitEx** 呼び出しに関連付けるハンドル。

dsmSendType sendType (I)

実行される送信のタイプ。指定できる値は次のとおりです。

stBackup

サーバーに送信されるバックアップ・オブジェクト。

stArchive

サーバーに送信されるアーカイブ・オブジェクト。

dsmObjName *objNameP (I)

ファイル・スペース名、高位オブジェクト名、低位オブジェクト名、およびオブジェクト・タイプを含む構造を指すポインター。ワイルドカードは、使用できません。

ObjAttr *objAttrPtr (I)

オブジェクト属性をアプリケーションに渡します。更新される値は、**objUpdAct** フィールドのフラグによって決まります。この呼び出しでは、**objCompressed** 属性は無視されます。

属性には、次のものがあります。

- **owner** は、新規名を入力する場合の所有者を変更するためのものです。
- **sizeEstimate** は、バイト単位で送信されるデータの実際の容量です。この値は、将来の利用に備えて IBM Spectrum Protect メタ・データに保管されます。
- **objCompressed** は、オブジェクト・データが既に圧縮されているかどうかを示すブール値です。
- **objInfo** は、**objInfo** フィールドに入れられる新規の情報を含む属性です。**objInfoLength** を新規 **objInfo** の長さに設定してください。
- **mcNameP** には、**dsmBindMC** から取得した管理クラスを指定変更する管理クラスの名前が入っています。

dsUInt32_t objUpdAct

指定する **objUpdAct** のビット・マスクおよびアクションは、次のとおりです。

DSM_BACKUPD_MC

オブジェクトの管理クラスを更新します。

DSM_BACKUPD_OBJINFO

バックアップ・オブジェクトの情報オブジェクト (**objInfo**)、情報オブジェクトの長さ (**objInfoLength**)、および送信されるデータの容量 (**sizeEstimate**) を更新します。

DSM_BACKUPD_OWNER

バックアップ・オブジェクトの所有者を更新します。

DSM_ARCHUPD_DESCR

アーカイブ・オブジェクトの **Description** (説明) フィールドを更新します。新しい記述を **sendBuff** パラメーターを介して入力します。

DSM_ARCHUPD_OBJINFO

アーカイブ・オブジェクトの情報オブジェクト (**objInfo**)、情報オブジェクトの長さ (**objInfoLength**)、および送信されるデータの容量 (**sizeEstimate**) を更新します。

DSM_ARCHUPD_OWNER

アーカイブ・オブジェクトの所有者を更新します。

ObjID archObjId

特定のアーカイブ・オブジェクトの固有のオブジェクト ID を指定します。複数のアーカイブ・オブジェクトは同じ名前を持つことができるため、このパラメーターは特定のアーカイブ・オブジェクトを識別するためのものです。照会アーカイブ呼び出しを使用してオブジェクト ID を取得することができます。

dsmUpdateObjExOut_t *dsmUpdateObjExOutP

この構造には、以下の出力パラメーターが含まれます。

dsUint16_t stVersion (I)

使用される構造の現行バージョン。

戻りコード

次の表では、戻りコード番号を括弧 () 内に示してあります。

表 56. *dsmUpdateObjEx* の戻りコード

戻りコード	説明
DSM_RC_INVALID_ACTION (2012)	アクションが無効。
DSM_RC_FS_NOT_REGISTERED (2061)	ファイル・スペースが登録されていない。
DSM_RC_BAD_CALL_SEQUENCE (2041)	呼び出しの順序が無効。
DSM_RC_WILDCHAR_NOTALLOWED (2050)	ワイルドカード文字は使用できない。
DSM_RC_ABORT_NO_MATCH (2)	以前の照会と一致しない。

付録 A API 戻りコード・ソース・ファイル: dsmrc.h

dsmrc.h ヘッダー・ファイルには、API がアプリケーションに返す可能性があるすべての戻りコードが入っています。

ここに示す情報には、API とともに配布される dsmrc.h ファイルのポイント・イン・タイム・コピーが含まれています。最新バージョンの API 配布パッケージ内のファイルを参照してください。

```

/*****
* IBM Spectrum Protect
* API Client Component
*
* IBM Confidential
* (IBM Confidential-Restricted when combined with the Aggregated OCO
* source modules for this program)
*
* OCO Source Materials
*
* 5648-020 (C) Copyright IBM Corporation 1993, 2016
*****/

/*****/
/* Header File Name: dsmrc.h */
/*
/* Descriptive-name: Return codes from IBM Spectrum Protect APIs */
/*****/
#ifndef _H_DSMRC
#define _H_DSMRC

#ifndef DSMAPILIB

#ifndef _H_ANSMACH
typedef int RetCode ;
#endif

#endif

#define DSM_RC_SUCCESSFUL          0 /* successful completion */
#define DSM_RC_OK                  0 /* successful completion */

#define DSM_RC_UNSUCCESSFUL        -1 /* unsuccessful completion */

/* dsmEndTxn reason code */
#define DSM_RS_ABORT_SYSTEM_ERROR      1
#define DSM_RS_ABORT_NO_MATCH          2
#define DSM_RS_ABORT_BY_CLIENT         3
#define DSM_RS_ABORT_ACTIVE_NOT_FOUND  4
#define DSM_RS_ABORT_NO_DATA           5
#define DSM_RS_ABORT_BAD_VERIFIER       6
#define DSM_RS_ABORT_NODE_IN_USE        7
#define DSM_RS_ABORT_EXPDATE_TOO_LOW    8
#define DSM_RS_ABORT_DATA_OFFLINE       9
#define DSM_RS_ABORT_EXCLUDED_BY_SIZE   10
#define DSM_RS_ABORT_NO_STO_SPACE_SKIP   11
#define DSM_RS_ABORT_NO_REPOSIT_SPACE    DSM_RS_ABORT_NO_STO_SPACE_SKIP
#define DSM_RS_ABORT_MOUNT_NOT_POSSIBLE  12
#define DSM_RS_ABORT_SIZEESTIMATE_EXCEED 13
#define DSM_RS_ABORT_DATA_UNAVAILABLE    14
#define DSM_RS_ABORT_RETRY               15
#define DSM_RS_ABORT_NO_LOG_SPACE        16
#define DSM_RS_ABORT_NO_DB_SPACE         17
#define DSM_RS_ABORT_NO_MEMORY           18

#define DSM_RS_ABORT_FS_NOT_DEFINED      20
#define DSM_RS_ABORT_NODE_ALREADY_DEFED  21
#define DSM_RS_ABORT_NO_DEFAULT_DOMAIN   22
#define DSM_RS_ABORT_INVALID_NODENAME    23
#define DSM_RS_ABORT_INVALID_POL_BIND    24
#define DSM_RS_ABORT_DEST_NOT_DEFINED    25
#define DSM_RS_ABORT_WAIT_FOR_SPACE       26
#define DSM_RS_ABORT_NOT_AUTHORIZED      27
#define DSM_RS_ABORT_RULE_ALREADY_DEFED   28
#define DSM_RS_ABORT_NO_STOR_SPACE_STOP  29
```

```

#define DSM_RS_ABORT_LICENSE_VIOLATION 30
#define DSM_RS_ABORT_EXTOBJID_ALREADY_EXISTS 31
#define DSM_RS_ABORT_DUPLICATE_OBJECT 32

#define DSM_RS_ABORT_INVALID_OFFSET 33 /* Partial Object Retrieve */
#define DSM_RS_ABORT_INVALID_LENGTH 34 /* Partial Object Retrieve */
#define DSM_RS_ABORT_STRING_ERROR 35
#define DSM_RS_ABORT_NODE_NOT_AUTHORIZED 36
#define DSM_RS_ABORT_RESTART_NOT_POSSIBLE 37
#define DSM_RS_ABORT_RESTORE_IN_PROGRESS 38
#define DSM_RS_ABORT_SYNTAX_ERROR 39

#define DSM_RS_ABORT_DATA_SKIPPED 40
#define DSM_RS_ABORT_EXCEED_MAX_MP 41
#define DSM_RS_ABORT_NO_OBJSET_MATCH 42
#define DSM_RS_ABORT_PVR_ERROR 43
#define DSM_RS_ABORT_BAD_RECOGToken 44
#define DSM_RS_ABORT_MERGE_ERROR 45
#define DSM_RS_ABORT_FSRENAME_ERROR 46
#define DSM_RS_ABORT_INVALID_OPERATION 47
#define DSM_RS_ABORT_STGPOOL_UNDEFINED 48
#define DSM_RS_ABORT_INVALID_DATA_FORMAT 49
#define DSM_RS_ABORT_DATAMOVER_UNDEFINED 50

#define DSM_RS_ABORT_INVALID_MOVER_TYPE 231
#define DSM_RS_ABORT_ITEM_IN_USE 232
#define DSM_RS_ABORT_LOCK_CONFLICT 233
#define DSM_RS_ABORT_SRV_PLUGIN_COMM_ERROR 234
#define DSM_RS_ABORT_SRV_PLUGIN_OS_ERROR 235
#define DSM_RS_ABORT_CRC_FAILED 236
#define DSM_RS_ABORT_INVALID_GROUP_ACTION 237
#define DSM_RS_ABORT_DISK_UNDEFINED 238
#define DSM_RS_ABORT_BAD_DESTINATION 239
#define DSM_RS_ABORT_DATAMOVER_NOT_AVAILABLE 240
#define DSM_RS_ABORT_STGPOOL_COPY_CONT_NO 241
#define DSM_RS_ABORT_RETRY_SINGLE_TXN 242
#define DSM_RS_ABORT_TOC_CREATION_FAIL 243
#define DSM_RS_ABORT_TOC_LOAD_FAIL 244
#define DSM_RS_ABORT_PATH_RESTRICTED 245
#define DSM_RS_ABORT_NO_LANFREE_SCRATCH 246
#define DSM_RS_ABORT_INSERT_NOT_ALLOWED 247
#define DSM_RS_ABORT_DELETE_NOT_ALLOWED 248
#define DSM_RS_ABORT_TXN_LIMIT_EXCEEDED 249
#define DSM_RS_ABORT_OBJECT_ALREADY_HELD 250
#define DSM_RS_ABORT_INVALID_CHUNK_REFERENCE 254
#define DSM_RS_ABORT_DESTINATION_NOT_DEDUP 255
#define DSM_RS_ABORT_DESTINATION_POOL_CHANGED 257
#define DSM_RS_ABORT_NOT_ROOT 258

/* RETURN CODE */

#define DSM_RC_ABORT_SYSTEM_ERROR DSM_RS_ABORT_SYSTEM_ERROR
#define DSM_RC_ABORT_NO_MATCH DSM_RS_ABORT_NO_MATCH
#define DSM_RC_ABORT_BY_CLIENT DSM_RS_ABORT_BY_CLIENT
#define DSM_RC_ABORT_ACTIVE_NOT_FOUND DSM_RS_ABORT_ACTIVE_NOT_FOUND
#define DSM_RC_ABORT_NO_DATA DSM_RS_ABORT_NO_DATA
#define DSM_RC_ABORT_BAD_VERIFIER DSM_RS_ABORT_BAD_VERIFIER
#define DSM_RC_ABORT_NODE_IN_USE DSM_RS_ABORT_NODE_IN_USE
#define DSM_RC_ABORT_EXPIRATE_TOO_LOW DSM_RS_ABORT_EXPIRATE_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_SIZEESTIMATE_EXCEED DSM_RS_ABORT_SIZEESTIMATE_EXCEED
#define DSM_RC_ABORT_DATA_UNAVAILABLE DSM_RS_ABORT_DATA_UNAVAILABLE
#define DSM_RC_ABORT_RETRY DSM_RS_ABORT_RETRY
#define DSM_RC_ABORT_NO_LOG_SPACE DSM_RS_ABORT_NO_LOG_SPACE
#define DSM_RC_ABORT_NO_DB_SPACE DSM_RS_ABORT_NO_DB_SPACE
#define DSM_RC_ABORT_NO_MEMORY DSM_RS_ABORT_NO_MEMORY

#define DSM_RC_ABORT_FS_NOT_DEFINED DSM_RS_ABORT_FS_NOT_DEFINED
#define DSM_RC_ABORT_NODE_ALREADY_DEFED DSM_RS_ABORT_NODE_ALREADY_DEFED
#define DSM_RC_ABORT_NO_DEFAULT_DOMAIN DSM_RS_ABORT_NO_DEFAULT_DOMAIN
#define DSM_RC_ABORT_INVALID_NODENAME DSM_RS_ABORT_INVALID_NODENAME
#define DSM_RC_ABORT_INVALID_POL_BIND DSM_RS_ABORT_INVALID_POL_BIND
#define DSM_RC_ABORT_DEST_NOT_DEFINED DSM_RS_ABORT_DEST_NOT_DEFINED
#define DSM_RC_ABORT_WAIT_FOR_SPACE DSM_RS_ABORT_WAIT_FOR_SPACE

```

```

#define DSM_RC_ABORT_NOT_AUTHORIZED DSM_RS_ABORT_NOT_AUTHORIZED
#define DSM_RC_ABORT_RULE_ALREADY_DEFED DSM_RS_ABORT_RULE_ALREADY_DEFED
#define DSM_RC_ABORT_NO_STOR_SPACE_STOP DSM_RS_ABORT_NO_STOR_SPACE_STOP

#define DSM_RC_ABORT_LICENSE_VIOLATION DSM_RS_ABORT_LICENSE_VIOLATION
#define DSM_RC_ABORT_EXTOBJID_ALREADY_EXISTS DSM_RS_ABORT_EXTOBJID_ALREADY_EXISTS
#define DSM_RC_ABORT_DUPLICATE_OBJECT DSM_RS_ABORT_DUPLICATE_OBJECT

#define DSM_RC_ABORT_INVALID_OFFSET DSM_RS_ABORT_INVALID_OFFSET
#define DSM_RC_ABORT_INVALID_LENGTH DSM_RS_ABORT_INVALID_LENGTH

#define DSM_RC_ABORT_STRING_ERROR DSM_RS_ABORT_STRING_ERROR
#define DSM_RC_ABORT_NODE_NOT_AUTHORIZED DSM_RS_ABORT_NODE_NOT_AUTHORIZED
#define DSM_RC_ABORT_RESTART_NOT_POSSIBLE DSM_RS_ABORT_RESTART_NOT_POSSIBLE
#define DSM_RC_ABORT_RESTORE_IN_PROGRESS DSM_RS_ABORT_RESTORE_IN_PROGRESS
#define DSM_RC_ABORT_SYNTAX_ERROR DSM_RS_ABORT_SYNTAX_ERROR

#define DSM_RC_ABORT_DATA_SKIPPED DSM_RS_ABORT_DATA_SKIPPED
#define DSM_RC_ABORT_EXCEED_MAX_MP DSM_RS_ABORT_EXCEED_MAX_MP
#define DSM_RC_ABORT_NO_OBJSET_MATCH DSM_RS_ABORT_NO_OBJSET_MATCH
#define DSM_RC_ABORT_PVR_ERROR DSM_RS_ABORT_PVR_ERROR
#define DSM_RC_ABORT_BAD_RECOGTOKEN DSM_RS_ABORT_BAD_RECOGTOKEN
#define DSM_RC_ABORT_MERGE_ERROR DSM_RS_ABORT_MERGE_ERROR
#define DSM_RC_ABORT_FSRENAME_ERROR DSM_RS_ABORT_FSRENAME_ERROR
#define DSM_RC_ABORT_INVALID_OPERATION DSM_RS_ABORT_INVALID_OPERATION
#define DSM_RC_ABORT_STGPOOL_UNDEFINED DSM_RS_ABORT_STGPOOL_UNDEFINED
#define DSM_RC_ABORT_INVALID_DATA_FORMAT DSM_RS_ABORT_INVALID_DATA_FORMAT
#define DSM_RC_ABORT_DATAMOVER_UNDEFINED DSM_RS_ABORT_DATAMOVER_UNDEFINED

#define DSM_RC_ABORT_INVALID_MOVER_TYPE DSM_RS_ABORT_INVALID_MOVER_TYPE
#define DSM_RC_ABORT_ITEM_IN_USE DSM_RS_ABORT_ITEM_IN_USE
#define DSM_RC_ABORT_LOCK_CONFLICT DSM_RS_ABORT_LOCK_CONFLICT
#define DSM_RC_ABORT_SRV_PLUGIN_COMM_ERROR DSM_RS_ABORT_SRV_PLUGIN_COMM_ERROR
#define DSM_RC_ABORT_SRV_PLUGIN_OS_ERROR DSM_RS_ABORT_SRV_PLUGIN_OS_ERROR
#define DSM_RC_ABORT_CRC_FAILED DSM_RS_ABORT_CRC_FAILED
#define DSM_RC_ABORT_INVALID_GROUP_ACTION DSM_RS_ABORT_INVALID_GROUP_ACTION
#define DSM_RC_ABORT_DISK_UNDEFINED DSM_RS_ABORT_DISK_UNDEFINED
#define DSM_RC_ABORT_BAD_DESTINATION DSM_RS_ABORT_BAD_DESTINATION
#define DSM_RC_ABORT_DATAMOVER_NOT_AVAILABLE DSM_RS_ABORT_DATAMOVER_NOT_AVAILABLE
#define DSM_RC_ABORT_STGPOOL_COPY_CONT_NO DSM_RS_ABORT_STGPOOL_COPY_CONT_NO
#define DSM_RC_ABORT_RETRY_SINGLE_TXN DSM_RS_ABORT_RETRY_SINGLE_TXN
#define DSM_RC_ABORT_TOC_CREATION_FAIL DSM_RS_ABORT_TOC_CREATION_FAIL
#define DSM_RC_ABORT_TOC_LOAD_FAIL DSM_RS_ABORT_TOC_LOAD_FAIL
#define DSM_RC_ABORT_PATH_RESTRICTED DSM_RS_ABORT_PATH_RESTRICTED
#define DSM_RC_ABORT_NO_LANFREE_SCRATCH DSM_RS_ABORT_NO_LANFREE_SCRATCH
#define DSM_RC_ABORT_INSERT_NOT_ALLOWED DSM_RS_ABORT_INSERT_NOT_ALLOWED
#define DSM_RC_ABORT_DELETE_NOT_ALLOWED DSM_RS_ABORT_DELETE_NOT_ALLOWED
#define DSM_RC_ABORT_TXN_LIMIT_EXCEEDED DSM_RS_ABORT_TXN_LIMIT_EXCEEDED
#define DSM_RC_ABORT_OBJECT_ALREADY_HELD DSM_RS_ABORT_OBJECT_ALREADY_HELD
#define DSM_RC_ABORT_INVALID_CHUNK_REFERENCE DSM_RS_ABORT_INVALID_CHUNK_REFERENCE
#define DSM_RC_ABORT_DESTINATION_NOT_DEDUP DSM_RS_ABORT_DESTINATION_NOT_DEDUP
#define DSM_RC_ABORT_DESTINATION_POOL_CHANGED DSM_RS_ABORT_DESTINATION_POOL_CHANGED
#define DSM_RC_ABORT_NOT_ROOT DSM_RS_ABORT_NOT_ROOT

#define DSM_RC_ABORT_CERTIFICATE_NOT_FOUND DSM_RS_ABORT_CERTIFICATE_NOT_FOUND

/* Definitions for server signon reject codes */
/* These error codes are in the range (51 to 99) inclusive. */
#define DSM_RC_REJECT_NO_RESOURCES 51
#define DSM_RC_REJECT_VERIFIER_EXPIRED 52
#define DSM_RC_REJECT_ID_UNKNOWN 53
#define DSM_RC_REJECT_DUPLICATE_ID 54
#define DSM_RC_REJECT_SERVER_DISABLED 55
#define DSM_RC_REJECT_CLOSED_REGISTER 56
#define DSM_RC_REJECT_CLIENT_DOWNLEVEL 57
#define DSM_RC_REJECT_SERVER_DOWNLEVEL 58
#define DSM_RC_REJECT_ID_IN_USE 59
#define DSM_RC_REJECT_ID_LOCKED 61
#define DSM_RC_SIGNONREJECT_LICENSE_MAX 62
#define DSM_RC_REJECT_NO_MEMORY 63
#define DSM_RC_REJECT_NO_DB_SPACE 64
#define DSM_RC_REJECT_NO_LOG_SPACE 65
#define DSM_RC_REJECT_INTERNAL_ERROR 66
#define DSM_RC_SIGNONREJECT_INVALID_CLI 67 /* client type not licensed */
#define DSM_RC_CLIENT_NOT_ARCHRETPROT 68
#define DSM_RC_REJECT_LASTSESS_CANCELED 69
#define DSM_RC_REJECT_UNICODE_NOT_ALLOWED 70
#define DSM_RC_REJECT_NOT_AUTHORIZED 71
#define DSM_RC_REJECT_TOKEN_TIMEOUT 72
#define DSM_RC_REJECT_INVALID_NODE_TYPE 73
#define DSM_RC_REJECT_INVALID_SESSIONINIT 74
#define DSM_RC_REJECT_WRONG_PORT 75

```

```

#define DSM_RC_USER_ABORT 101 /* processing aborted by user */
#define DSM_RC_NO_MEMORY 102 /* no RAM left to complete request */
#define DSM_RC_TA_COMM_DOWN 2021 /* no longer used */
#define DSM_RC_FILE_NOT_FOUND 104 /* specified file not found */
#define DSM_RC_PATH_NOT_FOUND 105 /* specified path doesn't exist */
#define DSM_RC_ACCESS_DENIED 106 /* denied due to improper permission */
#define DSM_RC_NO_HANDLES 107 /* no more file handles available */
#define DSM_RC_FILE_EXISTS 108 /* file already exists */
#define DSM_RC_INVALID_PARM 109 /* invalid parameter passed. CRITICAL */
#define DSM_RC_INVALID_HANDLE 110 /* invalid file handle passed */
#define DSM_RC_DISK_FULL 111 /* out of disk space */
#define DSM_RC_PROTOCOL_VIOLATION 113 /* call protocol violation. CRITICAL */
#define DSM_RC_UNKNOWN_ERROR 114 /* unknown system error. CRITICAL */
#define DSM_RC_UNEXPECTED_ERROR 115 /* unexpected error. CRITICAL */
#define DSM_RC_FILE_BEING_EXECUTED 116 /* No write is allowed */
#define DSM_RC_DIR_NO_SPACE 117 /* directory can't be expanded */
#define DSM_RC_LOOPED_SYM_LINK 118 /* too many symbolic links were
    encountered in translating path. */
#define DSM_RC_FILE_NAME_TOO_LONG 119 /* file name too long */
#define DSM_RC_FILE_SPACE_LOCKED 120 /* filespace is locked by the system */
#define DSM_RC_FINISHED 121 /* finished processing */
#define DSM_RC_UNKNOWN_FORMAT 122 /* unknown format */
#define DSM_RC_NO_AUTHORIZATION 123 /* server response when the client has
    no authorization to read another
    host's owner backup/archive data */
#define DSM_RC_FILE_SPACE_NOT_FOUND 124 /* specified file space not found */
#define DSM_RC_TXN_ABORTED 125 /* transaction aborted */
#define DSM_RC_SUBDIR_AS_FILE 126 /* Subdirectory name exists as file */
#define DSM_RC_PROCESS_NO_SPACE 127 /* process has no more disk space. */
#define DSM_RC_PATH_TOO_LONG 128 /* a directory path being build became
    too long */
#define DSM_RC_NOT_COMPRESSED 129 /* file thought to be compressed is
    actually not */
#define DSM_RC_TOO_MANY_BITS 130 /* file was compressed using more bits
    then the expander can handle */
#define DSM_RC_SYSTEM_ERROR 131 /* internal system error */
#define DSM_RC_NO_SERVER_RESOURCES 132 /* server out of resources. */
#define DSM_RC_FS_NOT_KNOWN 133 /* the file space is not known by the
    server */
#define DSM_RC_NO_LEADING_DIRSEP 134 /* no leading directory separator */
#define DSM_RC_WILDCARD_DIR 135 /* wildcard character in directory
    path when not allowed */
#define DSM_RC_COMM_PROTOCOL_ERROR 136 /* communications protocol error */
#define DSM_RC_AUTH_FAILURE 137 /* authentication failure */
#define DSM_RC_TA_NOT_VALID 138 /* TA not a root and/or SUID program */
#define DSM_RC_KILLED 139 /* process killed. */

#define DSM_RC_RETRY 143 /* retry same operation again */

#define DSM_RC_WOULD_BLOCK 145 /* operation would cause the system to
    block waiting for input. */
#define DSM_RC_TOO_SMALL 146 /* area for compiled pattern small */
#define DSM_RC_UNCLOSED 147 /* no closing bracket in pattern */
#define DSM_RC_NO_STARTING_DELIMITER 148 /* pattern has to start with
    directory delimiter */
#define DSM_RC_NEEDED_DIR_DELIMITER 149 /* a directory delimiter is needed
    immediately before and after the
    "match directories" metastring
    ("...") and one wasn't found */
#define DSM_RC_UNKNOWN_FILE_DATA_TYPE 150 /* structured file data type is
    unknown */
#define DSM_RC_BUFFER_OVERFLOW 151 /* data buffer overflow */

#define DSM_RC_NO_COMPRESS_MEMORY 154 /* Compress/Expand out of memory */
#define DSM_RC_COMPRESS_GREW 155 /* Compression grew */
#define DSM_RC_INV_COMM_METHOD 156 /* Invalid comm method specified */
#define DSM_RC_WILL_ABORT 157 /* Transaction will be aborted */
#define DSM_RC_FS_WRITE_LOCKED 158 /* File space is write locked */
#define DSM_RC_SKIPPED_BY_USER 159 /* User wanted file skipped in the
    case of ABORT_DATA_OFFLINE */
#define DSM_RC_TA_NOT_FOUND 160 /* TA not found in it's directory */
#define DSM_RC_TA_ACCESS_DENIED 161 /* Access to TA is denied */
#define DSM_RC_FS_NOT_READY 162 /* File space not ready */
#define DSM_RC_FS_IS_BAD 163 /* File space is bad */
#define DSM_RC_FIO_ERROR 164 /* File input/output error */
#define DSM_RC_WRITE_FAILURE 165 /* Error writing to file */
#define DSM_RC_OVER_FILE_SIZE_LIMIT 166 /* File over system/user limit */

