IBM Spectrum Protect
Version 8.1.0

Utilisation de l'API

IBM Spectrum Protect
Version 8.1.0

Utilisation de l'API

Important
Avant d'utiliser le présent document et le produit associé, prenez connaissance des informations générales figurant à la section «Remarques», à la page 219.

La présente édition s'applique à la version 8.1.0 d'IBM Spectrum Protect (numéros de produit 5725-W98, 5725-W99 et 5725-X15), ainsi qu'à toutes les éditions et modifications ultérieures, sauf mention contraire dans les nouvelles éditions.

LE PRESENT DOCUMENT EST LIVRE EN L'ETAT SANS AUCUNE GARANTIE EXPLICITE OU IMPLICITE. IBM DECLINE NOTAMMENT TOUTE RESPONSABILITE RELATIVE A CES INFORMATIONS EN CAS DE CONTREFACON AINSI QU'EN CAS DE DEFAUT D'APTITUDE A L'EXECUTION D'UN TRAVAIL DONNE.

Ce document est mis à jour périodiquement. Chaque nouvelle édition inclut les mises à jour. Les informations qui y sont fournies sont susceptibles d'être modifiées avant que les produits décrits ne deviennent eux-mêmes disponibles. En outre, il peut contenir des informations ou des références concernant certains produits, logiciels ou services non annoncés dans ce pays. Cela ne signifie cependant pas qu'ils y seront annoncés.

Pour plus de détails, pour toute demande d'ordre technique, ou pour obtenir des exemplaires de documents IBM, référez-vous aux documents d'annonce disponibles dans votre pays, ou adressez-vous à votre partenaire commercial.

Vous pouvez également consulter les serveurs Internet suivants :

- http://www.fr.ibm.com (serveur IBM en France)
- http://www.ibm.com/ca/fr (serveur IBM au Canada)
- http://www.ibm.com (serveur IBM aux Etats-Unis)

Compagnie IBM France
Direction Qualité
17, avenue de l'Europe
92275 Bois-Colombes Cedex

Table des matières

Avis aux lecteurs canadiens

Le présent document a été traduit en France. Voici les principales différences et particularités dont vous devez tenir compte.

Illustrations

Les illustrations sont fournies à titre d'exemple. Certaines peuvent contenir des données propres à la France.

Terminologie

La terminologie des titres IBM peut différer d'un pays à l'autre. Reportez-vous au tableau ci-dessous, au besoin.

IBM France	**IBM Canada**
ingénieur commercial	représentant
agence commerciale	succursale
ingénieur technico-commercial	informaticien
inspecteur	technicien du matériel

Claviers

Les lettres sont disposées différemment : le clavier français est de type AZERTY, et le clavier français-canadien de type QWERTY.

OS/2 et Windows - Paramètres canadiens

Au Canada, on utilise :

- les pages de codes 850 (multilingue) et 863 (français-canadien),
- le code pays 002,
- le code clavier CF.

Nomenclature

Les touches présentées dans le tableau d'équivalence suivant sont libellées différemment selon qu'il s'agit du clavier de la France, du clavier du Canada ou du clavier des États-Unis. Reportez-vous à ce tableau pour faire correspondre les touches françaises figurant dans le présent document aux touches de votre clavier.

France	Canada	Etats-Unis
(Pos1)		Home
Fin	Fin	End
(PgAr)		PgUp
(PgAv)		PgDn
Inser	Inser	Ins
Suppr	Suppr	Del
Echap	Echap	Esc
Attn	Intrp	Break
Impr écran	ImpEc	PrtSc
Verr num	Num	Num Lock
Arrêt défil	Défil	Scroll Lock
(Verr maj)	FixMaj	Caps Lock
AltGr	AltCar	Alt (à droite)

Brevets

Il est possible qu'IBM détienne des brevets ou qu'elle ait déposé des demandes de brevets portant sur certains sujets abordés dans ce document. Le fait qu'IBM vous fournisse le présent document ne signifie pas qu'elle vous accorde un permis d'utilisation de ces brevets. Vous pouvez envoyer, par écrit, vos demandes de renseignements relatives aux permis d'utilisation au directeur général des relations commerciales d'IBM, 3600 Steeles Avenue East, Markham, Ontario, L3R 9Z7.

Assistance téléphonique

Si vous avez besoin d'assistance ou si vous voulez commander du matériel, des logiciels et des publications IBM, contactez IBM direct au 1 800 465-1234.

A propos de cette publication

Cette publication fournit des informations qui vont vous aider dans les tâches suivantes :

- Ajout d'appels d'API IBM Spectrum Protect à une application existante
- Ecriture de programmes à l'aide d'interfaces de programmation à usage général obtenant des services de IBM Spectrum Protect

Outre l'interface de programme d'application (API), les programmes suivants sont fournis avec plusieurs systèmes d'exploitation :

- Un programme client de sauvegarde-archivage permettant de sauvegarder et d'archiver des fichiers de votre poste de travail ou serveur de fichiers sur l'espace de stockage puis de restaurer et récupérer les versions sauvegardées et les copies archivées des fichiers sur vos systèmes de fichiers locaux.
- Un client Web de sauvegarde-archivage permettant à un administrateur autorisé, à une personne chargée de l'assistance ou à un utilisateur final de sauvegarder, archiver et récupérer des fichiers via un navigateur sur une machine distante.
- Un programme client d'administration auquel vous pouvez accéder à partir d'un navigateur Web ou de la ligne de commande. Un administrateur peut ainsi contrôler et gérer les activités du serveur, définir les règles de gestion de la mémoire pour la sauvegarde, l'archivage et les services de gestion de l'espace, et configurer des programmes pour exécuter ces services à intervalles réguliers.

Public visé

Vous trouverez dans cette publication des instructions vous permettant d'ajouter des appels d'API à une application existante. De bonnes connaissances du langage C et des fonctions de IBM Spectrum Protect sont également nécessaires.

Publications

La famille de produits IBM Spectrum Protect inclut IBM Spectrum Protect Snapshot, IBM Spectrum Protect for Space Management, IBM Spectrum Protect for Databases et plusieurs autres produits de gestion de l'espace de stockage IBM®.

Pour consulter la documentation des produits IBM, accédez au site IBM Knowledge Center.

Conventions utilisées dans cette publication

La présente publication utilise les conventions typographiques suivantes :

Exemple	Description
autoexec.ncf hsmgui.exe	Une suite de lettres minuscules avec suffixe indique un nom de fichier programme.
DSMI_DIR	Une série de lettres majuscules indique des codes retour, ainsi que d'autres valeurs.
dsmQuerySessInfo	Les caractères gras indiquent une commande entrée dans une ligne de commande, le nom d'un appel de fonction, le nom d'une structure, une zone présente dans une structure ou un paramètre.

Exemple	Description
timeformat	Les caractères gras en italique indiquent une option de client de sauvegarde-archivage. Les caractères gras sont utilisés pour introduire l'option, ou dans un exemple.
dateformat	Les caractères italiques indiquent une option, la valeur d'une option, un nouveau terme, un marqueur pour des informations que vous fournissez ou une mise en évidence spéciale dans le texte.
`maxcmdretries`	Une suite de caractères sans espacement indique des fragments de programme ou des informations telles qu'elles peuvent s'afficher sur un écran (une commande par exemple).
signe plus (+)	Un signe plus placé entre deux touches indique que vous devez appuyer simultanément sur ces touches.

Nouveautés de la version 8.1.0

IBM Spectrum Protect version 8.1.0 contient de nouvelles fonctions et mises à jour.

Pour la liste des nouvelles fonctions et mises à jour de cette édition, consultez Mises à jour API.

Chapitre 1. Présentation générale de l'API

L'interface de programme d'application (API) de IBM Spectrum Protect permet à un client d'application d'utiliser des fonctions de gestion de stockage.

L'API comprend des appels de fonction que vous pouvez utiliser dans une application pour effectuer les opérations suivantes :
- Lancer ou fermer une session
- Attribuer des classes de gestion à des objets avant qu'ils ne soient stockés sur un serveur
- Sauvegarder ou archiver des objets sur un serveur
- Restaurer ou récupérer des objets à partir d'un serveur
- Interroger le serveur afin d'obtenir des informations sur les objets stockés
- Gérer les espaces fichier
- Envoyer les événements de conservation

Lorsque vous installez l'API en tant que développeur d'applications, vous recevez les fichiers dont a besoin l'utilisateur final d'une application :
- La bibliothèque partagée de l'API.
- Le fichier des messages.
- Les exemples de fichier d'options client.
- Le code source des fichiers d'en-tête de l'API dont votre application a besoin.
- Le code source pour un modèle d'application et le fichier makefile pour le générer.
- Le fichier dsmtca (UNIX et Linux uniquement) .

Pour les applications 64 bits, toutes les compilations doivent être effectuées à l'aide des options de compilation permettant une prise en charge en 64 bits. Par exemple, '-q64' doit être utilisé pour générer des applications API sous AIX et '-m64' doit être utilisé sous Linux. Pour plus d'informations, voir les modèles de fichiers makefile.