```

```

#define DSM_RC_CANNOT_MAKE          167 /* Could not create file/directory,
                                         could be a bad name */
#define DSM_RC_NO_PASS_FILE         168 /* password file needed and user is
                                         not root */
#define DSM_RC_VERFILE_OLD          169 /* password stored locally doesn't
                                         match the one at the host */
#define DSM_RC_INPUT_ERROR          173 /* unable to read keyboard input */
#define DSM_RC_REJECT_PLATFORM_MISMATCH 174 /* Platform name doesn't match
                                         up with what the server says
                                         is the platform for the client */
#define DSM_RC_TL_NOT_FILE_OWNER    175 /* User trying to backup a file is not
                                         the file's owner. */
#define DSM_RC_COMPRESSED_DATA_CORRUPTED 176 /* Compressed data is corrupted */
#define DSM_RC_UNMATCHED_QUOTE      177 /* missing starting or ending quote */

#define DSM_RC_SIGNON_FAILOVER_MODE 178 /* Failed over to the replication server,
                                         running in failover mode */
#define DSM_RC_FAILOVER_MODE_FUNC_BLOCKED 179 /* function is blocked because
                                         session is in failover mode */

/*-----*/
/* Return codes 180-199 are reserved for Policy Set handling */
/*-----*/
#define DSM_RC_PS_MULTBCG           181 /* Multiple backup copy groups in 1 MC*/
#define DSM_RC_PS_MULTACG           182 /* Multiple arch. copy groups in 1 MC*/
#define DSM_RC_PS_NODFLTMC          183 /* Default MC name not in policy set */
#define DSM_RC_TL_NOBCG             184 /* Backup req, no backup copy group */
#define DSM_RC_TL_EXCLUDED          185 /* Backup req, excl. by in/ex filter */
#define DSM_RC_TL_NOACG             186 /* Archive req, no archive copy group */
#define DSM_RC_PS_INVALID_ARCHMC    187 /* Invalid MC name in archive override*/
#define DSM_RC_NO_PS_DATA           188 /* No policy set data on the server */
#define DSM_RC_PS_INVALID_DIRMC     189 /* Invalid directory MC specified in
                                         the options file. */
#define DSM_RC_PS_NO_CG_IN_DIR_MC   190 /* No backup copy group in directory MC.
                                         Must specify an MC using DirMC
                                         option. */

#define DSM_RC_WIN32_UNSUPPORTED_FILE_TYPE 280 /* File is not of
                                         Win32 type FILE_TYPE_DISK */

/*-----*/
/* Return codes for the Trusted Communication Agent */
/*-----*/
#define DSM_RC_TCA_NOT_ROOT         161 /* Access to TA is denied */
#define DSM_RC_TCA_ATTACH_SHR_MEM_ERR 200 /* Error attaching shared memory */
#define DSM_RC_TCA_SHR_MEM_BLOCK_ERR 200 /* Shared memory block error */
#define DSM_RC_TCA_SHR_MEM_IN_USE    200 /* Shared memory block error */
#define DSM_RC_TCA_SHARED_MEMORY_ERROR 200 /* Shared memory block error */
#define DSM_RC_TCA_SEGMENT_MISMATCH 200 /* Shared memory block error */
#define DSM_RC_TCA_FORK_FAILED       292 /* Error forking off TCA process */
#define DSM_RC_TCA_DIED              294 /* TCA died unexpectedly */
#define DSM_RC_TCA_INVALID_REQUEST  295 /* Invalid request sent to TCA */
#define DSM_RC_TCA_SEMGET_ERROR      297 /* Error getting semaphores */
#define DSM_RC_TCA_SEM_OP_ERROR      298 /* Error in semaphore set or wait */
#define DSM_RC_TCA_NOT_ALLOWED       299 /* TCA not allowed (multi thread) */

/*-----*/
/* 400-430 for options */
/*-----*/
#define DSM_RC_INVALID_OPT           400 /* invalid option */
#define DSM_RC_NO_HOST_ADDR          405 /* Not enuf info to connect server */
#define DSM_RC_NO_OPT_FILE           406 /* No default user configuration file*/
#define DSM_RC_MACHINE_SAME          408 /* -MACHINE_NAME same as real name */
#define DSM_RC_INVALID_SERVER        409 /* Invalid server name from client */
#define DSM_RC_INVALID_KEYWORD       410 /* Invalid option keyword */
#define DSM_RC_PATTERN_TOO_COMPLEX  411 /* Can't match Include/Exclude entry*/
#define DSM_RC_NO_CLOSING_BRACKET    412 /* Missing closing bracket inc/excl */
#define DSM_RC_OPT_CLIENT_NOT_ACCEPTING 417/* Client doesn't accept this option
                                         from the server */
#define DSM_RC_OPT_CLIENT_DOES_NOT_WANT 418/* Client doesn't want this value
                                         from the server */
#define DSM_RC_OPT_NO_INCLEXCL_FILE  419 /* inclexcl file not found */
#define DSM_RC_OPT_OPEN_FAILURE      420 /* can't open file */
#define DSM_RC_OPT_INV_NODENAME      421/* used for Windows if nodename=local
                                         machine when CLUSTERMODE=YES */
#define DSM_RC_OPT_NODENAME_INVALID  423/* generic invalid nodename */
#define DSM_RC_OPT_ERRORLOG_CONFLICT 424/* both logmax & retention specified */
#define DSM_RC_OPT_SCHEDLOG_CONFLICT 425/* both logmax & retention specified */
#define DSM_RC_CANNOT_OPEN_TRACEFILE 426/* cannot open trace file */
#define DSM_RC_CANNOT_OPEN_LOGFILE   427/* cannot open error log file */
#define DSM_RC_OPT_SESSINIT_LF_CONFLICT 428/* both sessioninit=server and
                                         enablelanfree=yes are specified*/

```

```

#define DSM_RC_OPT_OPTION_IGNORE      429 /* option will be ignored */
#define DSM_RC_OPT_DEDUP_CONFLICT     430 /* cannot open error log file */
#define DSM_RC_OPT_HSMLOG_CONFLICT    431 /* both logmax & retention specified */

/*-----*/
/* 600 to 610 for volume label codes */
/*-----*/
#define DSM_RC_DUP_LABEL              600 /* duplicate volume label found */
#define DSM_RC_NO_LABEL               601 /* drive has no label */

/*-----*/
/* Return codes for message file processing */
/*-----*/
#define DSM_RC_NLS_CANT_OPEN_TXT      610 /* error trying to open msg txt file */
#define DSM_RC_NLS_CANT_READ_HDR      611 /* error trying to read header */
#define DSM_RC_NLS_INVALID_CNTL_REC   612 /* invalid control record */
#define DSM_RC_NLS_INVALID_DATE_FMT   613 /* invalid default date format */
#define DSM_RC_NLS_INVALID_TIME_FMT   614 /* invalid default time format */
#define DSM_RC_NLS_INVALID_NUM_FMT    615 /* invalid default number format */

/*-----*/
/* Return codes 620-630 are reserved for log message return codes */
/*-----*/
#define DSM_RC_LOG_CANT_BE_OPENED      620 /* error trying to open error log */
#define DSM_RC_LOG_ERROR_WRITING_TO_LOG 621 /* error occurred writing to
log file */
#define DSM_RC_LOG_NOT_SPECIFIED       622 /* no error log file was specified */

/*-----*/
/* Return codes 900-999 IBM SPECTRUM PROTECT CLIENT ONLY */
/*-----*/
#define DSM_RC_NOT_ADSM_AUTHORIZED     927 /* Must be ADSM authorized to perform */
/* action : root user or pwd auth */
#define DSM_RC_REJECT_USERID_UNKNOWN   940 /* userid unknown on server */
#define DSM_RC_FILE_IS_SYMLINK         959 /* errorlog or trace is a symbolic
link */

/*
#define DSM_RC_DIRECT_STORAGE_AGENT_UNSUPPORTED 961 /* Direct connection to SA not supported */
#define DSM_RC_FS_NAMESPACES_DOWNLEVEL 963 /* Long namespace has been removed from
from the Netware volume */
#define DSM_RC_CONTINUE_NEW_CONSUMER   972 /* Continue processing using a new consumer */
#define DSM_RC_CONTINUE_NEW_CONSUMER_NODEDUP 973 /* Continue processing using a new consumer no dedup */
#define DSM_RC_CONTINUE_NEW_CONSUMER_NOCOMPRESS 976 /* Continue processing using a new consumer no
compression */

#define DSM_RC_SERVER_SUPPORTS_FUNC     994 /* the server supports this function */
#define DSM_RC_SERVER_AND_SA_SUPPORT_FUNC 995 /* Both server and SA support func */
#define DSM_RC_SERVER_DOWNLEVEL_FUNC    996 /* The server is downlevel for func */
#define DSM_RC_STORAGEAGENT_DOWNLEVEL    997 /* the storage agent is downlevel */
#define DSM_RC_SERVER_AND_SA_DOWNLEVEL  998 /* both server and SA downlevel */

/* TCP/IP error codes */
#define DSM_RC_TCPIP_FAILURE             -50 /* TCP/IP communications failure */
#define DSM_RC_CONN_TIMEDOUT             -51 /* TCP/IP connection attempt timedout */
#define DSM_RC_CONN_REFUSED              -52 /* TCP/IP connection refused by host */
#define DSM_RC_BAD_HOST_NAME             -53 /* TCP/IP invalid host name specified */
#define DSM_RC_NETWORK_UNREACHABLE       -54 /* TCP/IP host name unreachable */
#define DSM_RC_WINSOCK_MISSING           -55 /* TCP/IP WINSOCK.DLL missing */
#define DSM_RC_TCPIP_DLL_LOADFAILURE     -56 /* Error from LoadLibrary */
#define DSM_RC_TCPIP_LOADFAILURE          -57 /* Error from GetProcAddress */
#define DSM_RC_TCPIP_USER_ABORT          -58 /* User aborted while in TCP/IP layer */

/*-----*/
/* Return codes (-71)-(-90) are reserved for CommTSM error codes */
/*-----*/
#define DSM_RC_TSM_FAILURE                -71 /* IBM Spectrum Protect comm failure */
#define DSM_RC_TSM_ABORT                  -72 /* Session aborted abnormally */

/*comm3270 error codes - no longer used*/
#define DSM_RC_COMM_TIMEOUT              2021 /* no longer used */
#define DSM_RC_EMULATOR_INACTIVE        2021 /* no longer used */
#define DSM_RC_BAD_HOST_ID               2021 /* no longer used */
#define DSM_RC_HOST_SESS_BUSY             2021 /* no longer used */
#define DSM_RC_3270_CONNECT_FAILURE      2021 /* no longer used */
#define DSM_RC_NO_ACS3ELKE_DLL           2021 /* no longer used */
#define DSM_RC_EMULATOR_ERROR           2021 /* no longer used */
#define DSM_RC_EMULATOR_BACKLEVEL       2021 /* no longer used */

```

```

#define DSM_RC_CKSUM_FAILURE      2021    /* no longer used          */

/* The following Return codes are for EHLLAPI for Windows */
#define DSM_RC_3270COMMError_DLL  2021    /* no longer used          */
#define DSM_RC_3270COMMError_GetProc 2021    /* no longer used          */
#define DSM_RC_EHLLAPIError_DLL   2021    /* no longer used          */
#define DSM_RC_EHLLAPIError_GetProc 2021    /* no longer used          */
#define DSM_RC_EHLLAPIError_HostConnect 2021 /* no longer used          */
#define DSM_RC_EHLLAPIError_AllocBuff 2021 /* no longer used          */
#define DSM_RC_EHLLAPIError_SendKey  2021 /* no longer used          */
#define DSM_RC_EHLLAPIError_PacketChk 2021 /* no longer used          */
#define DSM_RC_EHLLAPIError_ChkSum   2021 /* no longer used          */
#define DSM_RC_EHLLAPIError_HostTimeOut 2021 /* no longer used          */
#define DSM_RC_EHLLAPIError_Send     2021 /* no longer used          */
#define DSM_RC_EHLLAPIError_Recv     2021 /* no longer used          */
#define DSM_RC_EHLLAPIError_General  2021 /* no longer used          */
#define DSM_RC_PC3270_MISSING_DLL    2021 /* no longer used          */
#define DSM_RC_3270COMM_MISSING_DLL  2021 /* no longer used          */

/* NETBIOS error codes */
#define DSM_RC_NETB_ERROR          -151 /* Could not add node to LAN */
#define DSM_RC_NETB_NO_DLL         -152 /* The ACSNETB.DLL could not be loaded */
#define DSM_RC_NETB_LAN_ERR        -155 /* LAN error detected */
#define DSM_RC_NETB_NAME_ERR       -158 /* Netbios error on Add Name */
#define DSM_RC_NETB_TIMEOUT        -159 /* Netbios send timeout */
#define DSM_RC_NETB_NOTINST        -160 /* Netbios not installed - DOS */
#define DSM_RC_NETB_REBOOT         -161 /* Netbios config err - reboot DOS */

/* Named Pipe error codes */
#define DSM_RC_NP_ERROR            -190

/* CPIC error codes */
#define DSM_RC_CPIC_ALLOCATE_FAILURE 2021 /* no longer used */
#define DSM_RC_CPIC_TYPE_MISMATCH    2021 /* no longer used */
#define DSM_RC_CPIC_PIPE_NOT_SPECIFY_ERR 2021 /* no longer used */
#define DSM_RC_CPIC_SECURITY_NOT_VALID 2021 /* no longer used */
#define DSM_RC_CPIC_SYNC_LVL_NO_SUPPORT 2021 /* no longer used */
#define DSM_RC_CPIC_TPN_NOT_RECOGNIZED 2021 /* no longer used */
#define DSM_RC_CPIC_TP_ERROR          2021 /* no longer used */
#define DSM_RC_CPIC_PARAMETER_ERROR   2021 /* no longer used */
#define DSM_RC_CPIC_PROD_SPECIFIC_ERR 2021 /* no longer used */
#define DSM_RC_CPIC_PROGRAM_ERROR     2021 /* no longer used */
#define DSM_RC_CPIC_RESOURCE_ERROR    2021 /* no longer used */
#define DSM_RC_CPIC_DEALLOCATE_ERROR  2021 /* no longer used */
#define DSM_RC_CPIC_SVC_ERROR          2021 /* no longer used */
#define DSM_RC_CPIC_PROGRAM_STATE_CHECK 2021 /* no longer used */
#define DSM_RC_CPIC_PROGRAM_PARAM_CHECK 2021 /* no longer used */
#define DSM_RC_CPIC_UNSUCCESSFUL       2021 /* no longer used */
#define DSM_RC_UNKNOWN_CPIC_PROBLEM   2021 /* no longer used */
#define DSM_RC_CPIC_MISSING_LU        2021 /* no longer used */
#define DSM_RC_CPIC_MISSING_TP        2021 /* no longer used */
#define DSM_RC_CPIC_SNA6000_LOAD_FAIL 2021 /* no longer used */
#define DSM_RC_CPIC_STARTUP_FAILURE    2021 /* no longer used */

/*-----*/
/* Return codes -300 to -307 are reserved for IPX/SPX communications */
/*-----*/
#define DSM_RC_TLI_ERROR            2021 /* no longer used */
#define DSM_RC_IPXSPX_FAILURE       2021 /* no longer used */
#define DSM_RC_TLI_DLL_MISSING      2021 /* no longer used */
#define DSM_RC_DLL_LOADFAILURE      2021 /* no longer used */
#define DSM_RC_DLL_FUNCTION_LOADFAILURE 2021 /* no longer used */
#define DSM_RC_IPXCONN_REFUSED      2021 /* no longer used */
#define DSM_RC_IPXCONN_TIMEDOUT     2021 /* no longer used */
#define DSM_RC_IPXADDR_UNREACHABLE  2021 /* no longer used */
#define DSM_RC_CPIC_MISSING_DLL     2021 /* no longer used */
#define DSM_RC_CPIC_DLL_LOADFAILURE 2021 /* no longer used */
#define DSM_RC_CPIC_FUNC_LOADFAILURE 2021 /* no longer used */

/*=== Shared Memory Protocol error codes ===*/
#define DSM_RC_SHM_TCPIP_FAILURE    -450
#define DSM_RC_SHM_FAILURE          -451
#define DSM_RC_SHM_NOTAUTH          -452

#define DSM_RC_NULL_OBJNAME         2000 /* Object name pointer is NULL */
#define DSM_RC_NULL_DATAblkPTR      2001 /* dataBlkPtr is NULL */
#define DSM_RC_NULL_MSG              2002 /* msg parm in dsmRCMsg is NULL */

#define DSM_RC_NULL_OBJATTRPTR      2004 /* Object Attr Pointer is NULL */

```



```

#define DSM_RC_NO_SESS_BLK          2006 /* no server session info          */
#define DSM_RC_NO_POLICY_BLK        2007 /* no policy hdr info          */
#define DSM_RC_ZERO_BUFLEN          2008 /* bufferLen is zero for dataBlkPtr */
#define DSM_RC_NULL_BUFPTR          2009 /* bufferPtr is NULL for dataBlkPtr */

#define DSM_RC_INVALID_OBJTYPE      2010 /* invalid object type          */
#define DSM_RC_INVALID_VOTE         2011 /* invalid vote                 */
#define DSM_RC_INVALID_ACTION       2012 /* invalid action               */
#define DSM_RC_INVALID_DS_HANDLE    2014 /* invalid ADSM handle          */
#define DSM_RC_INVALID_REPOS        2015 /* invalid value for repository */
#define DSM_RC_INVALID_FSNAME       2016 /* fs should start with dir delim */
#define DSM_RC_INVALID_OBJNAME      2017 /* invalid full path name       */
#define DSM_RC_INVALID_LLNAME       2018 /* ll should start with dir delim */
#define DSM_RC_INVALID_OBJOWNER     2019 /* invalid object owner name    */
#define DSM_RC_INVALID_ACTYPE        2020 /* invalid action type          */
#define DSM_RC_INVALID_RETCODE      2021 /* dsmRC in dsmRCMsg is invalid */
#define DSM_RC_INVALID_SENDTYPE     2022 /* invalid send type            */
#define DSM_RC_INVALID_PARAMETER    2023 /* invalid parameter            */
#define DSM_RC_INVALID_OBJSTATE     2024 /* active, inactive, or any match? */
#define DSM_RC_INVALID_MCNAME       2025 /* Mgmt class name not found    */
#define DSM_RC_INVALID_DRIVE_CHAR   2026 /* Drive letter is not alphabet */
#define DSM_RC_NULL_FSNAME          2027 /* Filespace name is NULL      */
#define DSM_RC_INVALID_HLNAME       2028 /* hl should start with dir delim */

#define DSM_RC_NUMOBJ_EXCEED        2029 /* BeginGetData num objs exceeded */

#define DSM_RC_NEWPW_REQD           2030 /* new password is required     */
#define DSM_RC_OLDPW_REQD           2031 /* old password is required     */
#define DSM_RC_NO_OWNER_REQD        2032 /* owner not allowed. Allow default */
#define DSM_RC_NO_NODE_REQD         2033 /* node not allowed w/ pw=generate */
#define DSM_RC_KEY_MISSING          2034 /* key file can't be found      */
#define DSM_RC_KEY_BAD              2035 /* content of key file is bad   */

#define DSM_RC_BAD_CALL_SEQUENCE    2041 /* Sequence of DSM calls not allowed*/
#define DSM_RC_INVALID_TSMBUFFER    2042 /* invalid value for tsmbuffhandle or dataPtr */
#define DSM_RC_TOO_MANY_BYTES       2043 /* too many bytes copied to buffer */
#define DSM_RC_MUST_RELEASE_BUFFER  2044 /* cant exit app needs to release buffers */
#define DSM_RC_BUFF_ARRAY_ERROR      2045 /* internal buff array error     */
#define DSM_RC_INVALID_DATABLK      2046 /* using tsmbuff datablk should be null */
#define DSM_RC_ENCR_NOT_ALLOWED      2047 /* when using tsmbuffers encryption not allowed */
#define DSM_RC_OBJ_COMPRESSED        2048 /* Can't restore using tsmBuff on compressed object */
#define DSM_RC_OBJ_ENCRYPTED          2049 /* Cant restore using tsmbuff an encr obj */
#define DSM_RC_WILDCHAR_NOTALLOWED  2050 /* Wild card not allowed for hl,ll */
#define DSM_RC_POR_NOT_ALLOWED       2051 /* Can't use partial object restore with tsmBuffers */
#define DSM_RC_NO_ENCRYPTION_KEY     2052 /* Encryption key not found*/
#define DSM_RC_ENCR_CONFLICT         2053 /* mutually exclusive options */

#define DSM_RC_FSNAME_NOTFOUND       2060 /* Filespace name not found     */
#define DSM_RC_FS_NOT_REGISTERED     2061 /* Filespace name not registered */
#define DSM_RC_FS_ALREADY_REGED      2062 /* Filespace already registered */
#define DSM_RC_OBJID_NOTFOUND        2063 /* No object id to restore      */
#define DSM_RC_WRONG_VERSION         2064 /* Wrong level of code          */
#define DSM_RC_WRONG_VERSION_PARM    2065 /* Wrong level of parameter struct */

#define DSM_RC_NEEDTO_ENDTXN         2070 /* Need to call dsmEndTxn       */

#define DSM_RC_OBJ_EXCLUDED          2080 /* Object is excluded by MC     */
#define DSM_RC_OBJ_NOBCG             2081 /* Object has no backup copy group */
#define DSM_RC_OBJ_NOACG             2082 /* Object has no archive copy group */

#define DSM_RC_APISYSTEM_ERROR       2090 /* API internal error           */

#define DSM_RC_DESC_TOOLONG          2100 /* description is too long      */
#define DSM_RC_OBJINFO_TOOLONG       2101 /* object attr objinfo too long */
#define DSM_RC_HL_TOOLONG            2102 /* High level qualifier is too long */
#define DSM_RC_PASSWD_TOOLONG        2103 /* password is too long         */
#define DSM_RC_FILESPACE_TOOLONG     2104 /* filespace name is too long    */
#define DSM_RC_LL_TOOLONG            2105 /* Low level qualifier is too long */
#define DSM_RC_FSINFO_TOOLONG        2106 /* filespace length is too big   */
#define DSM_RC_SENDDATA_WITH_ZERO_SIZE 2107 /* send data w/ zero est        */

/*=== new return codes for dsmaccess ===*/
#define DSM_RC_INVALID_ACCESS_TYPE  2110 /* invalid access type          */
#define DSM_RC_QUERY_COMM_FAILURE    2111 /* communication error during query */
#define DSM_RC_NO_FILES_BACKUP       2112 /* No backed up files for this fs */
#define DSM_RC_NO_FILES_ARCHIVE      2113 /* No archived files for this fs */
#define DSM_RC_INVALID_SETACCESS     2114 /* invalid set access format     */

/*=== new return codes for dsmaccess ===*/
#define DSM_RC_STRING_TOO_LONG       2120 /* String parameter too long     */

#define DSM_RC_MORE_DATA              2200 /* There are more data to restore */

```

```

#define DSM_RC_BUFF_TOO_SMALL      2210 /* DataBlk buffer too small for qry */

#define DSM_RC_NO_API_CONFIGFILE    2228 /*specified API cfg file not found*/
#define DSM_RC_NO_INCLEXCL_FILE     2229 /* specified inclexcl file not found*/
#define DSM_RC_NO_SYS_OR_INCLEXCL   2230 /* either dsm.sys or inclexcl file
                                         specified in dsm.sys not found */
#define DSM_RC_REJECT_NO_POR_SUPPORT 2231 /* server doesn't have POR support*/

#define DSM_RC_NEED_ROOT            2300 /* API caller must be root */
#define DSM_RC_NEEDTO_CALL_BINDMC   2301 /* dsmBindMC must be called first */
#define DSM_RC_CHECK_REASON_CODE    2302 /* check reason code from dsmEndTxn */
#define DSM_RC_NEEDTO_ENDTXN_DEDUP_SIZE_EXCEEDED 2303 /* max dedup bytes exceeded */

/**** return codes 2400 - 2410 used by lic file see agentrc.h ****/

/**** return codes 2410 - 2430 used by Oracle agent see agentrc.h ****/

#define DSM_RC_ENC_WRONG_KEY        4580 /* the key provided is incorrect */
#define DSM_RC_ENC_NOT_AUTHORIZED    4582 /* user is not allowed to decrypt */
#define DSM_RC_ENC_TYPE_UNKNOWN      4584 /* encryption type unknown */

/*****
Return codes (4600)-(4624) are reserved for clustering
*****/
#define DSM_RC_CLUSTER_INFO_LIBRARY_NOT_LOADED 4600
#define DSM_RC_CLUSTER_LIBRARY_INVALID         4601
#define DSM_RC_CLUSTER_LIBRARY_NOT_LOADED      4602
#define DSM_RC_CLUSTER_NOT_MEMBER_OF_CLUSTER   4603
#define DSM_RC_CLUSTER_NOT_ENABLED             4604
#define DSM_RC_CLUSTER_NOT_SUPPORTED            4605
#define DSM_RC_CLUSTER_UNKNOWN_ERROR           4606

/*****
Return codes (5200)-(5600) are reserved for new Server ABORT codes (dsmcomm.h)
*****/
#define DSM_RS_ABORT_CERTIFICATE_NOT_FOUND      5200

/*****
Return codes (5701)-(5749) are reserved for proxy
*****/
#define DSM_RC_PROXY_REJECT_NO_RESOURCES        5702
#define DSM_RC_PROXY_REJECT_DUPLICATE_ID        5705
#define DSM_RC_PROXY_REJECT_ID_IN_USE           5710
#define DSM_RC_PROXY_REJECT_INTERNAL_ERROR       5717
#define DSM_RC_PROXY_REJECT_NOT_AUTHORIZED       5722
#define DSM_RC_PROXY_INVALID_FROMNODE           5746
#define DSM_RC_PROXY_INVALID_SERVERFREE         5747
#define DSM_RC_PROXY_INVALID_CLUSTER            5748
#define DSM_RC_PROXY_INVALID_FUNCTION           5749

/*****
Return codes 5801 - 5849 are reserved for cryptography/security
*****/

#define DSM_RC_CRYPTO_ICC_ERROR                 5801
#define DSM_RC_CRYPTO_ICC_CANNOT_LOAD          5802
#define DSM_RC_SSL_NOT_SUPPORTED                5803
#define DSM_RC_SSL_INIT_FAILED                  5804
#define DSM_RC_SSL_KEYFILE_OPEN_FAILED          5805
#define DSM_RC_SSL_KEYFILE_BAD_PASSWORD        5806
#define DSM_RC_SSL_BAD_CERTIFICATE              5807

/*****
Return codes 6300 - 6399 are reserved for client-side deduplication
*****/
#define DSM_RC_DIGEST_VALIDATION_ERROR          6300 /* End-to-end digest validation err */
#define DSM_RC_DATA_FINGERPRINT_ERROR           6301 /* Failure in Rabin fingerprinting */
#define DSM_RC_DATA_DEDUP_ERROR                  6302 /* Error converting data into chunks */

#endif /* _H_DSMRC */