Important : Lorsque vous installez l'API, vérifiez que tous les fichiers sont au même niveau.

Pour plus d'informations sur l'installation de l'API, voir Installation des clients de sauvegarde-archivage IBM Spectrum Protect.

Les références à UNIX et Linux incluent les systèmes d'exploitation AIX, HP-UX, Linux, Mac OS X et Oracle Solaris.

Analyse des fichiers de configuration et d'options

Les fichiers de configuration et d'options définissent les conditions et les limites relatives à l'exécution de votre session.

Un administrateur, un utilisateur final et vous-même pouvez définir les valeurs d'option de manière à :
- Configurer la connexion à un serveur
- Contrôler les objets envoyés au serveur et à la classe de gestion à laquelle ils sont associés

Vous définissez les options dans un ou deux fichiers lors de l'installation de l'API sur votre poste de travail.

Sur les systèmes d'exploitation UNIX et Linux, les options résident dans deux fichiers d'options :
- dsm.opt - le fichier d'options client
- dsm.sys - le fichier d'options système d'un client

Sous d'autres systèmes d'exploitation, le fichier d'options client (dsm.opt) contient toutes les options.

Restriction : L'API ne prend pas en charge les options de client de sauvegarde-archivage suivantes :
- `autofsrename`
- `changingretries`
- `domaine`
- `eventlogging`
- `groups`
- `subdir`
- `users`
- `virtualmountpoint`

Vous pouvez également indiquer des options sur l'appel de fonction **`dsmInitEx`**. Utilisez le paramètre de la chaîne d'option ou le paramètre du fichier de configuration de l'API.

La même option peut provenir de plusieurs sources de configuration. Lorsque cela se produit, la source doté de la priorité supérieure est prioritaire. Le tableau 1 indique l'ordre de priorité.

Tableau 1. Sources de configuration par ordre de priorité décroissant

Priorité	UNIX et Linux	Windows	Description
1	fichier dsm.sys (options du système client)	non applicable	Ce fichier contient les options définies par un administrateur système uniquement pour UNIX et Linux. **Conseil :** Si votre fichier dsm.sys contient des sections serveur, assurez-vous que l'option `passwordaccess` spécifie la même valeur (`prompt` ou `generate`) dans chacune des sections.

Tableau 1. Sources de configuration par ordre de priorité décroissant (suite)

Priorité	UNIX et Linux	Windows	Description
2	Chaîne d'option (options client)	Chaîne d'option (toutes les options)	L'un de ces options prend effet lorsqu'elle est transmise comme paramètre vers un appel **dsmInitEx**. La liste peut contenir des options client telles que `compressalways`, `servername` (UNIX et Linux uniquement) ou `tcpserveraddr` (non UNIX). Avec la chaîne d'option de l'API, un client d'application peut apporter des modifications aux valeurs des options dans le fichier de configuration de l'API et dans le fichier d'options client. Par exemple, votre application peut demander à l'utilisateur final si une compression est nécessaire. En fonction des réponses des utilisateurs, vous pouvez créer une chaîne d'option API à l'aide de cette option et la passer dans l'appel vers **dsmInitEx**. Pour plus d'informations sur le format de la chaîne d'option de l'API, voir «**dsmInitEx**», à la page 124. Vous pouvez également définir ce paramètre sur `NULL`. Cela indique qu'il n'existe pas de chaîne d'option API pour cette session.
3	fichier de configuration de l'API (options client)	fichier de configuration de l'API (toutes les options)	Les valeurs que vous définissez dans le fichier de configuration de l'API remplacent les valeurs que vous configurez dans le fichier d'options client. Configurez les options dans le fichier de configuration de l'API en utilisant les valeurs appropriées dans la session IBM Spectrum Protect pour l'utilisateur. Les valeurs sont appliquées lorsque le nom du fichier de configuration de l'API est passé comme un paramètre dans l'appel **dsmInitEx**. Vous pouvez également définir ce paramètre à la valeur NULL. Cela indique qu'il n'existe pas de fichier de configuration de l'API pour cette session.
4	Fichier dsm.opt (options client)	Fichier dsm.opt (toutes les options)	Sous des systèmes d'exploitation UNIX et Linux, le fichier dsm.opt contient uniquement les options utilisateur. Sous d'autres systèmes d'exploitation, le fichier dsm.opt contient toutes les options. Pour substituer les options de ces fichiers, suivez les méthodes décrites dans le tableau ci-dessous.