```

関連資料

[API 戻りコード](#)

付録 B API タイプ定義ソース・ファイル

この付録には、API のための構造定義、タイプ定義、および定数が記述されています。

- 最初のヘッダー・ファイル `dsmapitd.h` および `tmapitd.h` は、すべてのオペレーティング・システムに共通する定義を示しています。
- 2 番目のヘッダー・ファイル `dsmapips.h` は、特定のオペレーティング・システムに固有の定義の 1 つの例で、このケースは Windows プラットフォームです。
- 3 番目のヘッダー・ファイル `release.h` は、バージョンおよびリリース情報を含んでいます。

ここに示す情報には、API とともに配布されるファイルのポイント・イン・タイム・コピーが含まれています。最新バージョンの API 配布パッケージ内のファイルを参照してください。

dsmapitd.h

```
/*
 * IBM Spectrum Protect
 * API Client Component
 *
 * IBM Confidential
 * (IBM Confidential-Restricted when combined with the Aggregated OCO
 * source modules for this program)
 *
 * OCO Source Materials
 *
 * 5648-020 (C) Copyright IBM Corporation 1993, 2018
 */

/*
 * Header File Name: dsmapitd.h
 *
 * Environment:
 * ** This is a platform-independent source file **
 *
 * Design Notes: This file contains basic data types and constants
 * includable by all client source files. The constants
 * within this file should be set properly for the
 * particular machine and operating system on which the
 * client software is to be run.
 *
 * Platform specific definitions are included in dsmapips.h
 *
 * Descriptive-name: Definitions for IBM Spectrum Protect API constants
 */

#ifndef _H_DSMAPITD
#define _H_DSMAPITD

#include "dsmapips.h" /* Platform specific definitions*/
#include "release.h"

/*=== set the structure alignment to pack the structures ===*/
#if (_OPSYS_TYPE == DS_WINNT) && !defined(_WIN64)
#pragma pack(1)
#endif

#ifdef _MAC
/*
 * choices are:
 * http://developer.apple.com/documentation/DeveloperTools/Conceptual/PowerPCRuntime/Data/
 * chapter_2_section_3.html
 *
 * #pragma option align=<mode>
 * where <mode> is power, mac68k, natural, or packed.
 */
#pragma options align=packed
#endif

typedef char osChar_t;
```



```

#define DSM GROUPTYPE_RESERVED1 0x01 /* for future use */
#define DSM GROUPTYPE_PEER 0x02 /* Peer group */
#define DSM GROUPTYPE_RESERVED2 0x03 /* for future use */

/*-----+
| Definitions for "member type" field in tsmGroupHandlerIn_t |
+-----*/

#define DSM_MEMBERTYPE_LEADER 0x01 /* group leader */
#define DSM_MEMBERTYPE_MEMBER 0x02 /* group member */

/*-----+
| Definitions for "operation type" field in tsmGroupHandlerIn_t |
+-----*/
#define DSM_GROUP_ACTION_BEGIN 0x01
#define DSM_GROUP_ACTION_OPEN 0x02 /* create new group */
#define DSM_GROUP_ACTION_CLOSE 0x03 /* commit and save an open group */
#define DSM_GROUP_ACTION_ADD 0x04 /* Append to a group */
#define DSM_GROUP_ACTION_ASSIGNTO 0x05 /* Assign to a another group */
#define DSM_GROUP_ACTION_REMOVE 0x06 /* remove a member from a group */

/*-----+
| Values for copySer in DetailCG structures for Query Mgmt Class response |
+-----*/
#define Copy_Serial_Static 1 /*Copy Serialization Static */
#define Copy_Serial_Shared_Static 2 /*Copy Serialization Shared Static*/
#define Copy_Serial_Shared_Dynamic 3 /*Copy Serialization Shared Dynamic*/
#define Copy_Serial_Dynamic 4 /*Copy Serialization Dynamic */

/*-----+
| Values for copyMode in DetailCG structures for Query Mgmt Class response |
+-----*/
#define Copy_Mode_Modified 1 /*Copy Mode Modified */
#define Copy_Mode_Absolute 2 /*Copy Mode Absolute */

/*-----+
| Values for objState in qryBackupData structure |
+-----*/
#define DSM_ACTIVE 0x01 /* query only active objects */
#define DSM_INACTIVE 0x02 /* query only inactive objects */
#define DSM_ANY_MATCH 0xFF /* query all backup objects */

/*-----+
| Boundary values for dsmDate.year field in qryArchiveData structure |
+-----*/
#define DATE_MINUS_INFINITE 0x0000 /* lowest boundary */
#define DATE_PLUS_INFINITE 0xFFFF /* highest upper boundary */

/*-----+
| Bits masks for update action parameter on dsmUpdateFS() |
+-----*/
#define DSM_FSUPD_FSTYPE ((unsigned) 0x00000002)
#define DSM_FSUPD_FSINFO ((unsigned) 0x00000004)
#define DSM_FSUPD_BACKSTARTDATE ((unsigned) 0x00000008)
#define DSM_FSUPD_BACKCOMPLETEDATE ((unsigned) 0x00000010)
#define DSM_FSUPD_OCCUPANCY ((unsigned) 0x00000020)
#define DSM_FSUPD_CAPACITY ((unsigned) 0x00000040)
#define DSM_FSUPD_RESERVED1 ((unsigned) 0x00000100)

/*-----+
| Bits mask for backup update action parameter on dsmUpdateObj() |
+-----*/
#define DSM_BACKUPD_OWNER ((unsigned) 0x00000001)
#define DSM_BACKUPD_OBJINFO ((unsigned) 0x00000002)
#define DSM_BACKUPD_MC ((unsigned) 0x00000004)

#define DSM_ARCHUPD_OWNER ((unsigned) 0x00000001)
#define DSM_ARCHUPD_OBJINFO ((unsigned) 0x00000002)
#define DSM_ARCHUPD_DESCR ((unsigned) 0x00000004)

/*-----+
| Values for repository parameter on dsmDeleteFS() |
+-----*/
#define DSM_ARCHIVE_REP 0x0A /* archive repository */
#define DSM_BACKUP_REP 0x0B /* backup repository */
#define DSM_REPOS_ALL 0x01 /* all repository types */

/*-----+
| Values for vote parameter on dsmEndTxn() |
+-----*/
#define DSM_VOTE_COMMIT 1 /* commit current transaction */
#define DSM_VOTE_ABORT 2 /* roll back current transaction */

```

```

/*-----+
| Values for various flags returned in ApiSessInfo structure.          |
+-----*/
/* Client compression field codes */
#define COMPRESS_YES    1    /* client must compress data      */
#define COMPRESS_NO     2    /* client must NOT compress data   */
#define COMPRESS_CD     3    /* client determined               */

/* Archive delete permission codes. */
#define ARCHDEL_YES     1    /* archive delete allowed          */
#define ARCHDEL_NO      2    /* archive delete NOT allowed      */

/* Backup delete permission codes. */
#define BACKDEL_YES     1    /* backup delete allowed           */
#define BACKDEL_NO      2    /* backup delete NOT allowed       */

/*-----+
| Values for various flags returned in optStruct structure.          |
+-----*/
#define DSM_PASSWD_GENERATE 1
#define DSM_PASSWD_PROMPT  0

#define DSM_COMM_TCP        1    /* tcpip                */
#define DSM_COMM_NAMEDPIPE  2    /* Named pipes          */
#define DSM_COMM_SHM        3    /* Shared Memory        */

/* obsolete commmethods */
#define DSM_COMM_PVM_IUCV    12
#define DSM_COMM_3270        12
#define DSM_COMM_IUCV        12
#define DSM_COMM_PWSCS       12
#define DSM_COMM_SNA_LU6_2   12
#define DSM_COMM_IPXSPX      12    /* For IPX/SPX support */
#define DSM_COMM_NETBIOS     12    /* NETBIOS             */
#define DSM_COMM_400COMM     12
#define DSM_COMM_CLIO        12    /* CLIO/S              */

/*-----+
| Values for userNameAuthorities in dsmInitEx for future use        |
+-----*/
#define DSM_USERAUTH_NONE    ((dsInt16_t)0x0000)
#define DSM_USERAUTH_ACCESS  ((dsInt16_t)0x0001)
#define DSM_USERAUTH_OWNER   ((dsInt16_t)0x0002)
#define DSM_USERAUTH_POLICY   ((dsInt16_t)0x0004)
#define DSM_USERAUTH_SYSTEM   ((dsInt16_t)0x0008)

/*-----+
| Values for encryptionType on dsmEndSendObjEx, queryResp          |
+-----*/
#define DSM_ENCRYPT_NO        ((dsUInt8_t)0x00)
#define DSM_ENCRYPT_USER      ((dsUInt8_t)0x01)
#define DSM_ENCRYPT_CLIENTENCRKEY ((dsUInt8_t)0x02)
#define DSM_ENCRYPT_DES_56BIT  ((dsUInt8_t)0x04)
#define DSM_ENCRYPT_AES_128BIT ((dsUInt8_t)0x08)
#define DSM_ENCRYPT_AES_256BIT ((dsUInt8_t)0x10)

/*-----+
| Definitions for mediaClass field.                                  |
+-----*/
/*
 * The following constants define a hierarchy of media access classes.
 * Lower numbers indicate media which can supply faster access to data.
 */

/* Fixed: represents the class of on-line, fixed media (such as
   hard disks). */
#define MEDIA_FIXED          0x10

/* Library: represents the class of mountable media accessible
   through a mechanical mounting device. */
#define MEDIA_LIBRARY        0x20

/* future use */
#define MEDIA_NETWORK        0x30

/* future use */
#define MEDIA_SHELF          0x40

/* future use */
#define MEDIA_OFFSITE        0x50

```

```

/* future use */
#define MEDIA_UNAVAILABLE    0xF0

/*-----+
| Type definition for partial object data for dsmBeginGetData() |
+-----*/
typedef struct
{
    dsUInt16_t    stVersion;        /* Structure version */
    dsStruct64_t  partialObjOffset; /* offset into object to begin reading */
    dsStruct64_t  partialObjLength; /* amount of object to read */
} PartialObjData ;                /* partial object data */

#define PartialObjDataVersion 1 /*

/*-----+
| Type definition for date structure |
+-----*/
typedef struct
{
    dsUInt16_t    year;              /* year, 16-bit integer (e.g., 1990) */
    dsUInt8_t     month;             /* month, 8-bit integer (1 - 12) */
    dsUInt8_t     day;               /* day, 8-bit integer (1 - 31) */
    dsUInt8_t     hour;              /* hour, 8-bit integer (0 - 23) */
    dsUInt8_t     minute;            /* minute, 8-bit integer (0 - 59) */
    dsUInt8_t     second;            /* second, b-bit integer (0 - 59) */
} dsmDate ;

/*-----+
| Type definition for Object ID on dsmGetObj() and in dsmGetList structure |
+-----*/
typedef dsStruct64_t  ObjID ;

/*-----+
| Type definition for dsmQueryBuff on dsmBeginQuery() |
+-----*/
typedef void dsmQueryBuff ;

/*-----+
| Type definition for dsmGetType parameter on dsmBeginGetData() |
+-----*/
typedef enum
{
    gtBackup = 0x00,                /* Backup processing type */
    gtArchive                /* Archive processing type */
} dsmGetType ;

/*-----+
| Type definition for dsmQueryType parameter on dsmBeginQuery() |
+-----*/
typedef enum
{
    qtArchive = 0x00,                /* Archive query type */
    qtBackup,                        /* Backup query type */
    qtBackupActive,                  /* Fast query for active backup files */
    qtFilespace,                    /* Filespace query type */
    qtMC,                           /* Mgmt. class query type */
    qtReserved1,                    /* future use */
    qtReserved2,                    /* future use */
    qtReserved3,                    /* future use */
    qtReserved4,                    /* future use */
    qtBackupGroups,                 /* group leaders in a specific fs */
    qtOpenGroups,                   /* Open groups in a specific fs */
    qtReserved5,                    /* future use */
    qtProxyNodeAuth,                /* nodes that his node can proxy to */
    qtProxyNodePeer,                /* Peer nodes with the same target */
    qtReserved6,                    /* future use */
    qtReserved7,                    /* future use */
    qtReserved8,                    /* future use */
} dsmQueryType ;

/*-----+
| Type definition sendType parameter on dsmBindMC() and dsmSendObj() |
+-----*/
typedef enum
{
    stBackup = 0x00,                /* Backup processing type */
    stArchive,                      /* Archive processing type */
    stBackupMountWait,              /* Backup processing with mountwait on */
    stArchiveMountWait             /* Archive processing with mountwait on */
} dsmSendType ;

```



```

/*-----+
| Type definition for delType parameter on dsmDeleteObj() |
+-----*/
typedef enum
{
    dtArchive = 0x00,                /* Archive delete type */
    dtBackup,                        /* Backup delete (deactivate) type */
    dtBackupID,                      /* Backup delete (remove) type */
} dsmDelType ;

/*-----+
| Type definition sendType parameter on dsmSetAccess() |
+-----*/
typedef enum
{
    atBackup = 0x00,                /* Backup processing type */
    atArchive,                      /* Archive processing type */
} dsmAccessType;

/*-----+
| Type definition for API Version on dsmInit() and dsmQueryApiVersion() |
+-----*/
typedef struct
{
    dsUint16_t version;             /* API version */
    dsUint16_t release;            /* API release */
    dsUint16_t level;              /* API level */
} dsmApiVersion;

/*-----+
| Type definition for API Version on dsmInit() and dsmQueryApiVersion() |
+-----*/
typedef struct
{
    dsUint16_t stVersion;           /* Structure version */
    dsUint16_t version;            /* API version */
    dsUint16_t release;            /* API release */
    dsUint16_t level;             /* API level */
    dsUint16_t subLevel;          /* API sub level */
    dsmBool_t unicode;            /* API unicode? */
} dsmApiVersionEx;

#define apiVersionExVer    2

/*-----+
| Type definition for Application Version on dsmInit() |
+-----*/
typedef struct
{
    dsUint16_t stVersion;           /* Structure version */
    dsUint16_t applicationVersion; /* application version number */
    dsUint16_t applicationRelease; /* application release number */
    dsUint16_t applicationLevel;   /* application level number */
    dsUint16_t applicationSubLevel; /* application sub level number */
} dsmAppVersion;

#define appVersionVer    1

/*-----+
| Type definition for object name used on BindMC, Send, Delete, Query |
+-----*/
typedef struct S_dsmObjName
{
    char fs[DSM_MAX_FSNAME_LENGTH + 1]; /* Filespace name */
    char hl[DSM_MAX_HL_LENGTH + 1];     /* High level name */
    char ll[DSM_MAX_LL_LENGTH + 1];     /* Low level name */
    dsUint8_t objType;                  /* for object type values, see defines above */
} dsmObjName;

/*-----+
| Type definition for Backup delete info on dsmDeleteObj() |
+-----*/
typedef struct
{
    dsUint16_t stVersion;             /* structure version */
    dsmObjName *objNameP;             /* object name */
    dsUint32_t copyGroup;             /* copy group */
} delBack ;

```

```

#define delBackVersion 1

/*-----+
| Type definition for Archive delete info on dsmDeleteObj() |
+-----*/
typedef struct
{
    dsUInt16_t      stVersion ;          /* structure version */
    dsStruct64_t     objId ;             /* object ID */
}delArch ;

#define delArchVersion 1

/*-----+
| Type definition for Backup ID delete info on dsmDeleteObj() |
+-----*/
typedef struct
{
    dsUInt16_t      stVersion ;          /* structure version */
    dsStruct64_t     objId ;             /* object ID */
}delBackID;

#define delBackIDVersion 1

/*-----+
| Type definition for delete info on dsmDeleteObj() |
+-----*/
typedef union
{
    delBack    backInfo ;
    delArch    archInfo ;
    delBackID  backIDInfo ;
}dsmDelInfo ;

/*-----+
| Type definition for Object Attribute parameter on dsmSendObj() |
+-----*/
typedef struct
{
    dsUInt16_t      stVersion;          /* Structure version */
    char            owner[DSM_MAX_OWNER_LENGTH + 1]; /* object owner */
    dsStruct64_t     sizeEstimate;       /* Size estimate in bytes of the object */
    dsmBool_t        objCompressed;      /* Is object already compressed? */
    dsUInt16_t        objInfoLength;     /* length of object-dependent info */
    char            *objInfo;           /* object-dependent info */
    char            *mcNameP;           /* mgmnt class name for override */
    dsmBool_t        disableDeduplication; /* force no dedup for this object */
    dsmBool_t        useExtObjInfo;      /* use ext obj info up to 1536 */
}ObjAttr;

#define ObjAttrVersion 4

/*-----+
| Type definition for mcBindKey returned on dsmBindMC() |
+-----*/
typedef struct
{
    dsUInt16_t      stVersion;          /* structure version */
    char            mcName[DSM_MAX_MC_NAME_LENGTH + 1]; /* Name of mc bound to object. */
                                                    /* True/false */
    dsmBool_t        backup_cg_exists;   /* True/false */
    dsmBool_t        archive_cg_exists;  /* True/false */
    char            backup_copy_dest[DSM_MAX_CG_DEST_LENGTH + 1]; /* Backup copy dest. name */
                                                    /* Arch copy dest.name */
    char            archive_copy_dest[DSM_MAX_CG_DEST_LENGTH + 1];
}mcBindKey;

#define mcBindKeyVersion 1

/*-----+
| Type definition for object list on dsmBeginGetData() |
+-----*/
typedef struct
{
    dsUInt16_t      stVersion ;          /* structure version */
    dsUInt32_t       numObjId ;          /* number of object IDs in the list */
    ObjID           *objId ;            /* list of object IDs to restore*/
    PartialObjData  *partialObjData;     /*list of partial obj data info */

```

```

}dsmGetList ;

#define dsmGetListVersion      2 /* default if not using Partial Obj data */
#define dsmGetListPORVersion   3 /* version if using Partial Obj data */

/*-----+
| Type definition for DataBlk used to Get or Send data |
+-----*/
typedef struct
{
    dsUint16_t stVersion ; /* structure version */
    dsUint32_t bufferLen; /* Length of buffer passed below */
    dsUint32_t numBytes; /* Actual number of bytes read from */
                        /* or written to the buffer */
    char *bufferPtr; /* Data buffer */
    dsUint32_t numBytesCompressed; /* on send actual bytes compressed */
    dsUint16_t reserved; /* for future use */
}DataBlk;

#define DataBlkVersion 3

/*-----+
| Type definition for Mgmt Class queryBuffer on dsmBeginQuery() |
+-----*/
typedef struct S_qryMCData
{
    dsUint16_t stVersion; /* structure version */
    char *mcName; /* Mgmt class name */
    dsmBool_t mcDetail; /* single name to get one or empty string to get all*/
                        /* Want details or not? */
}qryMCData;

#define qryMCDataVersion 1

/*=== values for RETINIT ===*/
#define ARCH_RETINIT_CREATE 0
#define ARCH_RETINIT_EVENT 1

/*-----+
| Type definition for Archive Copy Group details on Query MC response |
+-----*/
typedef struct S_archDetailCG
{
    char cgName[DSM_MAX_CG_NAME_LENGTH + 1]; /* Copy group name */
    dsUint16_t frequency; /* Copy (archive) frequency */
    dsUint16_t retainVers; /* Retain version */
    dsUint8_t copySer; /* for copy serialization values, see defines */
    dsUint8_t copyMode; /* for copy mode values, see defines above */
    char destName[DSM_MAX_CG_DEST_LENGTH + 1]; /* Copy dest name */
    dsmBool_t bLanFreeDest; /* Destination has lan free path? */
    dsmBool_t reserved; /* Not currently used */
    dsUint8_t retainInit; /* possible values see above */
    dsUint16_t retainMin; /* if retInit is EVENT num of days */
    dsmBool_t bDeduplicate; /* destination has dedup enabled */
}archDetailCG;

/*-----+
| Type definition for Backup Copy Group details on Query MC response |
+-----*/
typedef struct S_backupDetailCG
{
    char cgName[DSM_MAX_CG_NAME_LENGTH + 1]; /* Copy group name */
    dsUint16_t frequency; /* Backup frequency */
    dsUint16_t verDataExst; /* Versions data exists */
    dsUint16_t verDataDltd; /* Versions data deleted */
    dsUint16_t retXtraVers; /* Retain extra versions */
    dsUint16_t retOnlyVers; /* Retain only versions */
    dsUint8_t copySer; /* for copy serialization values, see defines */
    dsUint8_t copyMode; /* for copy mode values, see defines above */
    char destName[DSM_MAX_CG_DEST_LENGTH + 1]; /* Copy dest name */
    dsmBool_t bLanFreeDest; /* Destination has lan free path? */
    dsmBool_t reserved; /* Not currently used */
    dsmBool_t bDeduplicate; /* destination has dedup enabled */
}backupDetailCG;

/*-----+
| Type definition for Query Mgmt Class detail response on dsmGetNextQObj() |
+-----*/
typedef struct S_qryRespMCDetailData
{

```

```

    dsUInt16_t    stVersion; /* structure version */
    char          mName[DSM_MAX_MC_NAME_LENGTH + 1]; /* mc name */
    char          mDesc[DSM_MAX_MC_DESCR_LENGTH + 1]; /* mc description */
    archDetailCG  archDet; /* Archive copy group detail */
    backupDetailCG backupDet; /* Backup copy group detail */
}qryRespMCDetailData;

#define qryRespMCDetailDataVersion 4

/*-----+
| Type definition for Query Mgmt Class summary response on dsmGetNextQObj()|
+-----*/
typedef struct S_qryRespMCData
{
    dsUInt16_t    stVersion; /* structure version */
    char          mName[DSM_MAX_MC_NAME_LENGTH + 1]; /* mc name */
    char          mDesc[DSM_MAX_MC_DESCR_LENGTH + 1]; /* mc description */
}qryRespMCData;

#define qryRespMCDataVersion 1

/*-----+
| Type definition for Archive queryBuffer on dsmBeginQuery()|
+-----*/
typedef struct S_qryArchiveData
{
    dsUInt16_t    stVersion; /* structure version */
    dsmObjName    *objName; /* Full dsm name of object */
    char          *owner; /* owner name */
    /* for maximum date boundaries, see defines above */
    dsmDate       insDateLowerBound; /* low bound archive insert date */
    dsmDate       insDateUpperBound; /* hi bound archive insert date */
    dsmDate       expDateLowerBound; /* low bound expiration date */
    dsmDate       expDateUpperBound; /* hi bound expiration date */
    char          *descr; /* archive description */
} qryArchiveData;

#define qryArchiveDataVersion 1

/*=== values for retentionInitiated field ===*/
#define DSM_ARCH_RETINIT_UNKNOWN 0 /* ret init is unknown (down-level srv) */
#define DSM_ARCH_RETINIT_STARTED 1 /* retention clock is started */
#define DSM_ARCH_RETINIT_PENDING 2 /* retention clock is not started */

/*=== Values for objHeld ===*/
#define DSM_ARCH_HELD_UNKNOWN 0 /* unknown hold status (down-level srv) */
#define DSM_ARCH_HELD_FALSE 1 /* object is NOT in a delete hold state */
#define DSM_ARCH_HELD_TRUE 2 /* object is in a delete hold state */

/*-----+
| Type definition for Query Archive response on dsmGetNextQObj()|
+-----*/
typedef struct S_qryRespArchiveData
{
    dsUInt16_t    stVersion; /* structure version */
    dsmObjName    objName; /* Filespace name qualifier */
    dsUInt32_t    copyGroup; /* copy group number */
    char          mName[DSM_MAX_MC_NAME_LENGTH + 1]; /* mc name */
    char          owner[DSM_MAX_OWNER_LENGTH + 1]; /* owner name */
    dsStruct64_t  objId; /* Unique copy id */
    dsStruct64_t  reserved; /* backward compatability */
    dsUInt8_t     mediaClass; /* media access class */
    dsmDate       insDate; /* archive insertion date */
    dsmDate       expDate; /* expiration date for object */
    char          descr[DSM_MAX_DESCR_LENGTH + 1]; /* archive description */
    dsUInt16_t    objInfolen; /* length of object-dependent info */
    char          reservedObjInfo[DSM_MAX_OBGINFO_LENGTH]; /*object-dependent info */
    dsUInt160_t   restoreOrderExt; /* restore order */
    dsStruct64_t  sizeEstimate; /* size estimate stored by user */
    dsUInt8_t     compressType; /* Compression flag */
    dsUInt8_t     retentionInitiated; /* object waiting on retention event */
    dsUInt8_t     objHeld; /*object is on retention "hold" see values above*/
    dsUInt8_t     encryptionType; /* type of encryption */
    dsmBool_t     clientDeduplicated; /* obj deduplicated by API */
    char          objInfo[DSM_MAX_EXT_OBGINFO_LENGTH]; /*object-dependent info */
    char          compressAlg[DSM_MAX_COMPRESSTYPE_LENGTH + 1]; /* compression algorithm name */
}qryRespArchiveData;

```

```

#define qryRespArchiveDataVersion 7

/*-----+
| Type definition for Archive sendBuff parameter on dsmSendObj()
+-----*/
typedef struct S_sndArchiveData
{
    dsUint16_t    stVersion;           /* structure version */
    char          *descr;              /* archive description */
}sndArchiveData;

#define sndArchiveDataVersion 1

/*-----+
| Type definition for Backup queryBuffer on dsmBeginQuery()
+-----*/
typedef struct S_qryBackupData
{
    dsUint16_t    stVersion;           /* structure version */
    dsmObjName    *objName;            /* full dsm name of object */
    char          *owner;              /* owner name */
    dsUint8_t     objState;            /* object state selector */
    dsmDate       pitDate;             /* Date value for point in time restore */
                                           /* for possible values, see defines above */
}qryBackupData;

#define qryBackupDataVersion 2

typedef struct
{
    dsUint8_t     reserved1;
    dsStruct64_t  reserved2;
} reservedInfo_t; /* for future use */

/*-----+
| Type definition for Query Backup response on dsmGetNextQObj()
+-----*/
typedef struct S_qryRespBackupData
{
    dsUint16_t    stVersion;           /* structure version */
    dsmObjName    objName;             /* full dsm name of object */
    dsUint32_t    copyGroup;           /* copy group number */
    char          mcName[DSM_MAX_MC_NAME_LENGTH + 1]; /* mc name */
    char          owner[DSM_MAX_OWNER_LENGTH + 1];    /* owner name */
    dsStruct64_t  objId;               /* Unique object id */
    dsStruct64_t  reserved;            /* backward compatability */
    dsUint8_t     mediaClass;          /* media access class */
    dsUint8_t     objState;            /* Obj state, active, etc. */
    dsmDate       insDate;             /* backup insertion date */
    dsmDate       expDate;            /* expiration date for object */
    dsUint16_t    objInfolen;          /* length of object-dependent info */
    char          reservedObjInfo[DSM_MAX_OBJINFO_LENGTH]; /*object-dependent info */
    dsUint160_t   restoreOrderExt;     /* restore order */
    dsStruct64_t  sizeEstimate;        /* size estimate stored by user */
    dsStruct64_t  baseObjId;
    dsUint16_t    baseObjInfolen;      /* length of base object-dependent info */
    dsUint8_t     baseObjInfo[DSM_MAX_OBJINFO_LENGTH]; /* base object-dependent info */
    dsUint160_t   baseRestoreOrder;    /* restore order */
    dsUint32_t    fsID;
    dsUint8_t     compressType;
    dsmBool_t     isGroupLeader;
    dsmBool_t     isOpenGroup;
    dsUint8_t     reserved1;           /* for future use */
    dsmBool_t     reserved2;           /* for future use */
    dsUint16_t    reserved3;           /* for future use */
    reservedInfo_t *reserved4;         /* for future use */
    dsUint8_t     encryptionType;      /* type of encryption */
    dsmBool_t     clientDeduplicated;  /* obj deduplicated by API */
    char          objInfo[DSM_MAX_EXT_OBJINFO_LENGTH]; /*object-dependent info */
    char          compressAlg[DSM_MAX_COMPRESSTYPE_LENGTH + 1]; /* compression algorithm name */
}qryRespBackupData;

#define qryRespBackupDataVersion 8

/*-----+
| Type definition for Active Backup queryBuffer on dsmBeginQuery()
|
| Notes: For the active backup query, only the fs (filesystem) and objType
|        fields of objName need be set.  objType can only be set to
|        DSM_OBJ_FILE or DSM_OBJ_DIRECTORY.  DSM_OBJ_ANY_TYPE will not
|        find a match on the query.
|
+-----*/

```

```

+-----*/
typedef struct S_qryABackupData
{
    dsUuint16_t    stVersion;                /* structure version */
    dsmObjName     *objName;                /* Only fs and objtype used */
}qryABackupData;

#define qryABackupDataVersion 1

/*-----+
| Type definition for Query Active Backup response on dsmGetNextQObj() |
+-----*/
typedef struct S_qryARespBackupData
{
    dsUuint16_t    stVersion;                /* structure version */
    dsmObjName     objName;                /* full dsm name of object */
    dsUuint32_t    copyGroup;              /* copy group number */
    char           mcName[DSM_MAX_MC_NAME_LENGTH + 1]; /*management class name*/
    char           owner[DSM_MAX_OWNER_LENGTH + 1];   /* owner name */
    dsmDate        insDate;                /* backup insertion date */
    dsUuint16_t    objInfolen;             /* length of object-dependent info*/
    char           reservedObjInfo[DSM_MAX_OBJINFO_LENGTH]; /*object-dependent info */
    char           objInfo[DSM_MAX_EXT_OBJINFO_LENGTH]; /*object-dependent info */
}qryARespBackupData;

#define qryARespBackupDataVersion 2

/*-----+
| Type definition for Backup queryBuffer on dsmBeginQuery() |
+-----*/
typedef struct qryBackupGroups
{
    dsUuint16_t    stVersion;                /* structure version */
    dsUuint8_t     groupType;
    char           *fsName;
    char           *owner;
    dsStruct64_t   groupLeaderObjId;
    dsUuint8_t     objType;
    dsmBool_t      noRestoreOrder;
    dsmBool_t      noGroupInfo;
    char           *hl;
}qryBackupGroups;

#define qryBackupGroupsVersion 3

/*-----+
| Type definition for proxynode queryBuffer on dsmBeginQuery() |
+-----*/
typedef struct qryProxyNodeData
{
    dsUuint16_t    stVersion;                /* structure version */
    char           *targetNodeName;         /* target node name */
}qryProxyNodeData;

#define qryProxyNodeDataVersion 1

/*-----+
| Type definition for qryRespProxyNodeData parameter used on dsmGetNextQObj() |
+-----*/
typedef struct
{
    dsUuint16_t    stVersion ;                /* structure version */
    char           targetNodeName[DSM_MAX_ID_LENGTH+1]; /* target node name */
    char           peerNodeName[DSM_MAX_ID_LENGTH+1]; /* Peer node name */
    char           hlAddress[DSM_MAX_ID_LENGTH+1]; /* peer hlAddress */
    char           llAddress[DSM_MAX_ID_LENGTH+1]; /* peer hlAddress */
}qryRespProxyNodeData;

#define qryRespProxyNodeDataVersion 1

/*-----+
| Type definition for WINNT and OS/2 Filespace attributes |
+-----*/
typedef struct
{
    char           driveLetter ;                /* drive letter for filesystem */
    dsUuint16_t    fsInfoLength;                /* fsInfo length used */
    char           fsInfo[DSM_MAX_FSINFO_LENGTH]; /*caller-determined data */
}dsmDosFSAttrib ;

```

```

/*-----+
| Type definition for UNIX Filespace attributes |
+-----*/
typedef struct
{
    dsUInt16_t    fsInfoLength;          /* fsInfo length used */
    char          fsInfo[DSM_MAX_FSINFO_LENGTH]; /* caller-determined data */
} dsmUnixFSAttr ;

/*-----+
| Type definition for NetWare Filespace attributes |
+-----*/
typedef dsmUnixFSAttr dsmNetwareFSAttr;

/*-----+
| Type definition for Filespace attributes on all Filespace calls |
+-----*/
typedef union
{
    dsmNetwareFSAttr    netwareFSAttr;
    dsmUnixFSAttr       unixFSAttr ;
    dsmDosFSAttr        dosFSAttr ;
} dsmFSAttr ;

/*-----+
| Type definition for fsUpd parameter on dsmUpdateFS() |
+-----*/
typedef struct S_dsmFSUpd
{
    dsUInt16_t    stVersion ;          /* structure version */
    char          *fsType ;           /* filespace type */
    dsStruct64_t  occupancy ;          /* occupancy estimate */
    dsStruct64_t  capacity ;           /* capacity estimate */
    dsmFSAttr     fsAttr ;             /* platform specific attributes */
} dsmFSUpd ;

#define dsmFSUpdVersion 1

/*-----+
| Type definition for Filespace queryBuffer on dsmBeginQuery() |
+-----*/
typedef struct S_qryFSData
{
    dsUInt16_t    stVersion;          /* structure version */
    char          *fsName;            /* File space name */
} qryFSData;

#define qryFSDataVersion 1

/*-----+
| Type definition for Query Filespace response on dsmGetNextQObj() |
+-----*/
typedef struct S_qryRespFSData
{
    dsUInt16_t    stVersion;          /* structure version */
    char          fsName[DSM_MAX_FSNAME_LENGTH + 1]; /* Filespace name */
    char          fsType[DSM_MAX_FSTYPE_LENGTH + 1]; /* Filespace type */
    dsStruct64_t  occupancy;           /* Occupancy est. in bytes */
    dsStruct64_t  capacity;            /* Capacity est. in bytes */
    dsmFSAttr     fsAttr ;             /* platform specific attributes */
    dsmDate       backStartDate;        /* start backup date */
    dsmDate       backCompleteDate;     /* end backup Date */
    dsmDate       reserved1;            /* For future use */
    dsmDate       lastReplStartDate;     /* The last time replication was started */
    dsmDate       lastReplCmpltDate;    /* The last time replication completed */
    /* (could have had a failure, */
    /* but it still completes) */
    dsmDate       lastBackOpDateFromServer; /* The last store time stamp the client */
    /* saved on the server */
    dsmDate       lastArchOpDateFromServer; /* The last store time stamp the client */
    /* saved on the server */
    dsmDate       lastSpMgOpDateFromServer; /* The last store time stamp the client */
    /* saved on the server */
    dsmDate       lastBackOpDateFromLocal; /* The last store time stamp the client */
    /* saved on the Local */
    dsmDate       lastArchOpDateFromLocal; /* The last store time stamp the client */
    /* saved on the Local */
    dsmDate       lastSpMgOpDateFromLocal; /* The last store time stamp the client */
    /* saved on the Local */
    dsInt32_t     failOverWriteDelay;   /* Minutes for client to wait before allowed */
    /* to store to this Repl srvr, Specail codes: */

```

```

/* NO_ACCESS(-1), ACCESS_RDONLY (-2) */
}qryRespFSData;

#define qryRespFSDataVersion 4

/*-----+
| Type definition for regFilespace parameter on dsmRegisterFS()
+-----*/
typedef struct S_regFSData
{
    dsUInt16_t    stVersion;                /* structure version */
    char          *fsName;                  /* Filespace name */
    char          *fsType;                  /* Filespace type */
    dsStruct64_t  occupancy;                /* Occupancy est. in bytes. */
    dsStruct64_t  capacity;                 /* Capacity est. in bytes. */
    dsmFSAttr     fsAttr;                  /* platform specific attributes */
}regFSData;

#define regFSDataVersion 1

/*-----+
| Type definition for dedupType used in apisessInfo
+-----*/
typedef enum
{
    dedupServerOnly= 0x00,                /* dedup only done on server */
    dedupClientOrServer                  /* dedup can be done on client or server */
}dsmDedupType ;

/*-----+
| Type definition for fail over configuration and status
+-----*/
typedef enum
{
    failOvrNotConfigured = 0x00,
    failOvrConfigured,
    failOvrConnectedToReplServer
}dsmFailOvrCfgType ;

/*-----+
| Type definition for session info response on dsmQuerySessionInfo()
+-----*/
typedef struct
{
    dsUInt16_t    stVersion;                /* Structure version */
    /*-----+
    /* Server information
    +-----*/
    char          serverHost[DSM_MAX_SERVERNAME_LENGTH+1];
    /* Network host name of DSM server */
    dsUInt16_t    serverPort;                /* Server comm port on host */
    dsmDate       serverDate;                /* Server's date/time */
    char          serverType[DSM_MAX_SERVERTYPE_LENGTH+1];
    /* Server's execution platform */
    dsUInt16_t    serverVer;                /* Server's version number */
    dsUInt16_t    serverRel;                /* Server's release number */
    dsUInt16_t    serverLev;                /* Server's level number */
    dsUInt16_t    serverSubLev;            /* Server's sublevel number */
    /*-----+
    /* Client Defaults
    +-----*/
    char          nodeType[DSM_MAX_PLATFORM_LENGTH+1]; /*node/application type*/
    char          fsdelim;                  /* File space delimiter */
    char          hldelim;                  /* Delimiter betw highlev & lowlev */
    dsUInt8_t     compression;              /* Compression flag */
    dsUInt8_t     archDel;                  /* Archive delete permission */
    dsUInt8_t     backDel;                  /* Backup delete permission */
    dsUInt32_t     maxBytesPerTxn;          /* for future use */
    dsUInt16_t     maxObjPerTxn;            /* The max objects allowed in a txn */
    /*-----+
    /* Session Information
    +-----*/
    char          id[DSM_MAX_ID_LENGTH+1]; /* Sign-in id node name */
    char          owner[DSM_MAX_OWNER_LENGTH+1]; /* Sign-in owner */
    /* (for multi-user platforms) */
    char          confFile[DSM_PATH_MAX + DSM_NAME_MAX +1];
    /* len is platform dep */
    /* dsInit name of appl config file */
    dsUInt8_t     opNoTrace;                /* dsInit option - NoTrace = 1 */
    /*-----+
    /* Policy Data
    +-----*/

```



```

char        domainName[DSM_MAX_DOMAIN_LENGTH+1]; /* Domain name          */
char        policySetName[DSM_MAX_PS_NAME_LENGTH+1];
            /* Active policy set name          */
dsmDate     polActDate; /* Policy set activation date      */
char        dfltMCName[DSM_MAX_MC_NAME_LENGTH+1]; /* Default Mgmt Class */
dsUint16_t  gpBackRetn; /* Grace-period backup retention  */
dsUint16_t  gpArchRetn; /* Grace-period archive retention */
char        adsmServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* adsm server name */
dsmBool_t   archiveRetentionProtection; /* is server Retention protection enabled */
dsStruct64_t maxBytesPerTxn_64; /* for future use */
dsmBool_t   lanFreeEnabled; /* lan free option is set */
dsmDedupType dedupType; /* server or clientOrServer */
char        accessNode[DSM_MAX_ID_LENGTH+1]; /* as node node name */

/*-----*/
/*          Replication and fail over information          */
/*-----*/
dsmFailOvrCfgType failOverCfgType; /* status of fail over */
char        replServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* repl server name */
char        homeServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* home server name */
char        replServerHost[DSM_MAX_SERVERNAME_LENGTH+1]; /* Network host name of DSM server */
dsInt32_t   replServerPort; /* Server comm port on host */

}ApiSessInfo;

#define ApiSessInfoVersion 6

/*-----+
| Type definition for Query options response on dsmQueryCliOptions()
| and dsmQuerySessOptions()
|-----*/

typedef struct
{
    char        dsmiDir[DSM_PATH_MAX + DSM_NAME_MAX +1];
    char        dsmiConfig[DSM_PATH_MAX + DSM_NAME_MAX +1];
    char        serverName[DSM_MAX_SERVERNAME_LENGTH+1];
    dsInt16_t   commMethod;
    char        serverAddress[DSM_MAX_SERVER_ADDRESS];
    char        nodeName[DSM_MAX_NODE_LENGTH+1];
    dsmBool_t   compression;
    dsmBool_t   compressalways;
    dsmBool_t   passwordAccess;
}optStruct;

/*-----+
| Type definition for LogType used in logInfo
|-----*/
typedef enum
{
    logServer = 0x00, /* log msg only to server */
    logLocal, /* log msg only to local error log */
    logBoth, /* log msg to server and to local error log */
    logNone
}dsmLogType;

/*-----+
| Type definition for logInfo parameter used on dsmLogEvent()
|-----*/

typedef struct
{
    char        *message; /* text of message to be logged */
    dsmLogType  logType; /* log type : local, server, both */
}logInfo;

/*-----+
| Type definition for qryRespAccessData parameter used on dsmQueryAccess()
|-----*/

typedef struct
{
    dsUint16_t   stVersion; /* structure version */
    char        node[DSM_MAX_ID_LENGTH+1]; /* node name */
    char        owner[DSM_MAX_OWNER_LENGTH+1]; /* owner */
    dsmObjName   objName; /* object name */
    dsmAccessType  accessType; /* archive or backup */
    dsUint32_t   ruleNumber; /* Access rule id */
}qryRespAccessData;

```

```

#define qryRespAccessDataVersion 1

/*-----+
| Type definition for envSetUp parameter on dsmSetUp()
+-----*/
typedef struct S_envSetUp
{
    dsUInt16_t      stVersion; /* structure version */
    char            dsmiDir[DSM_PATH_MAX + DSM_NAME_MAX +1];
    char            dsmiConfig[DSM_PATH_MAX + DSM_NAME_MAX +1];
    char            dsmiLog[DSM_PATH_MAX + DSM_NAME_MAX +1];
    char            **argv; /* for executables name argv[0] */
    char            logName[DSM_NAME_MAX +1];
    dsmBool_t       reserved1; /* for future use */
    dsmBool_t       reserved2; /* for future use */
}envSetUp;

#define envSetUpVersion 4

/*-----+
| Type definition for dsmInitExIn_t
+-----*/
typedef struct dsmInitExIn_t
{
    dsUInt16_t      stVersion; /* structure version */
    dsmApiVersionEx *apiVersionExP;
    char            *clientNodeNameP;
    char            *clientOwnerNameP;
    char            *clientPasswordP;
    char            *userNameP;
    char            *userPasswordP;
    char            *applicationTypeP;
    char            *configfile;
    char            *options;
    char            dirDelimiter;
    dsmBool_t       useUnicode;
    dsmBool_t       bCrossPlatform;
    dsmBool_t       bService;
    dsmBool_t       bEncryptKeyEnabled;
    char            *encryptionPasswordP;
    dsmBool_t       useTsmBuffers;
    dsUInt8_t       numTsmBuffers;
    dsmAppVersion   *appVersionP;
}dsmInitExIn_t;

#define dsmInitExInVersion 5

/*-----+
| Type definition for dsmInitExOut_t
+-----*/
typedef struct dsmInitExOut_t
{
    dsUInt16_t      stVersion; /* structure version */
    dsInt16_t       userNameAuthorities;
    dsInt16_t       infoRC; /* error return code if encountered */
    char            adsmServerName[DSM_MAX_SERVERNAME_LENGTH+1];
    dsUInt16_t      serverVer; /* Server's version number */
    dsUInt16_t      serverRel; /* Server's release number */
    dsUInt16_t      serverLev; /* Server's level number */
    dsUInt16_t      serverSubLev; /* Server's sublevel number */

    dsmBool_t       bIsFailOverMode; /* true if failover has occurred */
    char            replServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* repl server name */
    char            homeServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* home server name */
}dsmInitExOut_t;

#define dsmInitExOutVersion 3

/*-----+
| Type definition for LogType used in logInfo
+-----*/
typedef enum
{
    logSevInfo = 0x00, /* information ANE4991 */
    logSevWarning, /* warning ANE4992 */
    logSevError, /* Error ANE4993 */
    logSevSevere, /* severe ANE4994 */
    logSevLicense, /* License ANE4995 */
    logSevTryBuy /* try Buy ANE4996 */
}dsmLogSeverity ;

```

```

/*-----+
| Type definition for dsmLogExIn_t
+-----*/
typedef struct dsmLogExIn_t
{
    dsUuint16_t      stVersion; /* structure version */
    dsmLogSeverity   severity;
    char             appMsgID[8];
    dsmLogType       logType; /* log type : local, server, both */
    char             *message; /* text of message to be logged */
    char             appName[DSM_MAX_PLATFORM_LENGTH];
    char             osPlatform[DSM_MAX_PLATFORM_LENGTH];
    char             appVersion[DSM_MAX_PLATFORM_LENGTH];
}dsmLogExIn_t;

#define dsmLogExInVersion 2

/*-----+
| Type definition for dsmLogExOut_t
+-----*/
typedef struct dsmLogExOut_t
{
    dsUuint16_t      stVersion; /* structure version */
}dsmLogExOut_t;

#define dsmLogExOutVersion 1

/*-----+
| Type definition for dsmRenameIn_t
+-----*/
typedef struct dsmRenameIn_t
{
    dsUuint16_t      stVersion; /* structure version */
    dsUuint32_t      dsmHandle; /* handle for session */
    dsUuint8_t       repository; /* Backup or Archive */
    dsmObjNameP      *objNameP; /* object name */
    char             newHl[DSM_MAX_HL_LENGTH + 1]; /* new High level name */
    char             newLl[DSM_MAX_LL_LENGTH + 1]; /* new Low level name */
    dsmBool_t        merge; /* merge into existing name*/
    ObjID            objId; /* objId for Archive */
}dsmRenameIn_t;

#define dsmRenameInVersion 1

/*-----+
| Type definition for dsmRenameOut_t
+-----*/
typedef struct dsmRenameOut_t
{
    dsUuint16_t      stVersion; /* structure version */
}dsmRenameOut_t;

#define dsmRenameOutVersion 1

/*-----+
| Type definition for dsmEndSendObjExIn_t
+-----*/
typedef struct dsmEndSendObjExIn_t
{
    dsUuint16_t      stVersion; /* structure version */
    dsUuint32_t      dsmHandle; /* handle for session */
}dsmEndSendObjExIn_t;

#define dsmEndSendObjExInVersion 1

/*-----+
| Type definition for dsmEndSendObjExOut_t
+-----*/
typedef struct dsmEndSendObjExOut_t
{
    dsUuint16_t      stVersion; /* structure version */
    dsStruct64_t     totalBytesSent; /* total bytes read from app */
    dsmBool_t        objCompressed; /* was object compressed */
    dsStruct64_t     totalCompressSize; /* total size after compress */
    dsStruct64_t     totalLFBytesSent; /* total bytes sent Lan Free */
    dsUuint8_t       encryptionType; /* type of encryption used */
    dsmBool_t        objDeduplicated; /* was object processed for dist. data dedup */
    dsStruct64_t     totalDedupSize; /* total size after de-dup */
}dsmEndSendObjExOut_t;

#define dsmEndSendObjExOutVersion 3

```

```

/*-----+
| Type definition for dsmGroupHandlerIn_t
+-----*/
typedef struct dsmGroupHandlerIn_t
{
    dsUint16_t      stVersion;          /* structure version */
    dsUint32_t      dsmHandle;          /* handle for session */
    dsUint8_t       groupType;          /* Type of group */
    dsUint8_t       actionType;         /* Type of group operation */
    dsUint8_t       memberType;         /* Type of member: Leader or member */
    dsStruct64_t    leaderObjId;        /* OBJID of the groupleader when manipulating a member */
    char            *uniqueGroupTagP;   /* Unique group identifier */
    dsmObjName      *objNameP;         /* group leader object name */
    dsmGetList      memberObjList;      /* list of objects to remove, assign */
}dsmGroupHandlerIn_t;

#define dsmGroupHandlerInVersion 1

/*-----+
| Type definition for dsmGroupHandlerExOut_t
+-----*/
typedef struct dsmGroupHandlerOut_t
{
    dsUint16_t      stVersion;          /* structure version */
}dsmGroupHandlerOut_t;

#define dsmGroupHandlerOutVersion 1

/*-----+
| Type definition for dsmEndTxnExIn_t
+-----*/
typedef struct dsmEndTxnExIn_t
{
    dsUint16_t      stVersion;          /* structure version */
    dsUint32_t      dsmHandle;          /* handle for session */
    dsUint8_t       vote;
}dsmEndTxnExIn_t;

#define dsmEndTxnExInVersion 1

/*-----+
| Type definition for dsmEndTxnExOut_t
+-----*/
typedef struct dsmEndTxnExOut_t
{
    dsUint16_t      stVersion;          /* structure version */
    dsUint16_t      reason;             /* reason code */
    dsStruct64_t    groupLeaderObjId;   /* groupLeader obj id returned on */
                                         /* DSM_ACTION_OPEN */
    dsUint8_t       reserved1;          /* future use */
    dsUint16_t      reserved2;          /* future use */
}dsmEndTxnExOut_t;

#define dsmEndTxnExOutVersion 1

/*-----+
| Type definition for dsmEndGetDataExIn_t
+-----*/
typedef struct dsmEndGetDataExIn_t
{
    dsUint16_t      stVersion;          /* structure version */
    dsUint32_t      dsmHandle;          /* handle for session */
}dsmEndGetDataExIn_t;

#define dsmEndGetDataExInVersion 1

/*-----+
| Type definition for dsmEndGetDataExOut_t
+-----*/
typedef struct dsmEndGetDataExOut_t
{
    dsUint16_t      stVersion;          /* structure version */
    dsUint16_t      reason;             /* reason code */
    dsStruct64_t    totalLFBytesRecv;   /* total lan free bytes recieved */
}dsmEndGetDataExOut_t;

#define dsmEndGetDataExOutVersion 1

/*-----+
| Type definition for object list on dsmRetentionEvent()
+-----*/
typedef struct dsmObjList

```

```

{
    dsUInt16_t      stVersion;          /* structure version */
    dsUInt32_t      numObjId;           /* number of object IDs in the list */
    ObjID           *objId;             /* list of object IDs to signal */
}dsmObjList_t ;

#define dsmObjlistVersion 1

/*-----+
| Type definition eventType used on dsmRetentionEvent |
+-----*/
typedef enum
{
    eventRetentionActivate = 0x00,      /* signal the server that the event has occurred */
    eventHoldObj,                      /* suspend delete/expire of the object */
    eventReleaseObj                    /* Resume normal delete/expire processing */
}dsmEventType_t;

/*-----+
| Type definition for on dsmRetentionEvent() |
+-----*/
typedef struct dsmRetentionEventIn_t
{
    dsUInt16_t      stVersion;          /* structure version */
    dsUInt32_t      dsmHandle;          /* session Handle */
    dsmEventType_t  eventType;          /* Event type */
    dsmObjList_t    objList;           /* object ID */
}dsmRetentionEventIn_t;

#define dsmRetentionEventInVersion 1

/*-----+
| Type definition for on dsmRetentionEvent() |
+-----*/
typedef struct dsmRetentionEventOut_t
{
    dsUInt16_t      stVersion ;          /* structure version */
}dsmRetentionEventOut_t;

#define dsmRetentionEventOutVersion 1

/*-----+
| Type definition for on dsmRequestBuffer() |
+-----*/
typedef struct requestBufferIn_t
{
    dsUInt16_t      stVersion;          /* structure version */
    dsUInt32_t      dsmHandle;          /* session Handle */
}requestBufferIn_t;

#define requestBufferInVersion 1

/*-----+
| Type definition for on dsmRequestBuffer() |
+-----*/
typedef struct requestBufferOut_t
{
    dsUInt16_t      stVersion ;          /* structure version */
    dsUInt8_t       tsmBufferHandle;     /* handle to tsm Data buffer */
    char            *dataPtr;           /* Address to write data to */
    dsUInt32_t      bufferLen;          /* Max length of data to be written */
}requestBufferOut_t;

#define requestBufferOutVersion 1

/*-----+
| Type definition for on dsmReleaseBuffer() |
+-----*/
typedef struct releaseBufferIn_t
{
    dsUInt16_t      stVersion;          /* structure version */
    dsUInt32_t      dsmHandle;          /* session Handle */
    dsUInt8_t       tsmBufferHandle;     /* handle to tsm Data buffer */
    char            *dataPtr;           /* Address to write data to */
}releaseBufferIn_t;

#define releaseBufferInVersion 1

/*-----+
| Type definition for on dsmReleaseBuffer() |
+-----*/