Concepts associés:

Options de traitement

Configuration de l'environnement de l'API

L'API utilise des variables d'environnement uniques pour repérer des fichiers. Vous pouvez utiliser différents fichiers pour les applications de l'API à partir de ceux utilisés par le client de sauvegarde-archivage. Les applications peuvent utiliser l'appel de fonction **dsmSetup** pour remplacer les valeurs définies par les variables de l'environnement.

Conseil : Sous Windows, le répertoire d'installation par défaut est : `%SystemDrive%\Program Files\Common Files\Tivoli\TSM\api`

Le tableau 2 répertorie les variables d'environnement de l'API en fonction du système d'exploitation.

Tableau 2. Variables d'environnement de l'API

Variables	UNIX et Linux	Windows
DSMI_CONFIG	Nom qualifié complet du fichier d'options client (dsm.opt).	Nom qualifié complet du fichier d'options client (dsm.opt).
DSMI_DIR	Indique le chemin d'accès aux fichiers dsm.sys et dsmtca, au sous-répertoire en_US ainsi que d'autres supports de langue nationale (NLS). Le sous-répertoire en_US doit contenir le fichier dsmclientV3.cat.	Indique le chemin d'accès au fichier dscenu.txt et aux fichiers de message NLS.
DSMI_LOG	Indique le chemin d'accès au fichier dsierror.log.	Indique le chemin d'accès au fichier dsierror.log. Si l'option **errorlogname** du client est définie, l'emplacement spécifié par cette option se substitue au répertoire spécifié par DSMI_LOG.

Chapitre 2. Génération et exécution du modèle d'application de l'API

Le module de l'API inclut des modèles d'application illustrant les appels de fonction de l'API en contexte. Installez un modèle d'application et consultez le code source afin de comprendre comment vous pouvez utiliser les appels de fonction.

Sélectionnez un des modèles de modules d'applications de l'API suivants :
- Le module d'application interactif doté d'une unité d'exécution unique (dapi*)
- Le module d'application multitâche (callmt*)
- L'application test de groupage d'objets logiques (dsmgrp*)
- Le modèle d'application relatif à la politique de conservation d'événements (callevnt)
- Le modèle d'application relatif au maintien de la conservation(callhold)
- Le modèle d'application relatif à la protection de la conservation de données (callret)
- L'exemple de programme de mémoire tampon de données IBM Spectrum Protect (callbuff)

Pour vous aider lors de votre initiation, consultez la procédure de génération du modèle d'application dapismp par votre plateforme :
- Pour les applications UNIX ou Linux, voir «Fichiers source du modèle d'application sous UNIX ou Linux».
- Pour les applications Windows, voir «Modèle d'application Windows 64 bits», à la page 7.

Le modèle d'application dapismp crée ses propres flux de données lors de la sauvegarde ou de l'archivage des objets. Ce modèle ne peut lire ni écrire des objets sur le système de fichiers du disque local. Le nom de l'objet ne correspond à aucun fichier sur votre poste de travail. La «chaîne de valeur de départ» que vous entrez génère un modèle pouvant être vérifié lorsque l'objet est restauré ou récupéré. Une fois le modèle d'application compilé et sa commande de lancement **dapismp** exécutée, suivez les instructions qui s'affichent à l'écran.

Fichiers source du modèle d'application sous UNIX ou Linux

Pour générer et exécuter le modèle d'application sous UNIX ou Linux, vous devez vous assurer que vous disposez de certains fichiers. Une fois le modèle d'application généré, vous pouvez le compiler, puis l'exécuter.

Les fichiers répertoriés dans le tableau 3 incluent des fichiers source ainsi que d'autres fichiers dont vous avez besoin afin de générer le modèle d'application fourni avec le module API.

Tableau 3. Fichiers dont vous avez besoin pour générer le modèle d'application de l'API sous UNIX ou Linux

Noms de fichiers	Description
README_api_enu	Fichier README

Tableau 3. Fichiers dont vous avez besoin pour générer le modèle d'application de l'API sous UNIX ou Linux (suite)