```

```

typedef struct releaseBufferOut_t
{
    dsUInt16_t      stVersion ;                /* structure version */
}releaseBufferOut_t;

#define releaseBufferOutVersion 1

/*-----+
| Type definition for on dsmGetBufferData()      |
+-----*/
typedef struct getBufferDataIn_t
{
    dsUInt16_t      stVersion;                /* structure version */
    dsUInt32_t      dsmHandle;                /* session Handle */
}getBufferDataIn_t;

#define getBufferDataInVersion 1

/*-----+
| Type definition for on dsmGetBufferData()      |
+-----*/
typedef struct getBufferDataOut_t
{
    dsUInt16_t      stVersion ;                /* structure version */
    dsUInt8_t       tsmBufferHandle;           /* handle to tsm Data buffer */
    char            *dataPtr;                  /* Address of actual data to read */
    dsUInt32_t      numBytes;                  /* Actual number of bytes to read from dataPtr*/
}getBufferDataOut_t;

#define getBufferDataOutVersion 1

/*-----+
| Type definition for on dsmSendBufferData()     |
+-----*/
typedef struct sendBufferDataIn_t
{
    dsUInt16_t      stVersion;                /* structure version */
    dsUInt32_t      dsmHandle;                /* session Handle */
    dsUInt8_t       tsmBufferHandle;           /* handle to tsm Data buffer */
    char            *dataPtr;                  /* Address of actual data to send */
    dsUInt32_t      numBytes;                  /* Actual number of bytes to send from dataPtr*/
}sendBufferDataIn_t;

#define sendBufferDataInVersion 1

/*-----+
| Type definition for on dsmSendBufferData()     |
+-----*/
typedef struct sendBufferDataOut_t
{
    dsUInt16_t      stVersion ;                /* structure version */
}sendBufferDataOut_t;

#define sendBufferDataOutVersion 1

/*-----+
| Type definition for dsmUpdateObjExIn_t        |
+-----*/
typedef struct dsmUpdateObjExIn_t
{
    dsUInt16_t      stVersion;                /* structure version */
    dsUInt32_t      dsmHandle;                /* session Handle */
    dsmSendType     sendType;                 /* send type back/arch */
    char            *descrP;                  /* archive description */
    dsmObjName      *objNameP;                /* objName */
    ObjAttr         *objAttrPtr;              /* attribute */
    dsUInt32_t      objUpdAct;                 /* update action */
    ObjID           archObjId;                /* objId for archive */
}dsmUpdateObjExIn_t;

#define dsmUpdateObjExInVersion 1

/*-----+
| Type definition for dsmUpdateObjExOut_t        |
+-----*/
typedef struct dsmUpdateObjExOut_t
{
    dsUInt16_t      stVersion;                /* structure version */
}dsmUpdateObjExOut_t;

#define dsmUpdateObjExOutVersion 1

```

```
#if (_OPSYS_TYPE == DS_WINNT) && !defined(_WIN64)
#pragma pack()
#endif
```

```
#ifdef _MAC
#pragma options align=reset
#endif
#endif /* _H_DSMAPI_TD */
```

tsmapitd.h

```

/*****
* IBM Spectrum Protect
* API Client Component
*
* IBM Confidential
* (IBM Confidential-Restricted when combined with the Aggregated OCO
* source modules for this program)
*
* OCO Source Materials
*
* 5648-020 (C) Copyright IBM Corporation 1993, 2018
*****/

/*****
* Header File Name: tsmapitd.h
*
* Environment:
* ** This is a platform-independent source file **
*
*
* Design Notes: This file contains basic data types and constants
* includable by all client source files. The constants
* within this file should be set properly for the
* particular machine and operating system on which the
* client software is to be run.
*
* Platform specific definitions are included in dsmapips.h
*
* Descriptive-name: Definitions for IBM Spectrum Protect API constants
*-----*/

#ifndef _H_TSMAPITD
#define _H_TSMAPITD

/*=== set the structure alignment to pack the structures ===*/
#if _OPSYS_TYPE == DS_WINNT
#ifdef _WIN64
#pragma pack(8)
#else
#pragma pack(1)
#endif
#endif

#ifdef _MAC
#pragma options align = packed
#endif

/*=====
Win32 applications using the tsm interface must use the
-DUNICODE flag during compilation.
=====*/
#if _OPSYS_TYPE == DS_WINNT && !defined(DSMAPILIB)
#ifndef UNICODE
#error "Win32 applications using the TSM interface MUST be compiled with the -DUNICODE flag"
#endif
#endif

/*=====
Mac OS X applications using the tsm interface must use the
-DUNICODE flag during compilation.
=====*/
#if _OPSYS_TYPE == DS_MACOS && !defined(DSMAPILIB)

```

```

#ifndef UNICODE
#error "Mac OS X applications using the TSM interface MUST be compiled with the -DUNICODE flag"
#endif
#endif

/*-----+
| Type definition for dsmGetType parameter on tsmBeginGetData() |
+-----*/
typedef enum
{
    gtTsmBackup = 0x00,          /* Backup processing type */
    gtTsmArchive          /* Archive processing type */
} tsmGetType ;

/*-----+
| Type definition for dsmQueryType parameter on tsmBeginQuery() |
+-----*/
typedef enum
{
    qtTsmArchive = 0x00,          /* Archive query type */
    qtTsmBackup,          /* Backup query type */
    qtTsmBackupActive,      /* Fast query for active backup files */
    qtTsmFilespace,          /* Filespace query type */
    qtTsmMC,          /* Mgmt. class query type */
    qtTsmReserved1,          /* future use */
    qtTsmReserved2,          /* future use */
    qtTsmReserved3,          /* future use */
    qtTsmReserved4,          /* future use */
    qtTsmBackupGroups,      /* All group leaders in a specific filesystem */
    qtTsmOpenGroups,        /* All group members associated with a leader */
    qtTsmReserved5,          /* future use */
    qtTsmProxyNodeAuth,      /* nodes that this node can proxy to */
    qtTsmProxyNodePeer,      /* peer nodes under this target node */
    qtTsmReserved6,          /* future use */
    qtTsmReserved7,          /* future use */
    qtTsmReserved8,          /* future use */
} tsmQueryType ;

/*-----+
| Type definition sendType parameter on tsmBindMC() and tsmSendObj() |
+-----*/
typedef enum
{
    stTsmBackup = 0x00,          /* Backup processing type */
    stTsmArchive,          /* Archive processing type */
    stTsmBackupMountWait,      /* Backup processing with mountwait on */
    stTsmArchiveMountWait      /* Archive processing with mountwait on */
} tsmSendType ;

/*-----+
| Type definition for delType parameter on tsmDeleteObj() |
+-----*/
typedef enum
{
    dtTsmArchive = 0x00,          /* Archive delete type */
    dtTsmBackup,          /* Backup delete (deactivate) type */
    dtTsmBackupID          /* Backup delete (remove) type */
} tsmDelType ;

/*-----+
| Type definition sendType parameter on tsmSetAccess() |
+-----*/
typedef enum
{
    atTsmBackup = 0x00,          /* Backup processing type */
    atTsmArchive          /* Archive processing type */
} tsmAccessType;

/*-----+
| Type definition for Overwrite parameter on tsmSendObj() |
+-----*/
typedef enum
{
    owIGNORE = 0x00,
    owYES,
    owNO
} tsmOwType;

/*-----+
| Type definition for API Version on tsmInit() and tsmQueryApiVersion() |
+-----*/

```



```

typedef struct
{
    dsUInt16_t stVersion;      /* Structure version          */
    dsUInt16_t version;        /* API version                */
    dsUInt16_t release;        /* API release                */
    dsUInt16_t level;          /* API level                  */
    dsUInt16_t subLevel;       /* API sub level              */
    dsmBool_t unicode;         /* API unicode?               */
} tsmApiVersionEx;

#define tsmApiVersionExVer      2

/*-----+
| Type definition for Application Version on tsmInit()
+-----*/
typedef struct
{
    dsUInt16_t stVersion;      /* Structure version          */
    dsUInt16_t applicationVersion; /* application version number */
    dsUInt16_t applicationRelease; /* application release number */
    dsUInt16_t applicationLevel; /* application level number   */
    dsUInt16_t applicationSubLevel; /* application sub level number */
} tsmAppVersion;

#define tsmAppVersionVer      1

/*-----+
| Type definition for object name used on BindMC, Send, Delete, Query
+-----*/
typedef struct tsmObjName
{
    dsChar_t fs[DSM_MAX_FSNAME_LENGTH + 1]; /* Filespace name */
    dsChar_t hl[DSM_MAX_HL_LENGTH + 1]; /* High level name */
    dsChar_t ll[DSM_MAX_LL_LENGTH + 1]; /* Low level name */
    dsUInt8_t objType; /* for object type values, see defines above */
    dsChar_t dirDelimiter;
} tsmObjName;

/*-----+
| Type definition for Backup delete info on dsmDeleteObj()
+-----*/
typedef struct tsmDelBack
{
    dsUInt16_t stVersion; /* structure version */
    tsmObjName *objNameP; /* object name */
    dsUInt32_t copyGroup; /* copy group */
} tsmDelBack;

#define tsmDelBackVersion      1

/*-----+
| Type definition for Archive delete info on dsmDeleteObj()
+-----*/
typedef struct
{
    dsUInt16_t stVersion; /* structure version */
    dsStruct64_t objId; /* object ID */
} tsmDelArch;

#define tsmDelArchVersion      1

/*-----+
| Type definition for Backup ID delete info on dsmDeleteObj()
+-----*/
typedef struct
{
    dsUInt16_t stVersion; /* structure version */
    dsStruct64_t objId; /* object ID */
} tsmDelBackID;

#define tsmDelBackIDVersion      1

/*-----+
| Type definition for delete info on dsmDeleteObj()
+-----*/
typedef union
{
    tsmDelBack backInfo;
    tsmDelArch archInfo;
}

```

```

    tsmDelBackID backIDInfo;
} tsmDelInfo ;

/*-----+
| Type definition for Object Attribute parameter on dsmSendObj()
+-----*/
typedef struct tsmObjAttr
{
    dsUint16_t    stVersion;                /* Structure version */
    dsChar_t      owner[DSM_MAX_OWNER_LENGTH + 1]; /* object owner */
    dsStruct64_t  sizeEstimate;             /* Size estimate in bytes of the object */
    dsmBool_t     objCompressed;            /* Is object already compressed? */
    dsUint16_t    objInfoLength;           /* length of object-dependent info */
    char          *objInfo;                /* object-dependent info byte buffer */
    dsChar_t      *mcNameP;                /* mgmnt class name for override */
    tsmOwType      reserved1;              /* for future use */
    tsmOwType      reserved2;              /* for future use */
    dsmBool_t      disableDeduplication;    /* force no dedup for this object */
    dsmBool_t      useExtObjInfo;          /* use ext objinfo up to 1536 */
} tsmObjAttr;

#define tsmObjAttrVersion 5

/*-----+
| Type definition for mcBindKey returned on dsmBindMC()
+-----*/
typedef struct tsmMcBindKey
{
    dsUint16_t    stVersion;                /* structure version */
    dsChar_t      mcName[DSM_MAX_MC_NAME_LENGTH + 1];
    /* Name of mc bound to object. */
    dsmBool_t      backup_cg_exists;        /* True/false */
    dsmBool_t      archive_cg_exists;      /* True/false */
    dsChar_t      backup_copy_dest[DSM_MAX_CG_DEST_LENGTH + 1];
    /* Backup copy dest. name */
    dsChar_t      archive_copy_dest[DSM_MAX_CG_DEST_LENGTH + 1];
    /* Arch copy dest.name */
} tsmMcBindKey;

#define tsmMcBindKeyVersion 1

/*-----+
| Type definition for Mgmt Class queryBuffer on dsmBeginQuery()
+-----*/
typedef struct tsmQryMCData
{
    dsUint16_t    stVersion;                /* structure version */
    dsChar_t      *mcName;                 /* Mgmt class name */
    /* single name to get one or empty string to get all */
    dsmBool_t      mcDetail;               /* Want details or not? */
} tsmQryMCData;

#define tsmQryMCDataVersion 1

/*-----+
| Type definition for Archive Copy Group details on Query MC response
+-----*/
typedef struct tsmArchDetailCG
{
    dsChar_t      cgName[DSM_MAX_CG_NAME_LENGTH + 1]; /* Copy group name */
    dsUint16_t    frequency;                        /* Copy (archive) frequency */
    dsUint16_t    retainVers;                       /* Retain version */
    dsUint8_t     copySer;                          /* for copy serialization values, see defines */
    dsUint8_t     copyMode;                         /* for copy mode values, see defines above */
    dsChar_t      destName[DSM_MAX_CG_DEST_LENGTH + 1]; /* Copy dest name */
    dsmBool_t      blanFreeDest;                    /* Destination has lan free path? */
    dsmBool_t      reserved;                        /* Not currently used */
    dsUint8_t     retainInit;                       /* possible values see dsmapi.h */
    dsUint16_t    retainMin;                        /* if retInit is EVENT num of days */
    dsmBool_t      bDeduplicate;                    /* destination has dedup enabled */
} tsmArchDetailCG;

/*-----+
| Type definition for Backup Copy Group details on Query MC response
+-----*/
typedef struct tsmBackupDetailCG
{
    dsChar_t      cgName[DSM_MAX_CG_NAME_LENGTH + 1]; /* Copy group name */
    dsUint16_t    frequency;                        /* Backup frequency */
    dsUint16_t    verDataExst;                     /* Versions data exists */

```

```

    dsUInt16_t    verDataDltd;                                /* Versions data deleted */
    dsUInt16_t    retXtraVers;                                /* Retain extra versions */
    dsUInt16_t    retOnlyVers;                                /* Retain only versions */
    dsUInt8_t     copySer;                                    /* for copy serialization values, see defines */
    dsUInt8_t     copyMode;                                    /* for copy mode values, see defines above */
    dsChar_t      destName[DSM_MAX.CG_DEST_LENGTH + 1];      /* Copy dest name */
    dsmBool_t     bLanFreeDest;                                /* Destination has lan free path? */
    dsmBool_t     reserved;                                    /* Not currently used */
    dsmBool_t     bDeduplicate;                                /* destination has dedup enabled */
} tsmBackupDetailCG;

/*-----+
| Type definition for Query Mgmt Class detail response on dsmGetNextQObj()|
+-----*/
typedef struct tsmQryRespMCDetailData
{
    dsUInt16_t     stVersion;                                /* structure version */
    dsChar_t       mcName[DSM_MAX_MC_NAME_LENGTH + 1];      /* mc name */
    dsChar_t       mcDesc[DSM_MAX_MC_DESCR_LENGTH + 1];     /*mc description */
    archDetailCG   archDet;                                    /* Archive copy group detail */
    backupDetailCG backupDet;                                  /* Backup copy group detail */
} tsmQryRespMCDetailData;

#define tsmQryRespMCDetailDataVersion 4

/*-----+
| Type definition for Query Mgmt Class summary response on dsmGetNextQObj()|
+-----*/
typedef struct tsmQryRespMCData
{
    dsUInt16_t     stVersion;                                /* structure version */
    dsChar_t       mcName[DSM_MAX_MC_NAME_LENGTH + 1];      /* mc name */
    dsChar_t       mcDesc[DSM_MAX_MC_DESCR_LENGTH + 1];     /* mc description */
} tsmQryRespMCData;

#define tsmQryRespMCDataVersion 1

/*-----+
| Type definition for Archive queryBuffer on tsmBeginQuery()|
+-----*/
typedef struct tsmQryArchiveData
{
    dsUInt16_t     stVersion;                                /* structure version */
    tsmObjName     *objName;                                /* Full dsm name of object */
    dsChar_t       *owner;                                    /* owner name */
    /* for maximum date boundaries, see defines above */
    dsmDate        insDateLowerBound;                        /* low bound archive insert date */
    dsmDate        insDateUpperBound;                        /* hi bound archive insert date */
    dsmDate        expDateLowerBound;                        /* low bound expiration date */
    dsmDate        expDateUpperBound;                        /* hi bound expiration date */
    dsChar_t       *descr;                                    /* archive description */
} tsmQryArchiveData;

#define tsmQryArchiveDataVersion 1

/*-----+
| Type definition for Query Archive response on dsmGetNextQObj()|
+-----*/
typedef struct tsmQryRespArchiveData
{
    dsUInt16_t     stVersion;                                /* structure version */
    tsmObjName     objName;                                    /* Filespace name qualifier */
    dsUInt32_t     copyGroup;                                  /* copy group number */
    dsChar_t       mcName[DSM_MAX_MC_NAME_LENGTH + 1];      /* mc name */
    dsChar_t       owner[DSM_MAX_OWNER_LENGTH + 1];         /* owner name */
    dsStruct64_t   objId;                                      /* Unique copy id */
    dsStruct64_t   reserved;                                    /* backward compatability */
    dsUInt8_t      mediaClass;                                /* media access class */
    dsmDate        insDate;                                    /* archive insertion date */
    dsmDate        expDate;                                    /* expiration date for object */
    dsChar_t       descr[DSM_MAX_DESCR_LENGTH + 1];          /* archive description */
    dsUInt16_t     objInfoLen;                                /* length of object-dependent info */
    dsUInt8_t      reservedObjInfo[DSM_MAX_OBJINFO_LENGTH];  /*object-dependent info */
    dsUInt160_t    restoreOrderExt;                           /* restore order */
    dsStruct64_t   sizeEstimate;                               /* size estimate stored by user */
    dsUInt8_t      compressType;                               /* Compression flag */
    dsUInt8_t      retentionInitiated; /* object waiting on retention event */
    dsUInt8_t      objHeld; /* object is on "hold" see dsmapi.h for values */
    dsUInt8_t      encryptionType;                             /* type of encryption */
    dsmBool_t      clientDeduplicated;                          /* obj deduplicated by API */
    dsUInt8_t      objInfo[DSM_MAX_EXT_OBJINFO_LENGTH];      /*object-dependent info */

```

```

    dsChar_t        compressAlg[DSM_MAX_COMPRESSTYPE_LENGTH + 1]; /* compression algorithm name */
} tsmQryRespArchiveData;

#define tsmQryRespArchiveDataVersion 7

/*-----+
| Type definition for Archive sendBuff parameter on dsmSendObj()
+-----*/
typedef struct tsmSndArchiveData
{
    dsUInt16_t    stVersion;          /* structure version */
    dsChar_t      *descr;             /* archive description */
} tsmSndArchiveData;

#define tsmSndArchiveDataVersion 1

/*-----+
| Type definition for Backup queryBuffer on dsmBeginQuery()
+-----*/
typedef struct tsmQryBackupData
{
    dsUInt16_t    stVersion;          /* structure version */
    tsmObjName    *objName;          /* full dsm name of object */
    dsChar_t      *owner;            /* owner name */
    dsUInt8_t     objState;          /* object state selector */
    dsmDate       pitDate;           /* Date value for point in time restore */
    /* for possible values, see defines above */
    dsUInt32_t    reserved1;
    dsUInt32_t    reserved2;
} tsmQryBackupData;

#define tsmQryBackupDataVersion 3

/*-----+
| Type definition for Query Backup response on dsmGetNextQObj()
+-----*/
typedef struct tsmQryRespBackupData
{
    dsUInt16_t    stVersion;          /* structure version */
    tsmObjName    objName;            /* full dsm name of object */
    dsUInt32_t    copyGroup;          /* copy group number */
    dsChar_t      mcName[DSM_MAX_MC_NAME_LENGTH + 1]; /* mc name */
    dsChar_t      owner[DSM_MAX_OWNER_LENGTH + 1]; /* owner name */
    dsStruct64_t  objId;              /* Unique object id */
    dsStruct64_t  reserved;           /* backward compatability */
    dsUInt8_t     mediaClass;         /* media access class */
    dsUInt8_t     objState;           /* Obj state, active, etc. */
    dsmDate       insDate;            /* backup insertion date */
    dsmDate       expDate;            /* expiration date for object */
    dsUInt16_t    objInfolen;         /* length of object-dependent info */
    dsUInt8_t     reservedObjInfo[DSM_MAX_OBJINFO_LENGTH]; /*object-dependent info */
    dsUInt160_t   restoreOrderExt;    /* restore order */
    dsStruct64_t  sizeEstimate;       /* size estimate stored by user */
    dsStruct64_t  baseObjId;
    dsUInt16_t    baseObjInfolen;     /* length of base object-dependent info */
    dsUInt8_t     baseObjInfo[DSM_MAX_OBJINFO_LENGTH]; /* base object-dependent info */
    dsUInt160_t   baseRestoreOrder;   /* restore order */
    dsUInt32_t    fsID;
    dsUInt8_t     compressType;
    dsmBool_t     isGroupLeader;
    dsmBool_t     isOpenGroup;
    dsUInt8_t     reserved1;          /* for future use */
    dsmBool_t     reserved2;          /* for future use */
    dsUInt16_t    reserved3;          /* for future use */
    reservedInfo_t *reserved4;        /* for future use */
    dsUInt8_t     encryptionType;     /* type of encryption */
    dsmBool_t     clientDeduplicated; /* obj deduplicated by API */
    dsUInt8_t     objInfo[DSM_MAX_EXT_OBJINFO_LENGTH]; /*object-dependent info */
    dsChar_t      compressAlg[DSM_MAX_COMPRESSTYPE_LENGTH + 1]; /* compression algorithm name */
} tsmQryRespBackupData;

#define tsmQryRespBackupDataVersion 8

/*-----+
| Type definition for Active Backup queryBuffer on dsmBeginQuery()
|
| Notes: For the active backup query, only the fs (filesystem) and objType
|        fields of objName need be set.  objType can only be set to
|        DSM_OBJ_FILE or DSM_OBJ_DIRECTORY.  DSM_OBJ_ANY_TYPE will not
|        find a match on the query.
+-----*/
typedef struct tsmQryABBackupData
{

```

```

    dsUInt16_t    stVersion;                                /* structure version */
    tsmObjName    *objName;                                /* Only fs and objtype used */
} tsmQryABackupData;

#define tsmQryABackupDataVersion 1

/*-----+
| Type definition for Query Active Backup response on dsmGetNextQObj() |
+-----*/
typedef struct tsmQryARespBackupData
{
    dsUInt16_t    stVersion;                                /* structure version */
    tsmObjName    objName;                                /* full dsm name of object */
    dsUInt32_t    copyGroup;                               /* copy group number */
    dsChar_t      mcName[DSM_MAX_MC_NAME_LENGTH + 1]; /*management class name*/
    dsChar_t      owner[DSM_MAX_OWNER_LENGTH + 1];       /* owner name */
    dsmDate       insDate;                                 /* backup insertion date */
    dsUInt16_t    objInfoLen;                             /* length of object-dependent info*/
    dsUInt8_t     reservedObjInfo[DSM_MAX_OBJINFO_LENGTH]; /*object-dependent info */
    dsUInt8_t     objInfo[DSM_MAX_EXT_OBJINFO_LENGTH];   /*object-dependent info */
} tsmQryARespBackupData;

#define tsmQryARespBackupDataVersion 2

/*-----+
| Type definition for Backup queryBuffer on dsmBeginQuery() |
+-----*/
typedef struct tsmQryBackupGroups
{
    dsUInt16_t    stVersion;                                /* structure version */
    dsUInt8_t     groupType;
    dsChar_t      *fsName;
    dsChar_t      *owner;
    dsStruct64_t  groupLeaderObjId;
    dsUInt8_t     objType;
    dsUInt32_t    reserved1;
    dsUInt32_t    reserved2;
    dsmBool_t     noRestoreOrder;
    dsmBool_t     noGroupInfo;
    dsChar_t      *hl;
} tsmQryBackupGroups;

#define tsmQryBackupGroupsVersion 4

/*-----+
| Type definition for proxynode queryBuffer on tsmBeginQuery() |
+-----*/
typedef struct tsmQryProxyNodeData
{
    dsUInt16_t    stVersion;                                /* structure version */
    dsChar_t      *targetNodeName;                         /* target node name */
} tsmQryProxyNodeData;

#define tsmQryProxyNodeDataVersion 1

/*-----+
| Type definition for qryRespProxyNodeData parameter used on tsmGetNextQObj() |
+-----*/
typedef struct tsmQryRespProxyNodeData
{
    dsUInt16_t    stVersion;                                /* structure version */
    dsChar_t      targetNodeName[DSM_MAX_ID_LENGTH+1];    /* target node name */
    dsChar_t      peerNodeName[DSM_MAX_ID_LENGTH+1];      /* peer node name */
    dsChar_t      hlAddress[DSM_MAX_ID_LENGTH+1];         /* peer hlAddress */
    dsChar_t      llAddress[DSM_MAX_ID_LENGTH+1];         /* peer llAddress */
} tsmQryRespProxyNodeData;

#define tsmQryRespProxyNodeDataVersion 1

/*-----+
| Type definition for WINNT and OS/2 Filespace attributes |
+-----*/
typedef struct tsmDosFSAttrib
{
    osChar_t      driveLetter;                             /* drive letter for filespace */
    dsUInt16_t    fsInfoLength;                             /* fsInfo length used */
    osChar_t      fsInfo[DSM_MAX_FSINFO_LENGTH]; /*caller-determined data */
} tsmDosFSAttrib;

/*-----+
| Type definition for UNIX Filespace attributes |
+-----*/