Noms de fichiers		Description
dsmrc.h dsmapitd.h dsmapips.h dsmapifp.h release.h		Fichier d'en-tête des codes retour Fichier d'en-tête de définitions commun Fichier d'en-tête de définitions spécifiques au système d'exploitation Fichier d'entête d'un prototype de fonction Fichier d'en-tête des valeurs de l'édition
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	Modules du modèle d'application exécutés par la ligne de commande
makesmp[64].*xxx*		Fichier Makefile pour la génération de dapismp pour votre système d'exploitation. *xxx* indique le système d'exploitation.
callmt1.c callmt2.c		Exemples de fichiers à plusieurs unités d'exécution
callmtu1.c callmtu2.c		Exemples de fichier Unicode à unités d'exécutions multiples
libApiDS.xx libApiDS64.xx ou libApiTSM64.xx		Bibliothèque partagée (le suffixe varie en fonction de la plateforme)
dsmgrp.c callevnt.c callhold.c callret.c callbuff.c dpsthread.c		Regroupement d'exemples de fichiers Exemple de code source relatif à la politique de conservation d'événements Exemple de code source relatif à la suspension de la suppression Exemple de code source relatif à la protection de la conservation de données

Génération du modèle d'application UNIX ou Linux

Vous générez le modèle d'application d'API **dapismp** à l'aide d'un compilateur approprié à votre système d'exploitation.

Vous devez installer les compilateurs suivants pour générer le modèle d'application d'API UNIX ou Linux :

- IBM AIX - compilateur IBM Visual Age Version 6 ou ultérieure
- HP-IA64 - Compilateur aCC A.05.50 ou ultérieur
- Linux - Compilateur GCC version 3.3.3 ou ultérieure
- Mac OS X - Compilateur GCC version 4.0 ou ultérieure
- Oracle Solaris - Oracle Studio C++ compiler version 11 ou ultérieure

1. Pour générer les modèles d'API, exécutez la commande suivante :

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

   Où *xxx* indique le système d'exploitation.

2. Après avoir généré les modèles, définissez vos variables d'environnement, dont DSMI_DIRet vos fichiers d'options. Pour plus d'informations, voir «Analyse des fichiers de configuration et d'options», à la page 1.
3. Lors de votre première connexion, connectez-vous en tant que superutilisateur pour enregistrer votre mot de passe.

   Conseil : La définition de l'option compressalways sur no peut ne pas renvoyer d'objet non condensé. Ce comportement varie en fonction de la fonctionnalité de l'application.

   Pour définir la méthode de communication Shared Memory sous AIX, l'utilisateur du client de l'API de IBM Spectrum Protect doit respecter l'une des conditions suivantes :

   - Il doit être connecté en tant que superutilisateur.
   - Il doit avoir le même ID utilisateur que le processus exécutant le serveur IBM Spectrum Protect.

   Cette limitation ne s'applique pas si l'option passwordaccess est définie sur generate dans le fichier d'options système d'un client dsm.sys et la zone de contrôle des tâches est utilisée ou si vous modifiez vos droits d'accès aux fichiers du programme d'application à l'aide des commandes suivantes :

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

   Pour plus d'informations, consultez la documentation relative au programme d'application.
4. Exécutez la commande **dapismp** pour lancer cette application.
5. Effectuez votre sélection à partir de la liste d'options qui s'affiche. Vérifiez que vous exécutez l'action de connexion avant d'exécuter d'autres actions.

   Exigence : Lors de la saisie des noms d'espace fichier, de niveau supérieur et inférieur, insérez toujours le préfixe de délimiteur de chemin (/), par exemple : /myfilespace. Vous devez utiliser ce préfixe même lorsque vous indiquez un astérisque (*).

Concepts associés:

Variables d'environnement (systèmes UNIX et Linux)

Modèle d'application Windows 64 bits

Pour générer et exécuter le modèle d'application pour les systèmes Microsoft Windows 64 bits, vous devez installer l'API IBM Spectrum Protect et vérifier que vous disposez de certains fichiers source.

Restrictions :

- Pour des résultats optimaux, utilisez le chargement dynamique. Pour obtenir un exemple, consultez le fichier dynaload.c et l'implémentation dans code exemple.
- Les fichiers du modèle d'application sont situés dans les répertoires suivants :

  api64\obj
  Contient des exemples de fichiers objet programme de l'API.

  api64\samprun
  Contient l'exemple de programme **dapismp**. L'exemple de programme contient le répertoire d'exécution.
- La DLL tsmapi64.dll est une DLL de 64 bits.