```

```

+-----*/
typedef struct tsmUnixFSAttrib
{
    dsUInt16_t    fsInfoLength;          /* fsInfo length used          */
    osChar_t      fsInfo[DSM_MAX_FSINFO_LENGTH]; /* caller-determined data */
} tsmUnixFSAttrib ;

/*-----+
| Type definition for NetWare Filespace attributes |
+-----*/
typedef tsmUnixFSAttrib tsmNetwareFSAttrib;

/*-----+
| Type definition for Filespace attributes on all Filespace calls |
+-----*/
typedef union
{
    tsmNetwareFSAttrib  netwareFSAttr;
    tsmUnixFSAttrib     unixFSAttr ;
    tsmDosFSAttrib      dosFSAttr ;
} tsmFSAttr ;

/*-----+
| Type definition for fsUpd parameter on dsmUpdateFS() |
+-----*/
typedef struct    tsmFSUpd
{
    dsUInt16_t      stVersion ;          /* structure version          */
    dsChar_t        *fsType ;            /* filespace type             */
    dsStruct64_t     occupancy ;          /* occupancy estimate         */
    dsStruct64_t     capacity ;           /* capacity estimate          */
    tsmFSAttr        fsAttr ;            /* platform specific attributes */
} tsmFSUpd ;

#define tsmFSUpdVersion 1

/*-----+
| Type definition for Filespace queryBuffer on dsmBeginQuery() |
+-----*/
typedef struct tsmQryFSData
{
    dsUInt16_t  stVersion;          /* structure version */
    dsChar_t    *fsName;           /* File space name */
} tsmQryFSData;

#define tsmQryFSDataVersion 1

/*-----+
| Type definition for Query Filespace response on dsmGetNextQObj() |
+-----*/
typedef struct tsmQryRespFSData
{
    dsUInt16_t      stVersion;          /* structure version          */
    dsChar_t        fsName[DSM_MAX_FSNAME_LENGTH + 1]; /* Filespace name          */
    dsChar_t        fsType[DSM_MAX_FSTYPE_LENGTH + 1]; /* Filespace type           */
    dsStruct64_t     occupancy;          /* Occupancy est. in bytes.  */
    dsStruct64_t     capacity;           /* Capacity est. in bytes.   */
    tsmFSAttr        fsAttr ;           /* platform specific attributes */
    dsmDate          backStartDate;      /* start backup date         */
    dsmDate          backCompleteDate;   /* end backup Date          */
    dsmDate          reserved1 ;         /* For future use           */
    dsmBool_t        bIsUnicode;
    dsUInt32_t       fsID;
    dsmDate          lastReplStartDate;   /* The last time replication was started */
    dsmDate          lastReplCmpltDate;   /* The last time replication completed   */
    /* (could have had a failure, */
    /* but it still completes) */
    dsmDate          lastBackOpDateFromServer; /* The last store time stamp the client */
    /* saved on the server */
    dsmDate          lastArchOpDateFromServer; /* The last store time stamp the client */
    /* saved on the server */
    dsmDate          lastSpMgOpDateFromServer; /* The last store time stamp the client */
    /* saved on the server */
    dsmDate          lastBackOpDateFromLocal; /* The last store time stamp the client */
    /* saved on the Local */
    dsmDate          lastArchOpDateFromLocal; /* The last store time stamp the client */
    /* saved on the Local */
    dsmDate          lastSpMgOpDateFromLocal; /* The last store time stamp the client */
    /* saved on the Local */
    dsInt32_t        failOverWriteDelay; /* Minutes for client to wait before allowed */
    /* to store to this Repl srvr, Specail codes: */
    /* NO_ACCESS(-1), ACCESS_RDONLY (-2) */
}

```

```

} tsmQryRespFSData;

#define tsmQryRespFSDataVersion 5

/*-----+
| Type definition for regFilespace parameter on dsmRegisterFS()
+-----*/
typedef struct tsmRegFSData
{
    dsUInt16_t    stVersion;                /* structure version */
    dsChar_t      *fsName;                  /* Filespace name */
    dsChar_t      *fsType;                  /* Filespace type */
    dsStruct64_t  occupancy;                /* Occupancy est. in bytes. */
    dsStruct64_t  capacity;                 /* Capacity est. in bytes. */
    tsmFSAttr     fsAttr;                   /* platform specific attributes */
} tsmRegFSData;

#define tsmRegFSDataVersion 1

/*-----+
| Type definition for session info response on dsmQuerySessionInfo()
+-----*/
typedef struct
{
    dsUInt16_t    stVersion;                /* Structure version */
    /*-----*/
    /* Server information */
    /*-----*/
    dsChar_t      serverHost[DSM_MAX_SERVERNAME_LENGTH+1];
    /* Network host name of DSM server */
    dsUInt16_t    serverPort;               /* Server comm port on host */
    dsmDate       serverDate;               /* Server's date/time */
    dsChar_t      serverType[DSM_MAX_SERVERTYPE_LENGTH+1];
    /* Server's execution platform */
    dsUInt16_t    serverVer;                /* Server's version number */
    dsUInt16_t    serverRel;                /* Server's release number */
    dsUInt16_t    serverLev;                /* Server's level number */
    dsUInt16_t    serverSubLev;             /* Server's sublevel number */
    /*-----*/
    /* Client Defaults */
    /*-----*/
    dsChar_t      nodeType[DSM_MAX_PLATFORM_LENGTH+1]; /*node/application type*/
    dsChar_t      fsdelim;                  /* File space delimiter */
    dsChar_t      hldelim;                  /* Delimiter betw highlev & lowlev */
    dsUInt8_t     compression;              /* Compression flag */
    dsUInt8_t     archDel;                  /* Archive delete permission */
    dsUInt8_t     backDel;                  /* Backup delete permission */
    dsUInt32_t     maxBytesPerTxn;           /* for future use */
    dsUInt16_t     maxObjPerTxn;            /* The max objects allowed in a txn */
    /*-----*/
    /* Session Information */
    /*-----*/
    dsChar_t      id[DSM_MAX_ID_LENGTH+1]; /* Sign-in id node name */
    dsChar_t      owner[DSM_MAX_OWNER_LENGTH+1]; /* Sign-in owner */
    /* (for multi-user platforms) */
    dsChar_t      confFile[DSM_PATH_MAX + DSM_NAME_MAX +1];
    /* len is platform dep */
    /* dsInit name of appl config file */
    dsUInt8_t     opNoTrace;                /* dsInit option - NoTrace = 1 */
    /*-----*/
    /* Policy Data */
    /*-----*/
    dsChar_t      domainName[DSM_MAX_DOMAIN_LENGTH+1]; /* Domain name */
    dsChar_t      policySetName[DSM_MAX_PS_NAME_LENGTH+1];
    /* Active policy set name */
    dsmDate       polActDate;               /* Policy set activation date */
    dsChar_t      gpfltMCName[DSM_MAX_MC_NAME_LENGTH+1]; /* Default Mgmt Class */
    dsUInt16_t    gpBackRetn;              /* Grace-period backup retention */
    dsUInt16_t    gpArchRetn;              /* Grace-period archive retention */
    dsChar_t      adsmServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* adsm server name */
    dsmBool_t     archiveRetentionProtection; /* is server Retention protection enabled */
    dsUInt64_t     maxBytesPerTxn_64;       /* for future use */
    dsmBool_t     lanFreeEnabled;           /* lan free option is set */
    dsmDedupType   dedupType;              /* server or clientOrServer */
    dsChar_t      accessNode[DSM_MAX_ID_LENGTH+1]; /* as node node name */
    /*-----*/
    /* Replication and fail over information */
    /*-----*/
    dsmFailOvrCfgType failOverCfgType; /* status of fail over */
    dsChar_t      replServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* repl server name */
    dsChar_t      homeServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* home server name */

```

```

    dsChar_t      replServerHost[DSM_MAX_SERVERNAME_LENGTH+1]; /* Network host name of DSM server */
    dsInt32_t     replServerPort; /* Server comm port on host */
} tsmApiSessInfo;

#define tsmApiSessInfoVersion 6

/*-----+
| Type definition for Query options response on dsmQueryCliOptions()
| and dsmQuerySessOptions()
+-----*/

typedef struct
{
    dsUInt16_t    stVersion;
    dsChar_t      dsmiDir[DSM_PATH_MAX + DSM_NAME_MAX +1];
    dsChar_t      dsmiConfig[DSM_PATH_MAX + DSM_NAME_MAX +1];
    dsChar_t      serverName[DSM_MAX_SERVERNAME_LENGTH+1];
    dsInt16_t     commMethod;
    dsChar_t      serverAddress[DSM_MAX_SERVER_ADDRESS];
    dsChar_t      nodeName[DSM_MAX_NODE_LENGTH+1];
    dsmBool_t     compression;
    dsmBool_t     compressalways;
    dsmBool_t     passwordAccess;
} tsmOptStruct ;

#define tsmOptStructVersion 1

/*-----+
| Type definition for qryRespAccessData parameter used on dsmQueryAccess()
+-----*/

typedef struct
{
    dsUInt16_t     stVersion ; /* structure version */
    dsChar_t       node[DSM_MAX_ID_LENGTH+1]; /* node name */
    dsChar_t       owner[DSM_MAX_OWNER_LENGTH+1]; /* owner */
    tsmObjName     objName ; /* object name */
    dsmAccessType  accessType; /* archive or backup */
    dsUInt32_t     ruleNumber ; /* Access rule id */
} tsmQryRespAccessData;

#define tsmQryRespAccessDataVersion 1

/*-----+
| Type definition for envSetUp parameter on dsmSetUp()
+-----*/

typedef struct tsmEnvSetUp
{
    dsUInt16_t     stVersion; /* structure version */
    dsChar_t       dsmiDir[DSM_PATH_MAX + DSM_NAME_MAX +1];
    dsChar_t       dsmiConfig[DSM_PATH_MAX + DSM_NAME_MAX +1];
    dsChar_t       dsmiLog[DSM_PATH_MAX + DSM_NAME_MAX +1];
    char           **argv; /* for executables name argv[0] */
    dsChar_t       logName[DSM_NAME_MAX +1];
    dsmBool_t      reserved1; /* for future use */
    dsmBool_t      reserved2; /* for future use */
} tsmEnvSetUp;

#define tsmEnvSetUpVersion 4

/*-----+
| Type definition for dsmInitExIn_t
+-----*/

typedef struct tsmInitExIn_t
{
    dsUInt16_t     stVersion; /* structure version */
    tsmApiVersionEx *apiVersionExP;
    dsChar_t       *clientNodeNameP;
    dsChar_t       *clientOwnerNameP;
    dsChar_t       *clientPasswordP;
    dsChar_t       *userNameP;
    dsChar_t       *userPasswordP;
    dsChar_t       *applicationTypeP;
    dsChar_t       *configfile;
    dsChar_t       *options;
    dsChar_t       dirDelimiter;
    dsmBool_t      useUnicode;
    dsmBool_t      bCrossPlatform;
    dsmBool_t      bService;
    dsmBool_t      bEncryptKeyEnabled;

```



```

    dsChar_t          *encryptionPasswordP;
    dsmBool_t         useTsmBuffers;
    dsUInt8_t         numTsmBuffers;
    tsmAppVersion     appVersionP;
} tsmInitExIn_t;

#define tsmInitExInVersion 5

/*-----+
| Type definition for dsmInitExOut_t
+-----*/
typedef struct tsmInitExOut_t
{
    dsUInt16_t         stVersion; /* structure version */
    dsInt16_t          userNameAuthorities;
    dsInt16_t          infoRC; /* error return code if encountered */
    /* adsm server name */
    dsChar_t           adsmServerName[DSM_MAX_SERVERNAME_LENGTH+1];
    dsUInt16_t         serverVer; /* Server's version number */
    dsUInt16_t         serverRel; /* Server's release number */
    dsUInt16_t         serverLev; /* Server's level number */
    dsUInt16_t         serverSubLev; /* Server's sublevel number */
    dsmBool_t          bIsFailOverMode; /* true if failover has occurred */
    dsChar_t           replServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* repl server name */
    dsChar_t           homeServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* home server name */
} tsmInitExOut_t;

#define tsmInitExOutVersion 3

/*-----+
| Type definition for dsmLogExIn_t
+-----*/
typedef struct tsmLogExIn_t
{
    dsUInt16_t         stVersion; /* structure version */
    dsmLogSeverity     severity;
    dsChar_t           appMsgID[8];
    dsmLogType         logType; /* log type : local, server, both */
    dsChar_t           *message; /* text of message to be logged */
    dsChar_t           appName[DSM_MAX_PLATFORM_LENGTH];
    dsChar_t           osPlatform[DSM_MAX_PLATFORM_LENGTH];
    dsChar_t           appVersion[DSM_MAX_PLATFORM_LENGTH];
} tsmLogExIn_t;

#define tsmLogExInVersion 2

/*-----+
| Type definition for dsmLogExOut_t
+-----*/
typedef struct tsmLogExOut_t
{
    dsUInt16_t         stVersion; /* structure version */
} tsmLogExOut_t;

#define tsmLogExOutVersion 1

/*-----+
| Type definition for dsmRenameIn_t
+-----*/
typedef struct tsmRenameIn_t
{
    dsUInt16_t         stVersion; /* structure version */
    dsUInt32_t         tsmHandle; /* handle for session */
    dsUInt8_t          repository; /* Backup or Archive */
    tsmObjName         *objNameP; /* object name */
    dsChar_t           newHl[DSM_MAX_HL_LENGTH + 1]; /* new High level name */
    dsChar_t           newLl[DSM_MAX_LL_LENGTH + 1]; /* new Low level name */
    dsmBool_t          merge; /* merge into existing name */
    ObjID              objId; /* objId for Archive */
} tsmRenameIn_t;

#define tsmRenameInVersion 1

/*-----+
| Type definition for dsmRenameOut_t
+-----*/
typedef struct tsmRenameOut_t
{
    dsUInt16_t         stVersion; /* structure version */
} tsmRenameOut_t;

```

```

#define tsmRenameOutVersion 1

/*-----+
| Type definition for tsmEndSendObjExIn_t
+-----*/
typedef struct tsmEndSendObjExIn_t
{
    dsUInt16_t      stVersion;                /* structure version */
    dsUInt32_t      tsmHandle;                /* handle for session */
} tsmEndSendObjExIn_t;

#define tsmEndSendObjExInVersion 1

/*-----+
| Type definition for dsmEndSendObjExOut_t
+-----*/
typedef struct tsmEndSendObjExOut_t
{
    dsUInt16_t      stVersion;                /* structure version */
    dsStruct64_t     totalBytesSent;           /* total bytes read from app */
    dsmBool_t       objCompressed;            /* was object compressed */
    dsStruct64_t     totalCompressSize;        /* total size after compress */
    dsStruct64_t     totalLFBytesSent;        /* total bytes sent Lan Free */
    dsUInt8_t       encryptionType;          /* type of encryption used */
    dsmBool_t       objDeduplicated;          /* was object processed for dist. data dedup */
    dsStruct64_t     totalDedupSize;          /* total size after de-dup */
} tsmEndSendObjExOut_t;

#define tsmEndSendObjExOutVersion 3

/*-----+
| Type definition for tsmGroupHandlerIn_t
+-----*/
typedef struct tsmGroupHandlerIn_t
{
    dsUInt16_t      stVersion;                /* structure version */
    dsUInt32_t      tsmHandle;                /* handle for session */
    dsUInt8_t       groupType;                /* Type of group */
    dsUInt8_t       actionType;               /* Type of group operation */
    dsUInt8_t       memberType;              /* Type of member: Leader or member */
    dsStruct64_t     leaderObjId;             /* OBJID of the groupleader */
    dsChar_t        *uniqueGroupTagP;         /* Unique group identifier */
    tsmObjName       *objNameP;              /* group leader object name */
    dsmGetList       memberObjList;          /* list of objects to remove, assign */
} tsmGroupHandlerIn_t;

#define tsmGroupHandlerInVersion 1

/*-----+
| Type definition for tsmGroupHandlerExOut_t
+-----*/
typedef struct tsmGroupHandlerOut_t
{
    dsUInt16_t      stVersion;                /* structure version */
} tsmGroupHandlerOut_t;

#define tsmGroupHandlerOutVersion 1

/*-----+
| Type definition for tsmEndTxnExIn_t
+-----*/
typedef struct tsmEndTxnExIn_t
{
    dsUInt16_t      stVersion;                /* structure version */
    dsUInt32_t      tsmHandle;                /* handle for session */
    dsUInt8_t       vote;
} tsmEndTxnExIn_t;

#define tsmEndTxnExInVersion 1

/*-----+
| Type definition for tsmEndTxnExOut_t
+-----*/
typedef struct tsmEndTxnExOut_t
{
    dsUInt16_t      stVersion;                /* structure version */
    dsUInt16_t      reason;                   /* reason code */
    dsStruct64_t     groupleaderObjId;        /* groupleader obj id returned on */
    /* DSM_ACTION_OPEN */
    dsUInt8_t       reserved1;                /* future use */
    dsUInt16_t      reserved2;                /* future use */
} tsmEndTxnExOut_t;

```

```

#define tsmEndTxnExOutVersion 1

/*-----+
| Type definition for tsmEndGetDataExIn_t
+-----*/
typedef struct tsmEndGetDataExIn_t
{
    dsUint16_t      stVersion;    /* structure version */
    dsUint32_t      tsmHandle;    /* handle for session */
}tsmEndGetDataExIn_t;

#define tsmEndGetDataExInVersion 1

/*-----+
| Type definition for tsmEndGetDataExOut_t
+-----*/
typedef struct tsmEndGetDataExOut_t
{
    dsUint16_t      stVersion;    /* structure version */
    dsUint16_t      reason;       /* reason code */
    dsStruct64_t    totalLFBytesRecv; /* total lan free bytes recieved */
}tsmEndGetDataExOut_t;

#define tsmEndGetDataExOutVersion 1

/*-----+
| Type definition for on tsmRetentionEvent()
+-----*/
typedef struct tsmRetentionEventIn_t
{
    dsUint16_t      stVersion;    /* structure version */
    dsUint32_t      tsmHandle;    /* session Handle */
    dsmEventType_t  eventType;    /* Event type */
    dsmObjList_t    objList;      /* object ID */
}tsmRetentionEventIn_t;

#define tsmRetentionEventInVersion 1

/*-----+
| Type definition for on tsmRetentionEvent()
+-----*/
typedef struct tsmRetentionEventOut_t
{
    dsUint16_t      stVersion;    /* structure version */
}tsmRetentionEventOut_t;

#define tsmRetentionEventOutVersion 1

/*-----+
| Type definition for tsmUpdateObjExIn_t
+-----*/
typedef struct tsmUpdateObjExIn_t
{
    dsUint16_t      stVersion;    /* structure version */
    dsUint32_t      tsmHandle;    /* session Handle */
    tsmSendType     sendType;     /* send type back/arch */
    dsChar_t        *descrP;      /* archive description */
    tsmObjName      *objNameP;    /* objName */
    tsmObjAttr      *objAttrPtr;  /* attribute */
    dsUint32_t      objUpdAct;    /* update action */
    ObjID           archObjId;    /* objId for archive */
}tsmUpdateObjExIn_t;

#define tsmUpdateObjExInVersion 1

/*-----+
| Type definition for tsmUpdateObjExOut_t
+-----*/
typedef struct tsmUpdateObjExOut_t
{
    dsUint16_t      stVersion;    /* structure version */
}tsmUpdateObjExOut_t;

#define tsmUpdateObjExOutVersion 1

#if _OPSYS_TYPE == DS_WINNT
#pragma pack()
#endif

#ifdef _MAC

```



```

/* added for the extended restore order */
typedef struct
{
    dsUInt32_t top;
    dsUInt32_t hi_hi;
    dsUInt32_t hi_lo;
    dsUInt32_t lo_hi;
    dsUInt32_t lo_lo;
} dsUInt160_t ;

#if defined(_LONG_LONG)
    typedef __int64          dsInt64_t;
    typedef unsigned __int64 dsUInt64_t;
    /*=== A "true" unsigned 64-bit integer ===*/
    typedef __int64          dsLongLong_t;
#else
typedef struct tagUINT64_t
{
    dsUInt32_t hi;          /* Most significant 32 bits. */
    dsUInt32_t lo;          /* Least significant 32 bits. */
} dsUInt64_t;
#endif

/*-----+
| Type definition for bool_t |
+-----*/
/*
 * Had to create a Boolean type that didn't clash with any other predefined
 * version in any operating system or windowing system.
 */
typedef enum
{
    dsmFalse = 0x00,
    dsmTrue  = 0x01
} dsmBool_t ;

/*=== for backward compatability ===*/
#define uint8      dsUInt8_t
#define int8       dsInt8_t
#define uint16     dsUInt16_t
#define int16      dsInt16_t
#define uint32     dsUInt32_t
#define int32      dsInt32_t
#define uint64     dsStruct64_t
#define bool_t     dsBool_t
#define dsBool_t   dsmBool_t
#define bTrue      dsmTrue
#define bFalse     dsmFalse

typedef struct
{
    dsUInt32_t hi;          /* Most significant 32 bits. */
    dsUInt32_t lo;          /* Least significant 32 bits. */
} dsStruct64_t ;

#endif /* DSMAPILIB */

#ifndef _WIN64
#pragma pack()
#endif
#endif /* _H_DSMAPIPS */

```

release.h

```

/*****
 * IBM Spectrum Protect
 * Common Source Component
 *
 * (C) Copyright IBM Corporation 1993,2018
 *****/

/*****
 * Header File Name: release.h
 *
 * Environment: *****/

```

```

*                ** This is a platform-independent source file **
*                ****
*
* Design Notes:   This file contains the common information about
*                the actual version.release.level.sublevel
*
* Descriptive-name: Definitions for IBM Spectrum Protect version
*
* Note: This file should contain no LOG or CMVC information. It is
*       shipped with the API code.
*
*-----*/

#ifndef _H_RELEASE
#define _H_RELEASE

#define COMMON_VERSION      8
#define COMMON_RELEASE      1
#define COMMON_LEVEL        12
#define COMMON_SUBLEVEL     0
#define COMMON_DRIVER       dsTEXT("")

#define COMMON_VERSIONTXT "8.1.12.0"

#define SHIPYEARTXT ""
#define SHIPYEARTXTW dsTEXT("2021")
#define TSMPRODTXT "IBM Spectrum Protect"

/*=====
The following string definitions are used for VERSION information
and should not be converted to dsTEXT or osTEXT. They are used
only at link time.

These are also used when the Jar file is built on Unix. See the
the perl script tools/unx/mzbuild/createReleaseJava
=====*/
#define COMMON_VERSION_STR "8"
#define COMMON_RELEASE_STR "1"
#define COMMON_LEVEL_STR "12"
#define COMMON_SUBLEVEL_STR "0"
#define COMMON_DRIVER_STR ""

/*=== product names definitions ===*/
#define COMMON_NAME_DFDSM 1
#define COMMON_NAME_ADSM 2
#define COMMON_NAME_TSM 3
#define COMMON_NAME_ITSM 4
#define COMMON_NAME COMMON_NAME_ITSM

/*=====
Internal version, release, and level (build) version. This
should be unique for every version+release+ptf of a product.
This information is recorded in the file attributes and data
stream for diagnostic purposes.
NOTE: DO NOT MODIFY THESE VALUES. YOU CAN ONLY ADD NEW ENTRIES!
=====*/
#define COMMON_BUILD_TSM_510 1
#define COMMON_BUILD_TSM_511 2
#define COMMON_BUILD_TSM_515 3
#define COMMON_BUILD_TSM_516 4
#define COMMON_BUILD_TSM_520 5
#define COMMON_BUILD_TSM_522 6
#define COMMON_BUILD_TSM_517 7
#define COMMON_BUILD_TSM_523 8
#define COMMON_BUILD_TSM_530 9
#define COMMON_BUILD_TSM_524 10
#define COMMON_BUILD_TSM_532 11
#define COMMON_BUILD_TSM_533 12
#define COMMON_BUILD_TSM_525 13
#define COMMON_BUILD_TSM_534 14
#define COMMON_BUILD_TSM_540 15
#define COMMON_BUILD_TSM_535 16
#define COMMON_BUILD_TSM_541 17
#define COMMON_BUILD_TSM_550 18
#define COMMON_BUILD_TSM_542 19
#define COMMON_BUILD_TSM_551 20
#define COMMON_BUILD_TSM_610 21
#define COMMON_BUILD_TSM_552 22
#define COMMON_BUILD_TSM_611 23
#define COMMON_BUILD_TSM_543 24
#define COMMON_BUILD_TSM_620 25
#define COMMON_BUILD_TSM_612 26