- Utilisez le compilateur Microsoft C/C++ version 15 et le fichier Makefile `makesmp64.mak` pour compiler le modèle d'application de l'API **dapismp**. Il est peut être nécessaire d'ajuster les fichiers Makefile à votre environnement, plus précisément la bibliothèque ou les répertoires d'inclusion.
- Suite à la compilation de l'application, exécutez le modèle d'application en exécutant la commande **dapismp** à partir du répertoire `api64\samprun`.
- Effectuez votre sélection à partir de la liste d'options qui s'affiche. Vérifiez que vous exécutez l'action de connexion avant d'exécuter d'autres actions.
- Attribuez toujours un préfixe correspondant au délimiteur (\) de chemin d'accès approprié aux noms de haut et de bas niveau et à l'espace fichier lorsque vous entrez le nom, par exemple : `\myfilespace`. Vous devez utiliser ce préfixe même lorsque vous indiquez un astérisque (*).

Pour les systèmes d'exploitation Windows, les fichiers source dont vous avez besoin pour construire le modèle d'application sont répertoriés dans le tableau 4. Le modèle d'application est compris dans le module API. A titre d'information, un exécutable précompilé (dapismp.exe) est également inclus.

Tableau 4. Fichiers dont vous avez besoin pour générer le modèle d'application API Windows 64 bits

Noms de fichiers	Description
`api.txt`	Fichier README
`tsmapi64.dll`	Bibliothèques de liaison dynamique de l'API

`dsmrc.h`
`dsmapitd.h`
`dsmapips.h`
`dsmapifp.h`
`dsmapidl.h`
`release.h`
	Fichier d'en-tête des codes retour
	Fichier d'en-tête de définitions commun
	Fichier d'en-tête de définitions spécifiques à l'OS
	Fichier d'en-tête d'un prototype de fonction
	Fichier d'en-tête d'un prototype de fonction chargé de façon dynamique
	Fichier d'en-tête des valeurs de l'édition

`dapidata.h`
`dapint64.h`
`dapitype.h`
`dapiutil.h`
	Fichiers d'en-tête des codes source

`tsmapi64.lib`	Bibliothèque implicite

`dapibkup.c`
`dapiinit.c`
`dapint64.c`
`dapipref.c`
`dapiproc.c`
`dapiproc.h`
`dapipw.c`
`dapiqry.c`
`dapirc.c`
`dapismp64.c`
`dapiutil.c`
`dynaload.c`
	Fichiers d'en-tête des codes source pour dapismp.exe

`makesmpx64.mak`
(Windows x64)
`makesmp64.mak`
(Windows IA64)
	Fichiers Makefile pour la génération de modèles d'application

Tableau 4. Fichiers dont vous avez besoin pour générer le modèle d'application API Windows 64 bits (suite)

Noms de fichiers	Description

`callmt1.c`
`callmt2.c`
`callmtu164.c`
`callmtu264.c`
	Exemples de fichier à unités d'exécutions multiples

`dpsthread.c`	Exemple de fichier de code source

`callevnt.c`
`callhold.c`
`callret.c`
`callbuff.c`
	Code source relatif à la stratégie de conservation en fonction d'un événement
	Exemple de code source relatif au maintien de la conservation
	Exemple de code source relatif à la protection des données pendant la période de conservation
	Exemple de code source relatif à la mémoire tampon partagée (pas de copie)

Chapitre 3. Remarques concernant la conception d'une application

Lors de la conception d'une application, vous devez posséder une connaissance approfondie des différents aspects de cette API.

Pour obtenir une connaissance approfondie de l'API, consultez les rubriques suivantes :

Lors de la conception de votre application, prenez note des éléments du tableau 5. L'initialisation des structures à l'aide des zones **`memset`** peut changer dans les éditions ultérieures. La valeur `stVersion` s'incrémente avec extension du produit.

Tableau 5. Remarques concernant l'API lors de la conception d'une application

Elément de conception	Considérations
Définition de l'emplacement	L'application doit définir les paramètres régionaux avant d'appeler l'API. Pour définir la valeur par défaut des paramètres régionaux, ajoutez le code suivant à l'application: `setlocale(LC_ALL,"");` Pour définir l'emplacement sur une autre valeur, utilisez le même appel avec l'emplacement approprié dans le deuxième paramètre. Vérifiez les détails dans la documentation de chaque système d'exploitation que vous utilisez.

Tableau 5. Remarques concernant l'API lors de la conception d'une application (suite)

Elément de conception	Considérations

Contrôle de session	Appliquez les instructions suivantes au contrôle de session :
	• Affectez un nom de poste unique à chaque client de sauvegarde-archivage IBM TSM et produit client de l'API de TSM que vous utilisez. Voici quelques exemples de ces clients :
	– IBM Spectrum Protect for Mail
	– ou IBM Spectrum Protect HSM for Windows
	• Utilisez le même nom de propriétaire entre une sauvegarde et une restauration.
	• Utilisez l'option `passwordaccess` pour gérer l'accès au fichier de mots de passe sécurisé. Cette option affecte l'utilisation du processus enfant TCA sur UNIX et Linux uniquement, pour la gestion du mot de passe, le nom de poste et le nom du propriétaire de la session.
	• Vérifiez que les sessions de transfert de données s'arrêtent dès que la tâche se termine pour libérer les unités du serveur afin qu'elles puissent être utilisées par d'autres sessions.
	• Pour autoriser un transfert de données hors réseau local, utilisez l'appel de fonction **`dsmSetup`** en définissant l'indicateur `multithread` sur on.
	• Sous AIX, lorsque vous utilisez des applications multitâches ou hors réseau local, en particulier sur des machines multiprocesseurs, définissez la variable d'environnement `AIXTHREAD_SCOPE` sur `S` dans l'environnement avant de lancer l'application ; cela garantit de meilleures performances et une planification des tâches plus fiable. Par exemple :
	`EXPORT AIXTHREAD_SCOPE=S`
	En définissant `AIXTHREAD_SCOPE` sur `S`, user les unités d'exécution utilisateur créées avec les attributs par défaut sont placées dans l'espace de contention au niveau du système. Dans ce cas, l'unité d'exécution utilisateur est liée à l'unité d'exécution du noyau et elle est planifiée par le noyau. L'unité d'exécution du noyau sous-jacente n'est pas partagée avec aucune autre unité d'exécution utilisateur. Pour plus d'informations concernant cette variable d'environnement, consultez la rubrique suivante :
	«Utilisation du traitement multitâche», à la page 16
	• Vérifiez que seule une unité d'exécution d'une session appelle des fonctions de l'API à tout moment. Les applications utilisant plusieurs unités d'exécution avec le même descripteur de session doivent synchroniser leurs appels à l'API. Par exemple, utilisez une exclusion mutuelle **`mutex`** pour synchroniser les appels à l'API :
	getTSMMutex()
	issue TSM API call
	releaseTSMMutex()
	Utilisez cette approche uniquement lorsque les unités d'exécution partagent le même descripteur. Dans le cas contraire, vous pouvez faire des appels de fonction en parallèle.
	• Implémentez un modèle fournisseur/consommateur pour les transferts de données. Les appels de l'API se font de manière synchrone et les appels de **`dsmGetData function`** et **`dsmSendData function`** se bloquent jusqu'à ce que ces fonctions aient terminé. Avec ce modèle, l'application peut lire le tampon suivant en attendant le réseau. La dissociation entre la lecture/écriture de données et le réseau permet aussi d'améliorer les performances en cas de goulot d'étranglement du réseau ou de lenteurs. En général :
	`Data thread <---> shared queue of buffers <---> communication thread (issue calls to the IBM Spectrum Protect API)`
	• Utilisez la même session pour plusieurs opérations afin d'éviter une surcharge. Pour les applications traitant de nombreux petits objets, implémentez le regroupement de sessions de manière à ce que la même session puisse être utilisée lors de plusieurs petites opérations. Une surcharge est associée à l'ouverture et la fermeture d'une session sur le serveur IBM Spectrum Protect. L'appel à `dsmInit/dsmInitEX` est effectué en série. Ainsi, même dans une application multitâche, une seule unité d'exécution peut se connecter à tout moment. De plus, lors de la connexion, l'API envoie un nombre de requêtes uniques au serveur de manière ce dernier puisse effectuer toutes les opérations. Ces requêtes incluent une règle, une option, des espaces fichier et une configuration locale.

Tableau 5. Remarques concernant l'API lors de la conception d'une application (suite)