```

```

#define COMMON_BUILD_TSM_553 27
#define COMMON_BUILD_TSM_613 28
#define COMMON_BUILD_TSM_621 29
#define COMMON_BUILD_TSM_622 30
#define COMMON_BUILD_TSM_614 31
#define COMMON_BUILD_TSM_623 32
#define COMMON_BUILD_TSM_630 33
#define COMMON_BUILD_TSM_615 34
#define COMMON_BUILD_TSM_624 35
#define COMMON_BUILD_TSM_631 36
#define COMMON_BUILD_TSM_640 37
#define COMMON_BUILD_TSM_710 38
#define COMMON_BUILD_TSM_625 39
#define COMMON_BUILD_TSM_641 40
#define COMMON_BUILD_TSM_711 41
#define COMMON_BUILD_TSM_712 42
#define COMMON_BUILD_TSM_713 43
#define COMMON_BUILD_TSM_714 44
#define COMMON_BUILD_TSM_720 45
#define COMMON_BUILD_TSM_721 46
#define COMMON_BUILD_TSM_642 47
#define COMMON_BUILD_TSM_643 48
#define COMMON_BUILD_TSM_715 49
#define COMMON_BUILD_TSM_716 50
#define COMMON_BUILD_TSM_810 51
#define COMMON_BUILD_TSM_811 52
#define COMMON_BUILD_TSM_812 53
#define COMMON_BUILD_TSM_718 54
#define COMMON_BUILD_TSM_814 55
#define COMMON_BUILD_TSM_816 56
#define COMMON_BUILD_TSM_817 57
#define COMMON_BUILD_TSM_818 58
#define COMMON_BUILD_TSM_819 59
#define COMMON_BUILD_TSM_8110 60
#define COMMON_BUILD_TSM_8111 61
#define COMMON_BUILD_TSM_8112 62
#define COMMON_BUILD COMMON_BUILD_TSM_8112

/*==== 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_718 = 718;
static const int VRL_810 = 810;
static const int VRL_811 = 811;
static const int VRL_812 = 812;
static const int VRL_814 = 814;
static const int VRL_816 = 816;
static const int VRL_817 = 817;
static const int VRL_818 = 818;
static const int VRL_819 = 819;
static const int VRL_8110 = 8110;
static const int VRL_8111 = 8111;
static const int VRL_8112 = 8112;

#define TDP4VE_PLATFORM_STRING_MBCS "TDP VMware"
#define TDP4VE_PLATFORM_STRING dsTEXT("TDP VMware")

#define TDP4HYPERV_PLATFORM_STRING_MBCS "TDP HyperV"
#define TDP4HYPERV_PLATFORM_STRING dsTEXT("TDP HyperV")

#endif /* _H_RELEASE */

```

付録 C API 関数定義ソース・ファイル

この付録には、dsmapifp.h ヘッダー・ファイルが含まれているので、API のための関数定義が分かります。

注 : **DSMLINKAGE** は、オペレーティング・システム によって、定義が異なります。それぞれのオペレーティング・システム の dsmapips.h ファイル内の定義を参照してください。

ここに示す情報には、API とともに配布されるファイルのポイント・イン・タイム・コピーが含まれています。最新バージョンの API 配布パッケージ内のファイルを参照してください。

dsmapifp.h

```
/*
*****
* IBM Spectrum Protect
* API Client Component
*
* IBM Confidential
* (IBM Confidential-Restricted when combined with the Aggregated OCO
* source modules for this program)
*
* OCO Source Materials
*
* 5648-020 (C) Copyright IBM Corporation 1993, 2016
*****
*/

/*
*****
*/
/* Header File Name: dsmapifp.h
*/
/*
*/
/* Descriptive-name: IBM Spectrum Protect API function prototypes
*/
/*
*****
*/
#ifndef _H_DSMAPIFP
#define _H_DSMAPIFP

#ifdef __cplusplus
extern "C" {
#endif

#ifdef DYNALOAD_DSMAPI

/* function will be dynamically loaded */
#include "dsmapidl.h"

#else

/* functions will be implicitly loaded from library */

/*
*****
*/
/*
P U B L I C F U N C T I O N S
*/
/*
*****
*/

extern dsInt16_t DSMLINKAGE dsmBeginGetData(
    dsUInt32_t
    dsBool_t
    dsmGetType
    dsmGetList
    dsmHandle,
    mountWait,
    getType,
    *dsmGetObjListP
);

extern dsInt16_t DSMLINKAGE dsmBeginQuery(
    dsUInt32_t
    dsmQueryType
    dsmQueryBuff
    dsmHandle,
    queryType,
    *queryBuffer
);

extern dsInt16_t DSMLINKAGE dsmBeginTxn(
    dsUInt32_t
    dsmHandle
);

extern dsInt16_t DSMLINKAGE dsmBindMC(
    dsUInt32_t
    dsmObjName
    dsmSendType
    dsmHandle,
    *objNameP,
    sendType,

```



```

);    mcBindKey          *mcBindKeyP

extern dsInt16_t DSMLINKAGE dsmChangePW(
    dsUInt32_t
    char
    char
    *oldPW,
    *newPW
);

extern dsInt16_t DSMLINKAGE dsmCleanUp(
    dsBool_t
    mtFlag
);

extern dsInt16_t DSMLINKAGE dsmDeleteAccess(
    dsUInt32_t
    dsUInt32_t
    dsmHandle,
    ruleNum
);

extern dsInt16_t DSMLINKAGE dsmDeleteObj(
    dsUInt32_t
    dsmDelType
    dsmDelInfo
    dsmHandle,
    delType,
    delInfo
);

extern dsInt16_t DSMLINKAGE dsmDeleteFS(
    dsUInt32_t
    char
    dsUInt8_t
    dsmHandle,
    *fsName,
    repository
);

extern dsInt16_t DSMLINKAGE dsmEndGetData(
    dsUInt32_t
    dsmHandle
);

extern dsInt16_t DSMLINKAGE dsmEndGetDataEx(
    dsmEndGetDataExIn_t
    dsmEndGetDataExOut_t
    *dsmEndGetDataExInP,
    *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
    dsmEndSendObjExOut_t
    *dsmEndSendObjExInP,
    *dsmEndSendObjExOutP
);

extern dsInt16_t DSMLINKAGE dsmEndTxnEx(
    dsmEndTxnExIn_t
    dsmEndTxnExOut_t
    *dsmEndTxnExInP,
    *dsmEndTxnExOutP
);

extern dsInt16_t DSMLINKAGE dsmEndTxn(
    dsUInt32_t
    dsUInt8_t
    dsUInt16_t
    dsmHandle,
    vote,
    *reason
);

extern dsInt16_t DSMLINKAGE dsmGetData(
    dsUInt32_t
    DataBlk
    dsmHandle,
    *dataBlkPtr
);

extern dsInt16_t DSMLINKAGE dsmGetBufferData(
    getBufferDataIn_t
    getBufferDataOut_t
    *dsmGetBufferDataInP,
    *dsmGetBufferDataOutP
);

extern dsInt16_t DSMLINKAGE dsmGetNextQObj(
    dsUInt32_t
    DataBlk
    dsmHandle,
    *dataBlkPtr
);

```

```

extern dsInt16_t DSMLINKAGE dsmGetObj(
    dsUInt32_t
    ObjID
    DataBlk
    *dsmHandle,
    *objIdP,
    *dataBlkPtr
);

extern dsInt16_t DSMLINKAGE dsmGroupHandler(
    dsmGroupHandlerIn_t
    dsmGroupHandlerOut_t
    *dsmGroupHandlerInP,
    *dsmGroupHandlerOutP
);

extern dsInt16_t DSMLINKAGE dsmInit(
    dsUInt32_t
    dsmApiVersion
    char
    char
    char
    char
    char
    char
    *dsmHandle,
    *dsmApiVersionP,
    *clientNodeNameP,
    *clientOwnerNameP,
    *clientPasswordP,
    *applicationType,
    *configfile,
    *options
);

extern dsInt16_t DSMLINKAGE dsmInitEx(
    dsUInt32_t
    dsmInitExIn_t
    dsmInitExOut_t
    *dsmHandleP,
    *dsmInitExInP,
    *dsmInitExOutP
);

extern dsInt16_t DSMLINKAGE dsmLogEvent(
    dsUInt32_t
    logInfo
    *dsmHandle,
    *logInfoP
);

extern dsInt16_t DSMLINKAGE dsmLogEventEx(
    dsUInt32_t
    dsmLogExIn_t
    dsmLogExOut_t
    *dsmHandle,
    *dsmLogExInP,
    *dsmLogExOutP
);

extern dsInt16_t DSMLINKAGE dsmQueryAccess(
    dsUInt32_t
    qryRespAccessData
    dsUInt16_t
    *dsmHandle,
    **accessListP,
    *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
    ApiSessInfo
    *dsmHandle,
    *SessInfoP
);

extern dsInt16_t DSMLINKAGE dsmQuerySessOptions(
    dsUInt32_t
    optStruct
    *dsmHandle,
    *optstructP
);

extern dsInt16_t DSMLINKAGE dsmRCMsg(
    dsUInt32_t
    dsInt16_t
    char
    *dsmHandle,
    dsmRC,
    *msg
);

extern dsInt16_t DSMLINKAGE dsmRegisterFS(
    dsUInt32_t
    regFSData
    *dsmHandle,
    *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 */

#ifdef __cplusplus
}
#endif

```

```
#endif /* _H_DSMAPIFP */
```

tsmapifp.h

このセクションには、API のための関数定義が含まれています。これは tsmapifp.h ヘッダー・ファイルのコピーです。

注 : DSMLINKAGE は、オペレーティング・システムによって、定義が異なります。それぞれのオペレーティング・システムの tsmapi.h ファイル内の定義を参照してください。

```
/******
 * IBM Spectrum Protect
 * API Client Component
 *
 * IBM Confidential
 * (IBM Confidential-Restricted when combined with the Aggregated OCO
 * source modules for this program)
 *
 * OCO Source Materials
 *
 * 5648-020 (C) Copyright IBM Corporation 1993, 2016
 *****/

/*****
 * Header File Name: tsmapifp.h
 */
/* Descriptive-name: IBM Spectrum Protect API function prototypes
 */
 *****/
#ifndef _H_TSMAPIFP
#define _H_TSMAPIFP

#ifdef __cplusplus
extern "C" {
#endif

#ifdef DYNALOAD_DSMAPI

/* function will be dynamically loaded */
#include "dsmapidl.h"

#else

/* functions will be implicitly loaded from library */

/*=====
 * P U B L I C F U N C T I O N S
 *=====
 */

typedef void tsmQueryBuff;

extern dsInt16_t DSMLINKAGE tsmBeginGetData(
    dsUInt32_t tsmHandle,
    dsBool_t mountWait,
    tsmGetType_t getType,
    dsmGetList_t *dsmGetObjListP
);

extern dsInt16_t DSMLINKAGE tsmBeginQuery(
    dsUInt32_t tsmHandle,
    tsmQueryType_t queryType,
    tsmQueryBuff_t queryBuffer
);

extern dsInt16_t DSMLINKAGE tsmBeginTxn(
    dsUInt32_t tsmHandle
);

extern dsInt16_t DSMLINKAGE tsmBindMC(
    dsUInt32_t tsmHandle,
    tsmObjName_t *objNameP,
    tsmSendType_t sendType,
    tsmMcBindKey_t *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(
    dsInt16_t DSMLINKAGE 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(
    dsInt16_t DSMLINKAGE 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(
    dsInt16_t DSMLINKAGE tsmEndTxnExIn_t      *tsmEndTxnExInP,
    tsmEndTxnExOut_t      *tsmEndTxnExOutP
);

extern dsInt16_t DSMLINKAGE tsmGetData(
    dsUInt32_t      tsmHandle,
    DataBlk*dataBlkPtr
);

extern dsInt16_t DSMLINKAGE tsmGetBufferData(
    dsInt16_t DSMLINKAGE 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(
    dsInt16_t releaseBufferIn_t,
    dsInt16_t releaseBufferOut_t,
    tsmReleaseBufferInP,
    tsmReleaseBufferOutP
);

extern dsInt16_t DSMLINKAGE tsmRenameObj(
    dsInt16_t tsmRenameIn_t,
    dsInt16_t tsmRenameOut_t,
    tsmRenameInP,
    tsmRenameOutP
);

extern dsInt16_t DSMLINKAGE tsmRequestBuffer(
    dsInt16_t requestBufferIn_t,
    dsInt16_t requestBufferOut_t,
    tsmRequestBufferInP,
    tsmRequestBufferOutP
);

extern dsInt16_t DSMLINKAGE tsmRetentionEvent(
    dsInt16_t tsmRetentionEventIn_t,
    dsInt16_t tsmRetentionEventOut_t,
    tsmRetentionEventInP,
    tsmRetentionEventOutP
);

extern dsInt16_t DSMLINKAGE tsmSendBufferData(
    dsInt16_t sendBufferDataIn_t,
    dsInt16_t sendBufferDataOut_t,
    tsmSendBufferDataInP,
    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 */

#ifdef __cplusplus
}
#endif

#endif /* _H_TSMAPIFP */

```

付録 D IBM Spectrum Protect 製品ファミリーのアクセシビリティ機能

アクセシビリティ機能は、運動障害または視覚障害など身体に障害を持つユーザーが情報技術コンテンツを快適に使用できるように支援します。

概要

IBM Spectrum Protect ファミリーの製品は、以下の主なアクセシビリティ機能を提供します。

- キーボードのみによる操作
- スクリーン・リーダー (読み上げソフトウェア) を使用する操作

IBM Spectrum Protect ファミリー製品は、最新の W3C 標準 WAI-ARIA 1.0 (www.w3.org/TR/wai-aria/) が、US Section 508 (www.access-board.gov/guidelines-and-standards/communications-and-it/about-the-section-508-standards/section-508-standards) および Web Content Accessibility Guidelines (WCAG) 2.0 (www.w3.org/TR/WCAG20/) に準拠するように使用されています。アクセシビリティ機能を利用するには、最新リリースのスクリーン・リーダーと、この製品によってサポートされる最新の Web ブラウザーを使用してください。

IBM Knowledge Center の製品資料は、アクセシビリティに対応しています。IBM Knowledge Center のアクセシビリティ機能については、Accessibility section of the IBM Knowledge Center help (www.ibm.com/support/knowledgecenter/about/releasesnotes.html?view=kc#accessibility) に記載されています。

キーボード・ナビゲーション

この製品は、標準のナビゲーション・キーを使用します。

インターフェース情報

ユーザー・インターフェースには、1 秒当たり 2 回から 55 回の点滅を行うコンテンツはありません。

Web ユーザー・インターフェースでは、コンテンツを正しくレンダリングするために、また使いやすさを実現するために、カスケーディング・スタイル・シートが使用されています。このアプリケーションには、視覚に障害のあるユーザーがシステム表示設定を使用するための、同等の方式 (ハイコントラスト・モードなど) が用意されています。フォント・サイズの制御は、デバイスまたは Web ブラウザーの設定を使用し行うことができます。

Web ユーザー・インターフェースには、アプリケーションの機能領域に素早くナビゲートできる WAI-ARIA ナビゲーション・ランドマークが含まれています。

ベンダー・ソフトウェア

IBM Spectrum Protect 製品ファミリーには、IBM の使用許諾契約書の対象とならないベンダー・ソフトウェアが含まれます。IBM は、それらの製品のアクセシビリティ機能を保証するものではありません。ベンダーの製品のアクセシビリティ機能については、ベンダーにお問い合わせください。

関連アクセシビリティ情報

IBM では、標準の IBM ヘルプ・デスクとサポート Web サイトに加えて、聴覚に障害のあるお客様が営業担当者やサポート・サービスに連絡が取れるように TTY 電話サービスを開設しています。

TTY サービス
800-IBM-3383 (800-426-3383)
(北アメリカ内)

IBM のアクセシビリティに対する取り組みについて詳しくは、[IBM Accessibility \(www.ibm.com/able\)](http://www.ibm.com/able) を参照してください。

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものです。この資料は、IBM から他の言語でも提供されている可能性があります。ただし、これを入手するには、本製品または当該言語版製品を所有している必要がある場合があります。

本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品とプログラムの操作またはサービスの評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒 103-8510

東京都中央区日本橋箱崎町 19 番 21 号

日本アイ・ビー・エム株式会社

法務・知的財産

知的財産権ライセンス 渉外

IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態を提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

IBM Director of Licensing

IBM Corporation

North Castle Drive, MD-NC119

Armonk, NY 10504-1785

US

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

本書に含まれるパフォーマンス・データは、特定の動作および環境条件下で得られたものです。実際の結果は、異なる可能性があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確認できません。IBM 以外の製品の性能に関する質問は、それらの製品の供給者をお願いします。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権使用許諾:

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・プラットフォームのアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。これらのサンプル・プログラムは特定物として現存するままの状態を提供されるものであり、いかなる保証も提供されません。IBM は、このサンプル・プログラムの使用から生ずるいかなる損害に対しても責任を負いません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生的創作物には、次のように、著作権表示を入れていただく必要があります。「© (お客様の会社名) (西暦年). このコードの一部は、IBM Corp. のサンプル・プログラムから取られています。」© Copyright IBM Corp. _年を入れる_.

商標

IBM、IBM ロゴ、および ibm.com[®] は、世界の多くの国で登録された International Business Machines Corp. の商標です。他の製品名およびサービス名等は、それぞれ IBM または各社の商標である場合があります。現時点での IBM の商標リストについては、www.ibm.com/legal/copytrade.shtml をご覧ください。

Adobe は、Adobe Systems Incorporated の米国およびその他の国における登録商標です。

Linear Tape-Open、LTO、および Ultrium は、HP、IBM Corp. および Quantum の米国およびその他の国における商標です。

Intel および Itanium は、Intel Corporation または子会社の米国およびその他の国における商標または登録商標です。

登録商標 Linux は、世界中で商標の所有者である Linux Torvalds の独占的ライセンシーである Linux Foundation のサブライセンスに従って使用されています。

Microsoft、Windows、および Windows NT は、Microsoft Corporation の米国およびその他の国における商標です。

Java[™] およびすべての Java 関連の商標およびロゴは Oracle やその関連会社の米国およびその他の国における商標または登録商標です。

UNIX は The Open Group の米国およびその他の国における登録商標です。

VMware、VMware vCenter Server、および VMware vSphere は VMware, Inc. または子会社の米国およびその他の国における登録商標または商標です。

製品資料に関するご使用条件

これらの資料は、以下のご使用条件に同意していただける場合に限りご使用いただけます。

適用条件

IBM Web サイトの「ご利用条件」に加えて、以下のご使用条件が適用されます。

個人使用

これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、非商業的な個人による使用目的に限り複製することができます。ただし、IBM の明示的な承諾をえずに、これらの資料またはその一部について、二次的著作物を作成したり、配布 (頒布、送信を含む) または表示 (上映を含む) することはできません。

商業的使用

これらの資料は、すべての著作権表示その他の所有権表示をしていただくことを条件に、お客様の企業内に限り、複製、配布、および表示することができます。ただし、IBM の明示的な承諾をえずにこれらの資料の二次的著作物を作成したり、お客様の企業外で資料またはその一部を複製、配布、または表示することはできません。

権利

ここで明示的に許可されているもの以外に、資料や資料内に含まれる情報、データ、ソフトウェア、またはその他の知的所有権に対するいかなる許可、ライセンス、または権利を明示的にも黙示的にも付与するものではありません。

資料の使用が IBM の利益を損なうと判断された場合や、上記の条件が適切に守られていないと判断された場合、IBM はいつでも自らの判断により、ここで与えた許可を撤回できるものとさせていただきます。

お客様がこの情報をダウンロード、輸出、または再輸出する際には、米国のすべての輸出入 関連法規を含む、すべての関連法規を遵守するものとします。

IBM は、これらの資料の内容についていかなる保証もしません。これらの資料は、特定物として現存するままの状態を提供され、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任なしで提供されます。

プライバシー・ポリシーに関する考慮事項

サービス・ソリューションとしてのソフトウェアも含めた IBM ソフトウェア製品 (「ソフトウェア・オファリング」) では、製品の使用に関する情報の収集、エンド・ユーザーの使用感の向上、エンド・ユーザーとの対話またはその他の目的のために、Cookie はじめさまざまなテクノロジーを使用することがあります。多くの場合、ソフトウェア・オファリングにより個人情報が収集されることはありません。IBM の「ソフトウェア・オファリング」の一部には、個人情報を収集できる機能を持つものがあります。ご使用の「ソフトウェア・オファリング」が、これらの Cookie およびそれに類するテクノロジーを通じてお客様による個人情報の収集を可能にする場合、以下の具体的事項をご確認ください。

この「ソフトウェア・オファリング」は、Cookie もしくはその他のテクノロジーを使用して個人情報を収集することはありません。

この「ソフトウェア・オファリング」が Cookie およびさまざまなテクノロジーを使用してエンド・ユーザーから個人を特定できる情報を収集する機能を提供する場合、お客様は、このような情報を収集するにあたって適用される法律、ガイドライン等を遵守する必要があります。これには、エンドユーザーへの通知や同意の要求も含まれますがそれらには限られません。

このような目的での Cookie などの各種テクノロジーの使用について詳しくは、「IBM プライバシー・ステートメント」 (<http://www.ibm.com/privacy/jp/ja/>)、「IBM オンライン・プライバシー・ステートメント」 (<http://www.ibm.com/privacy/details/jp/ja/>) の『クッキー、ウェブ・ビーコン、その他のテクノロジー』というタイトルのセクション、および「IBM Software Products and Software-as-a-Service Privacy Statement」 (<http://www.ibm.com/software/info/product-privacy>) を参照してください。

用語集

IBM Spectrum Protect 製品ファミリーの用語と定義が記載されている用語集を使用できます。

[IBM Spectrum Protect 用語集](#) を参照してください。

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。
なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

- アーカイブ・オブジェクト
 - 期限切れ [30](#)
 - 中断 [30](#)
 - 保留解除 [30](#)
- アーカイブ・コピー・グループ [28](#)
- アーカイブ・ファイル
 - 保存期間 [28](#)
- アクセシビリティ機能 [199](#)
- アクセス、オブジェクトへの
 - ユーザーによる [25](#)
- アクティブ・バージョン (active version)
 - 削除 [72](#)
- 圧縮 [42, 64](#)
- 圧縮タイプ
 - LZ4 [43](#)
 - LZW [43](#)
- アプリケーション・タイプ [17, 112, 116](#)
- アプリケーションのバージョン [ix](#)
- 暗号化
 - アプリケーションでの管理 [46](#)
 - インターオペラビリティ [77](#)
 - 透過的 [48](#)
 - 認証設定 [46](#)
- イベント
 - eventRetentionActivate [32](#)
- イベント・ベース
 - 保存ポリシー [32](#)
- イベント・ロギング [72](#)
- インターオペラビリティ
 - オペレーティング・システム [78](#)
 - 規則
 - UNIX または Linux [75](#)
 - Windows [75](#)
 - コマンド [77](#)
 - バックアップ/アーカイブ・クライアント [75](#)
 - API オブジェクトの命名 [75](#)
 - API オブジェクトへのアクセス [75](#)
- オブジェクト
 - オフにする [72](#)
 - 非活動コピー [41](#)
 - アクセス・ルール [24](#)
 - 活動コピー [41](#)
 - 期限切れサイクル [72](#)
 - 更新 [71](#)
 - サーバーから削除 [72](#)
 - 削除 [71](#)
 - バージョン制御 [41](#)
- オブジェクト ID の概要 [23](#)
- オブジェクト・タイプ [24](#)
- オブジェクトの選択
 - リストアするための [65](#)
- オブジェクトのソート
 - リストア順序による [65](#)

- オブジェクトの見積もり [41](#)
- オブジェクトの命名
 - インターオペラビリティ [75](#)
 - オブジェクト・タイプ [24](#)
 - 概要 [23](#)
 - 高位
 - オブジェクト名 [24](#)
 - 低位
 - オブジェクト名 [24](#)
 - ファイル・スペース名 [23](#)
 - dsmBindMC [24](#)
 - OS ごとの例 [24](#)
- オブジェクトへのアクセス
 - 複数ノード間 [25](#)
- オブジェクトをオフにする [72](#)
- オプション
 - 管理者によって設定された [2](#)
 - API にサポートされない [1](#)
 - compressalways [2](#)
 - enablearchiveretentionprotection [32](#)
 - errorlogretention [72](#)
 - fromnode [25](#)
 - fromowner [25](#)
 - passwordaccess [16, 111](#)
 - servername [2](#)
 - tcpbuffsize [38](#)
 - tcpnodelay [38](#)
 - tcpserveraddr [2](#)
- オプション・ストリング
 - API [2](#)
 - fromowner [25](#)
- オプション・ファイル
 - user [3](#)
- オプション・リスト
 - フォーマット [113, 116](#)
- オペレーティング・システムのインターオペラビリティ [78](#)

[カ行]

- 開始、セッションの [17](#)
- 活動コピー、オブジェクトの [41](#)
- 環境
 - API のセットアップ [3](#)
- 環境変数
 - オペレーティング・システムによる [3](#)
 - DSMI_CONFIG [3](#)
 - DSMI_DIR [3](#)
 - DSMI_LOG [4](#)
- 関数定義、API [191, 195](#)
- 関数呼び出し
 - 簡略説明 [83](#)
- 管理クラス
 - オブジェクトの関連付け [28](#)
 - 照会 [29](#)
 - ファイルへのバインドおよび再バインド [29](#)
 - dsmBindMC によって割り当てられる [28](#)

- 管理者オプション [2](#)
- 管理ユーザー
 - 作成 [22](#)
- キーボード [199](#)
- 許可規則
 - dsmDeleteAccess 関数 [95](#)
- 許可ユーザー (authorized user) [21](#), [24](#)
- クライアント・サイドのデータ重複排除 [50](#)
- クライアント所有者権限 [22](#)
- クライアント・ノード・プロキシ・サポート [78](#)
- クライアント・パフォーマンス・モニター [38](#)
- クライアント・パフォーマンス・モニター・オプション
 - PERFCOMMTIMEOUT [40](#)
 - PERFMONTCPPORT [40](#)
 - PERFMONTCPSERVERADDRESS [39](#)
- グループ・リーダー [61](#)
- 高位修飾子 [75](#)
- 高位名
 - dsmRenameObj 関数 [128](#)
- 構成ソース
 - 優先順位の順序 [2](#)
- 構成ファイル
 - API [3](#)
- 構造
 - サイズ限界 [14](#), [36](#)
 - qMCData [35](#)
 - qryRespBackupData [33](#)
 - qryRespFSData 関数 [25](#)
- 高速パス [33](#)
- 高速パス照会 [87](#)
- コード・ページ [81](#)
- 互換性
 - API のバージョン間の [15](#)
- コピー・グループ [28](#)
- コマンド
 - makemtu [81](#)
- コンパイル
 - ユニコード [81](#)

[サ行]

- サーバー
 - からのオブジェクトの削除 [72](#)
- サーバー・サイド・データ重複排除 [53](#)
- サイズ限界
 - API データ構造 [14](#), [36](#)
- サイズ見積もり [41](#)
- サインオン・プロセス [18](#)
- サンプル・アプリケーション
 - callmt1.c [16](#)
- サンプル・コード
 - dsmgrp.c [63](#)
- シグナルの使用 [17](#)
- シグナル・ハンドラー [17](#)
- システムの照会 [33](#)
- 自動化されたクライアント・フェイルオーバー [53](#)
- 終了、セッションの [17](#)
- 終了、セッションのにより
 - dsmTerminate [18](#)
- 受信、サーバーからのデータの
 - 一般的な説明 [63](#)
 - 手順 [64](#)
 - 部分オブジェクト・リストア/リトリブ [64](#)
- 照会、システムの [33](#)

- 状態
 - InSession [119](#)
- 状態遷移
 - バックアップおよびアーカイブの例 [56](#)
 - リストアとリトリブ [68](#)
- 除外オブジェクト [23](#)
- 所有者権限 [22](#)
- 所有者名
 - NULL [24](#)
- 資料 [vii](#)
- 身体障害 [199](#)
- ストレージ・プール
 - 同時書き込み操作 [37](#)
- 制約事項
 - バッファ・コピー除去を使用する場合の暗号化と圧縮 [45](#)
 - マルチスレッド化 [16](#)
- セキュリティ [18](#)
- 設計上の推奨事項 [11](#)
- セッション
 - セキュリティ [18](#)
 - パスワード
 - セッション [18](#)
 - dsmInitEx による開始 [17](#)
- 送信、サーバーへのデータの [36](#)

[タ行]

- ターゲット・ノードと通常のノード [79](#)
- 低位修飾子 [75](#)
- 低位名
 - dsmRenameObj 関数 [128](#)
- 停止、セッションの [17](#)
- データ構造
 - サイズ限界 [14](#), [36](#)
 - バージョン制御 [15](#)
- データ重複排除 [49](#)
- データ重複排除のファイル
 - 組み込み [53](#)
 - exclude [53](#)
- データ重複排除のファイルの組み込み [53](#)
- データ重複排除のファイルの除外 [53](#)
- データ転送
 - LAN フリー [37](#)
- データの送信
 - 非ユニコード・ファイル・スペースへの [81](#)
- データ保護 [32](#)
- データ保存 [32](#)
- データを保管するためのポリシー [28](#)
- 同時書き込み操作
 - ストレージ・プール [37](#)
- トランザクション・モデル
 - dsmBeginTxn 関数 [91](#)

[ナ行]

- ノード
 - 管理クラスの照会 [29](#)
 - クライアント・プロキシ・サポートによる [78](#)
 - 権限 [72](#)
 - 所有者間のアクセス [25](#)
 - 名前 [11](#)
- ノードのレプリカの生成 [53](#)

[ハ行]

- バージョン
 - 保存されたファイル [28](#)
- バージョン制御
 - バックアップ・コピーの管理 [41](#)
 - API データ構造 [15](#)
 - dsmQueryApiVersionEx の使用 [15](#)
- パス情報
 - インターオペラビリティ [75](#)
- パスの例
 - OS ごと [24](#)
- バックアップ
 - クライアント・ノード・プロキシの使用 [78](#)
 - 複数のノード [78](#)
- バックアップ、オブジェクトの [41](#)
- バックアップ/アーカイブ・クライアント
 - インターオペラビリティ [75](#)
- バックアップ・コピー・グループ [28](#)
- バッファ・コピー除去
 - 概要 [44](#)
 - リストアとリトリブ [44](#)
- バッファ・コピー除去を使用する場合の暗号化と圧縮 [45](#)
- パフォーマンスの考慮事項
 - dsmSendData 関数 [38](#)
- パフォーマンス・モニター
 - クライアント [38](#)
- 非活動コピー、オブジェクトの [41](#)
- ファイル
 - オブジェクト・タイプ [24](#)
 - オプション [1](#)
 - 構成 [1](#)
- ファイル・システムの管理
 - dsmDeleteFS [26](#)
- ファイル集合 [37](#)
- ファイル・スペース
 - 管理 [25](#)
 - 削除 [25](#)
 - 登録 [25](#)
 - 非ユニコード [81](#)
 - capacity [25](#)
- ファイル・スペースの管理
 - dsmUpdateFS [26](#)
- ファイル・スペースの登録 [25](#)
- ファイル・スペース名
 - 概要 [23](#)
 - ファイル集合 [37](#)
- ファイルのグループ化 [61](#)
- フェイルオーバー
 - 概要 [53](#)
 - 状況情報 [54](#)
- 複製状況 [54](#)
- 部分オブジェクト・リストア/リトリブ [64](#)
- ヘッダー・ファイル
 - dsmapifp.h [191](#)
 - dsmrc.h [145](#)
 - tmapifp.h [195](#)
- ヘッダー・ファイル、dmapips.h [155](#)
- ヘッダー・ファイル、dmapitd.h [155](#)
- ヘッダー・ファイル、release.h [155](#)
- ヘッダー・ファイル、tmapitd.h [155](#)
- 包含オブジェクト [23](#)
- 保存保護 [31](#)
- ポリシー

ポリシー (続き)
保存ポリシー [32](#)

[マ行]

- マルチスレッド化
 - 概要 [16](#)
 - 制約事項 [16](#)
 - フラグ [11](#)
 - マルチスレッド・オプション [16](#)
 - mtflag 値 [16](#)
- メタデータ
 - オブジェクトの命名 [23](#)
- メッセージ
 - dsmRCMsg 関数 [125](#)
- 文字セット [81](#)
- 戻りコード
 - ソース・ヘッダー・ファイル [145](#)
 - dsmRCMsg での取得 [125](#)

[ヤ行]

- ユニコード
 - セットアップ [81](#)
 - 非ユニコードのファイル・スペース [81](#)
 - mbcs [81](#)
 - Windows [81](#)

[ラ行]

- リストア (restore)
 - サーバーからのオブジェクト [63](#)
- リトリブ
 - サーバーからのオブジェクト [63](#)
- ロギング、イベントの [72](#)

[数字]

- 128 ビット AES 暗号化のサポート [46](#)
- 256 ビット AES 暗号化サポート [46](#)
- 64 ビット
 - コンパイル [1](#)
 - 要件 [1](#)

A

- API
 - ユニコードの使用 [81](#)
 - 概要 [1](#)
 - 環境のセットアップ [3](#)
 - サンプル・アプリケーション [5](#)
 - dsmInitEx
 - によって使用される構成ファイル [3](#)
 - dsmInitEx によって使用されるオプション・ストリング [2](#)
- API オプション・リスト
 - dsmInitEx が使用する [17](#)
- API 構成ファイル
 - dsmInitEx が使用する [17](#)
- API サンプル・アプリケーション
 - callbuff [5](#)
 - callbuff - データ・バッファ [5](#)
 - callevnt [5](#)

API サンプル・アプリケーション (続き)

- [callevnt - イベント・ベース保存 5](#)
- [callhold 5](#)
- [callhold - 保存保留 5](#)
- [callmt* 5](#)
- [callmt* - マルチスレッド API サンプル・アプリケーション 5](#)
- [callmtu1.c 81](#)
- [callmtu2.c 81](#)
- [callret 5](#)
- [callret - データ保存保護 API サンプル・アプリケーション 5](#)
- [dapi* 5](#)
- [dapi* - 対話式、単スレッド 5](#)
- [dsmgrp 5](#)
- [dsmgrp* - オブジェクトのグループ化サンプル 5](#)
- [UNIX または Linux 5](#)
- [Windows 64 ビット 7](#)

[archiveretentionprotection 31](#)

[asnodename 78](#)

C

[callbuff](#)

- [IBM Spectrum Protect データ・バッファ・サンプル API アプリケーション 5](#)

[callevnt](#)

- [イベント・ベース保存 5](#)

[callhold](#)

- [保存保留 API サンプル・アプリケーション 5](#)

[callmt*](#)

- [マルチスレッド API サンプル・アプリケーション 5](#)

[callmt1.c](#)

- [サンプル 16](#)

[callret](#)

- [データ保存保護 API サンプル・アプリケーション 5](#)

[capacity](#)

- [ファイル・スペース 25](#)

[compressalways](#)

- [オプション 7](#)

[CTRL+C 17](#)

D

[dapi*](#)

- [単スレッド、対話式 API サンプル・アプリケーション 5](#)

[DB Chg 操作 11](#)

[DBCS 81](#)

[delete archive 77](#)

[delete filespace 77](#)

[dir](#)

- [オブジェクト・タイプ 24](#)

[dscenu.txt 3](#)

[dserror.log 4](#)

[DSM_MAX_PLATFORM_LENGTH 17](#)

[dsm.opt](#)

- [asnodename オプション 78](#)
- [enablearchiveretentionprotection 31](#)
- [encryptkey 46](#)

[dsm.sys](#)

- [asnodename オプション 78](#)
- [enablearchiveretentionprotection 31](#)

dsm.sys (続き)

- [encryptkey 46](#)

[dsmapifp.h](#)

- [ヘッダー・ファイル 83, 191](#)

[dsmapiips.h](#) ヘッダー・ファイル [155](#)

[dsmapiitd.h](#)

- [ヘッダー・ファイル 121](#)

[dsmapiitd.h](#) ヘッダー・ファイル [155](#)

[dsmApiVersion](#) 関数

- [セッション 17](#)

[dsmBeginGetData](#) 関数

- [dsmEndGetData](#) 関数 [98](#)
- [dsmTerminate](#) 関数 [98](#)
- [フローチャート内の 69](#)
- [概要 85](#)
- [構文 86](#)
- [コーディング例 70](#)
- [状態 遷移 72](#)
- [状態遷移 68](#)
- [バッファ管理 44](#)
- [戻りコード 86](#)

[dsmBeginQuery](#) 関数

- [フローチャート 33](#)
- [概要 87](#)
- [管理クラス 28](#)
- [構文 87](#)
- [照会 33](#)
- [照会例 35](#)
- [状態 遷移 72](#)
- [状態遷移 33](#)
- [データの受信 65](#)
- [データの送信例 36](#)
- [戻りコード 91](#)
- [dsmEndQuery](#) 関数 [99](#)
- [dsmGetNextQObj](#) 関数 [106](#)

[dsmBeginTxn 25](#)

[dsmBeginTxn](#) 関数

- [削除 30](#)
- [保存ポリシー 32](#)
- [オブジェクトの削除 72](#)
- [概要 91](#)
- [期限切れ 30](#)
- [構文 92](#)
- [コーディング例 59](#)
- [状態 遷移 72](#)
- [トランザクション・モデル 36](#)
- [バッファ・コピー除去 44](#)
- [戻りコード 92](#)
- [dsmEndTxn](#) 関数 [101](#)
- [dsmRenameObj](#) 関数 [128](#)
- [dsmRetentionEvent](#) 関数 [130](#)

[dsmBindMC](#)

- [例 29](#)

[dsmBindMC](#) 関数

- [dsmSendObj](#) 関数 [134](#)
- [一般的な説明 59](#)
- [オブジェクト名 24](#)
- [概要 92](#)
- [管理クラス 29](#)
- [構文 92](#)
- [コーディング例 59](#)
- [状態 遷移 72](#)
- [バッファ・コピー除去 44](#)
- [戻される情報 28](#)

dsmBindMC 関数 (続き)
 [戻りコード 93](#)
 [include-exclude リスト 28](#)
 dsmChangePW
 [一般的な説明 73](#)
 dsmCleanUp 関数
 [概要 94](#)
 [構文 94](#)
 [信号 17](#)
 [マルチスレッド化 16](#)
 [dsmSetUp 関数 138](#)
 dsmclientV3.cat 3
 dsmDeleteAccess 関数
 [アクセス、オブジェクトへの 25](#)
 [概要 95](#)
 [構文 95](#)
 dsmDeleteFS 関数
 [コーディング例 25](#)
 [概要 95](#)
 [構文 96](#)
 [状態 遷移 72](#)
 [ファイル・システムの管理 26](#)
 [ファイル・スペース 25](#)
 [戻りコード 96](#)
 dsmDeleteObj 関数
 dsmSendObj 関数
 [管理クラス 11](#)
 [オブジェクト 41](#)
 [オブジェクトの削除 72](#)
 [オブジェクトの命名 11](#)
 [概要 96](#)
 [構文 97](#)
 [状態 遷移 72](#)
 [戻りコード 97](#)
 [dsmEndTxn 関数 101](#)
dsmEndGetData
 [処理の停止 67](#)
 dsmEndGetData 関数
 [フローチャート内の 69](#)
 [概要 98](#)
 [構文 98](#)
 [コーディング例 70](#)
 [状態 遷移 72](#)
 [状態遷移 68](#)
 [バッファ管理 44](#)
 [LAN フリー 37](#)
 dsmEndGetDataEx 関数
 [概要 98](#)
 [構文 98](#)
 dsmEndGetObj 関数
 [dsmBeginGetData 関数 85](#)
 [フローチャート内の 69](#)
 [概要 99](#)
 [構文 99](#)
 [コーディング例 70](#)
 [状態 遷移 72](#)
 [状態遷移 68](#)
 [バッファ管理 44](#)
 [戻りコード 99](#)
 dsmEndQuery
 [一般的な説明 33](#)
 dsmEndQuery 関数
 [フローチャート 33](#)
 [概要 99](#)
 dsmEndQuery 関数 (続き)
 [構文 99](#)
 [サーバーへの照会 65](#)
 [状態 遷移 72](#)
 [状態遷移 33](#)
 [dsmGetNextQObj 関数 106](#)
 dsmEndSendObj 関数
 [dsmSendData 関数 132](#)
 [dsmSendObj 関数 134](#)
 [フローチャート 57](#)
 [状態遷移 56](#)
 [オブジェクトの送信 41](#)
 [概要 100](#)
 [構文 100](#)
 [コーディング例 59](#)
 [状態 遷移 72](#)
 [戻りコード 100](#)
 [dsmEndTxn 関数 101](#)
 dsmEndTxn 関数
 [dsmSendObj 関数 134](#)
 [フローチャート 57](#)
 [オブジェクトの削除 72](#)
 [概要 101](#)
 [構文 101](#)
 [コーディング例 59](#)
 [状態 遷移 72](#)
 [状態遷移 56](#)
 [同時書き込み操作 37](#)
 [トランザクション・モデル 36](#)
 [バッファ・コピー除去 44](#)
 [ファイルのグループ化 61](#)
 [戻りコード 102](#)
 [dsmEndTxnEx 関数 103](#)
 [dsmRenameObj 関数 128](#)
 [dsmRetentionEvent 関数 130](#)
 dsmEndTxnEx 関数
 [概要 103](#)
 [構文 103](#)
 [ファイルのグループ化 61](#)
 [戻りコード 103](#)
 dsmEventType 関数
 [保存ポリシー 32](#)
 dsmGetBufferData 関数
 [概要 105](#)
 [構文 105](#)
 [戻りコード 105](#)
 dsmGetData 67
 dsmGetData 関数
 [フローチャート内の 69](#)
 [概要 104](#)
 [構文 104](#)
 [コーディング例 70](#)
 [状態 遷移 72](#)
 [状態遷移での 68](#)
 [戻りコード 104](#)
 dsmGetDataEx 関数
 [dsmReleaseBuffer 関数 127](#)
 [dsmRequestBuffer 関数 129](#)
 dsmGetList 関数
 [dsmGetObj 関数 109](#)
 dsmGetNextObj
 [dsmDeleteObj 関数 96](#)
dsmGetNextQObj
 [dsmEndQuery 関数 99](#)

dsmGetNextQObj 関数
 フローチャート [33](#)
 概要 [106](#)
 構文 [106](#)
 照会例 [35](#)
 状態 遷移 [72](#)
 状態遷移 [33](#)
 戻りコード [108](#)
 dsmRetentionEvent 関数 [130](#)

dsmGetObj
 オブジェクトの受信 [67](#)

dsmGetObj 関数
 dsmBeginGetData 関数 [85](#)
 フローチャート内の [69](#)
 概要 [109](#)
 構文 [109](#)
 コーディング例 [70](#)
 状態 遷移 [72](#)
 状態遷移 [68](#)
 戻りコード [110](#)
 dsmEndGetObj 関数 [99](#)
 dsmGetData 関数 [104](#)

dsmGroupHandler 関数
 概要 [110](#)
 構文 [110](#)
 ファイルのグループ化 [61](#)
 戻りコード [111](#)
 dsmEndTxnEx 関数 [103](#)

dsmgrp.c [63](#)

dsmgrp*
 論理オブジェクトのグループ化 API サンプル・アプリケーション [5](#)

dsmHandle [123](#), [124](#)

dsmHandle 関数
 セッション [17](#)

DSMI_CONFIG 環境変数 [3](#)

DSMI_DIR
 環境変数 [7](#)

DSMI_DIR 環境変数 [3](#)

DSMI_LOG 環境変数 [4](#)

dsmInit 関数
 概要 [111](#)
 構文 [111](#)
 保存保護 [31](#)
 戻りコード [113](#)

dsmInitEx 関数
 dsmChangePW 関数 [93](#)
 dsmEndGetData 関数 [98](#)
 暗号化 [46](#)
 インターオペラビリティ [78](#)
 オプション・ストリング [2](#)
 オプションの指定 [2](#)
 概要 [115](#)
 管理ユーザー [22](#)
 構文 [115](#)
 状態 遷移 [72](#)
 セッション [17](#)
 セッション所有者の設定 [24](#)
 セッション・セキュリティ [18](#)
 セッションの開始 [17](#)
 保存保護 [31](#)
 マルチスレッド化 [16](#)
 戻りコード [117](#)
 有効期限切れのパスワード [18](#)

dsmInitEx 関数 (続き)
 asnodename オプション [78](#)
 dsmGetBufferData 関数 [105](#)
 dsmGetNextQObj 関数 [106](#)
 dsmLogEvent 関数 [119](#)
 dsmQueryCliOptions 関数 [122](#)
 dsmQuerySessOptions [124](#)
 dsmReleaseBuffer 関数 [127](#)
 dsmSetUp 関数 [138](#)

dsmIntitEx 関数
 dsmQuerySessInfo 関数 [123](#)

dsmLogEvent 関数
 概要 [119](#)
 構文 [119](#)
 戻りコード [119](#)

dsmLogEventEx 関数
 概要 [119](#)
 構文 [120](#)
 戻りコード [120](#)

dsmQuery 関数
 複数のノード [78](#)

dsmQueryAccess 関数
 概要 [121](#)
 dsmDeleteAccess 関数 [95](#)

dsmQueryApiVersionEx 関数
 概要 [122](#)
 構文 [122](#)
 バージョン制御 [15](#)

dsmQueryAPIVersionEx 関数
 マルチスレッド化 [16](#)

dsmQueryCliOptions 関数
 概要 [122](#)
 構文 [123](#)
 セッション [17](#)
 dsmQuerySessOptions [124](#)

dsmQuerySessInfo
 dsmDeleteFS 関数 [95](#)

dsmQuerySessOptions 関数
 概要 [124](#)
 構文 [124](#)

dsmrc.h
 ヘッダー・ファイル [145](#)

dsmRegisterFS 関数
 コーディング例 [25](#)
 概要 [126](#)
 構文 [126](#)
 状態 遷移 [72](#)
 ファイル・スペース [25](#)
 戻りコード [127](#)

dsmReleaseBuffer 関数
 概要 [127](#)
 構文 [127](#)
 戻りコード [128](#)
 dsmGetBufferData 関数 [105](#)
 dsmReleaseBuffer 関数 [127](#)
 dsmRequestBuffer 関数 [129](#)
 dsmSendBufferData 関数 [131](#)

dsmRenameObj 関数
 概要 [128](#)
 構文 [128](#)
 戻りコード [129](#)

dsmRequestBuffer 関数
 概要 [129](#)
 構文 [130](#)

dsmRequestBuffer 関数 (続き)
バッファ・コピー除去 [44](#)
戻りコード [130](#)

dsmRetentionEvent 関数
削除 [30](#)
保存ポリシー [32](#)
概要 [130](#)
期限切れ [30](#)
構文 [131](#)
戻りコード [131](#)

dsmSendBufferData 関数
概要 [131](#)
構文 [132](#)
バッファ・コピー除去 [44](#)
戻りコード [132](#)

dsmSendObj
保存ポリシー [32](#)

dsmSendType 関数
オブジェクトの更新 [71](#)

dsmSetAccess 関数
アクセス、オブジェクトへの [25](#)
概要 [137](#)
構文 [137](#)
戻りコード [137](#)

dsmSetUp 関数
概要 [138](#)
構文 [138, 139](#)
マルチスレッド [16](#)
マルチスレッド化 [16, 37](#)
LAN フリー [11, 37](#)
passwordaccess [21](#)

dsmTerminate [67](#)

dsmTerminate 関数
一般的な説明 [18](#)
概要 [140](#)
構文 [140](#)
状態 遷移 [72](#)
信号 [17](#)
セッション [17](#)
バッファ [44](#)
バッファ・コピー除去 [44](#)
dsmInit 関数 [111](#)
dsmReleaseBuffer 関数 [127](#)
dsmRequestBuffer 関数 [129](#)
dsmSetUp 関数 [138](#)

dsmUpdateFS 関数
コーディング例 [25](#)
概要 [140](#)
構文 [140](#)
状態 遷移 [72](#)
ファイル・スペース [25](#)
ファイル・スペースの管理 [26](#)
戻りコード [141](#)

dsmUpdateObj 関数
概要 [141](#)
管理クラスの変更 [28](#)
構文 [141](#)
戻りコード [142](#)

dsmUpdateObject(Ex) 関数
オブジェクトの更新 [71](#)

dsmUpdateObjEx 関数
概要 [142](#)
管理クラスの変更 [28](#)
構文 [143](#)

dsmUpdateObjEx 関数 (続き)
戻りコード [144](#)

E

enablearchiveretentionprotection
dsm.opt [31](#)
dsm.sys [31](#)
encryptkey [46](#)
envSetUp [139](#)
errorlogretention
使用する時期 [72](#)
eventRetentionActivate イベント [32](#)

F

fromowner オプション [25](#)

H

HP スレッド・スタック [16](#)

I

IBM Knowledge Center [vii](#)
include-exclude
ファイル [140](#)
include-exclude リスト [28, 82](#)
InSession 状態 [119](#)

K

Knowledge Center [vii](#)

L

LAN フリー
データ転送 [37](#)
dsmEndGetDataEX 関数 [98](#)
dsmSetUp 関数 [11](#)
LZ4 圧縮 [43](#)

M

makemtu [81](#)
mbcs [81](#)

N

NULL
バックアップ・グループまたはアーカイブ・グループ
[28](#)

P

passwordaccess
オプション [11, 46](#)
generate [140](#)
passwordaccess prompt [18](#)
passwordaccess オプション
マルチスレッド化 [16](#)

passwordaccess オプション (続き)
 dsmInit 関数 [111](#)
 generate [18](#)
 userNamePswd 値 [22](#)
passworddir オプション
 dsm.sys [21](#)
PERFMONCOMMTIMEOUT [40](#)
PERFMONTCPPORT [40](#)
PERFMONTCPSERVERADDRESS [39](#)
proxynode [79](#)

Q

qMCData 構造 [35](#)
qryRespArchiveData [31](#)
qryRespBackupData
 dsmDeleteObj 関数 [96](#)
qryRespBackupData 構造 [33](#)
query
 クライアント・プロキシ・ノード権限を持つノード
 [78](#)
 コマンド [77](#)
 actlog [119](#)

R

rcApiOut
 例、詳細 [18](#)
rcApiOut 関数
 セッション [17](#)
release.h ヘッダー・ファイル [155](#)

S

servername [2](#)
set access [77](#)

T

TCPport [19](#)
TCPserver アドレス [19](#)
tcpserveraddr [2](#)
tmapifp.h [81](#)
tmapifp.h ヘッダー・ファイル [195](#)
tmapitd.h [81](#)
tmapitd.h ヘッダー・ファイル [155](#)

U

UNIX または Linux
 API サンプル・アプリケーション [5](#)
user
 介入 [17](#)

W

Windows 64 ビット
 サンプル・アプリケーション [7](#)

[特殊文字]

2 バイト文字セット [81](#)

dsmChangePW 関数
 概要 [93](#)
 構文 [94](#)
 状態 遷移 [72](#)
 セッション・セキュリティー [18](#)
 戻りコード [94](#)
dsmEndSendObjEx 関数
 圧縮 [42](#)
 暗号化 [46](#)
 概要 [100](#)
 構文 [100](#)
 戻りコード [101](#)
 LAN フリー [37](#)
dsmQueryApiVersion 関数
 概要 [121](#)
 構文 [122](#)
 状態 遷移 [72](#)
dsmQuerySessInfo 関数
 一般的な説明 [18](#)
 概要 [123](#)
 構文 [124](#)
 状態 遷移 [72](#)
 トランザクション・モデル [36](#)
 戻りコード [124](#)
 dsmRetentionEvent 関数 [130](#)
dsmRCMsg 関数
 概要 [125](#)
 構文 [126](#)
 戻りコード [126](#)
dsmSendData 関数
 dsmSendObj 関数 [134](#)
 フローチャート [57](#)
 圧縮 [42](#)
 オブジェクトの送信 [41](#)
 概要 [132](#)
 構文 [133](#)
 コーディング例 [59](#)
 状態 遷移 [72](#)
 状態遷移 [56](#)
 パフォーマンス [38](#)
 マルチスレッド化 [16](#)
 戻りコード [133](#)
 dsmEndSendObj 関数 [100](#)
 dsmEndTxn 関数 [101](#)
dsmSendObj 関数
 フローチャート [57](#)
 保存ポリシー [32](#)
 アクセス、オブジェクトへの [24](#)
 圧縮 [42](#)
 オブジェクトの削除 [72](#)
 オブジェクトの送信 [41](#)
 オブジェクトの命名 [11](#)
 概要 [134](#)
 構文 [134](#)
 コーディング例 [59](#)
 コピー・グループ [28](#)
 状態 遷移 [72](#)
 状態遷移での [56](#)
 バックアップ・コピー・グループ [28](#)
 dsmEndTxn 関数 [101](#)
オブジェクト ID 値 [11](#)
推奨事項
 dsmGetObject
 大規模データ [109](#)

推奨事項 (続き)

HP スレッド・スタックの設定 [16](#)

フローチャート

バックアップおよびアーカイブの例 [56](#)

リストアとリトリーブ [68](#)

LZW 圧縮 [43](#)



プログラム番号: 5725-W98
5725-W99
5725-X15