Elément de conception	Considérations
Suite d'opérations	Le serveur IBM Spectrum Protect verrouille les entrées de la base de données relatives à l'espace fichier lorsque certaines opérations sont en cours. Les règles suivantes s'appliquent lors de la conception d'applications de l'API de IBM Spectrum Protect : • Les requêtes verrouillent l'espace fichier pendant l'intégralité de la transaction. • Le verrou peut être partagé entre différentes opérations de requête, par conséquent, plusieurs opérations sur le même espace fichier peuvent s'exécuter en même temps. • Les opérations suivantes sont utilisées pour modifier la base de données du serveur IBM Spectrum Protect (**DB Chg**) : send, get, rename, update et delete. • L'exécution d'une opération **DB Chg** requiert un verrouillage de l'espace fichier pendant la modification de la base de données. • Plusieurs opérations **DB Chg** sur le même espace fichier peuvent s'exécuter en même temps. Il se peut qu'il y ait un temps d'attente lorsque la séquence attend le verrou à la fin de la transaction. • Le verrou de requête ne peut être partagé avec les opérations **DB Chg**. Une opération **DB Chg** retarde le début d'une requête sur le même espace fichier ; par conséquent, concevez vos applications de manière à séparer et sérialiser les requêtes des opérations **DB Chg** sur le même espace fichier.
Attribution de noms d'objet	Lorsque vous renommez des objets, tenez compte des facteurs suivants : • Les noms spécifiques des objets sont les noms d'objet de haut et de bas niveau. Si un identificateur unique, tel qu'un horodatage, fait partie du nom, alors des objets de sauvegarde sont toujours actifs. Les objets n'expirent que s'ils sont signalés comme inactifs, à l'aide d'un appel de fonction **dsmDeleteObj**. • La méthode de restauration pour les objets détermine le format du nom pour faciliter les requêtes. La compression n'est pas possible si vous décidez d'utiliser une restauration partielle d'objet (POR). Pour supprimer la compression, utilisez la fonction **dsmSendObj objAttr objCompressed=bTrue**.
Regroupement d'objets	Regroupez les objets de manière logique à l'aide d'espaces fichier. Un espace fichier est un conteneur du serveur qui fournit une catégorie de regroupement pour les objets. L'API interroge tous les espaces fichier lors de la connexion initiale et lors des requêtes, ainsi, le nombre d'espaces fichier doit être restreint. Il est raisonnable de supposer qu'une application définit de 20 à 100 espaces fichier par poste. L'API peut traiter plus d'espaces fichier, cependant chaque espace fichier entraîne une surcharge pour la session. Pour créer une séparation plus granulaire, utilisez l'objet `directory` dans l'application.
Manipulation des objets	N'enregistrez aucune valeur `objectID` pour des restaurations futures. Leur validité au cours de la durée de vie de l'objet n'est pas garantie. Faites particulièrement attention à l'ordre de restauration pendant une opération de ce type. A l'issue de la requête, faites un tri par rapport à cette valeur avant de procéder à la restauration. Si vous utilisez plusieurs types de media à accès séquentiel, alors vous devez y accéder dans des sessions différentes. Pour plus d'informations, consultez la rubrique suivante : «Sélection et tri d'objets par ordre de restauration», à la page 71
Classe de gestion	Prenez en considération le niveau de contrôle que l'application doit appliquer à la classe de gestion associée à ses objets. Vous pouvez utiliser des instructions include, ou alors préciser un nom lors de l'appel à la fonction **dsmSendObj**.
Taille des objets	IBM Spectrum Protect requiert une estimation de la taille de chaque objet. Prenez en compte la façon dont l'application estime la taille d'un objet. Une surestimation de la taille de l'objet est préférable à une sous-estimation.

Calcul des limites de taille

Certaines structures de données ou certaines zones dans l'API sont limitées en taille. Ces structures sont souvent des noms ou des zones de texte dont la taille ne peut dépasser une longueur prédéfinie.

Les zones suivante correspondent à de exemples de structures de données comportant des limites de taille :
- Type d'application
- Description d'archivage
- Destination du groupe de copie
- Nom du groupe de copie
- Informations sur l'espace fichier
- Nom de la classe de gestion
- Nom du propriétaire de l'objet
- Password

Ces limites sont définies en tant que constantes dans le fichier d'en-tête `dsmapitd.h`. Toute allocation de mémoire est basée sur ces constantes et non sur les valeurs numériques entrées. Pour plus d'informations, voir Annexe B, «Fichiers source des définitions de type d'API», à la page 165.

Maintien du contrôle des versions de l'API

Toutes les API proposent un contrôle des versions, qui peut prendre différentes formes. La version de l'API que vous utilisez dans votre application doit être compatible avec la bibliothèque d'API installée sur la station de travail de l'utilisateur final.

`dsmQueryApiVersionEx` devrait être le premier appel à l'API. Cet appel effectue les tâches suivantes :
- Confirme que la bibliothèque de l'API est installée et disponible sur le système de l'utilisateur final
- Renvoie le niveau de version de la bibliothèque de l'API à laquelle l'application a accès

L'interface est conçue de manière à assurer une compatibilité ascendante. Les applications développées avec des versions ou éditions plus anciennes de la bibliothèque fonctionnent correctement lorsque vous exécutez une version plus ancienne.

Il est toutefois important de déterminer le niveau d'édition de la bibliothèque API, car certaines versions peuvent avoir des besoins en mémoire et des définitions de structure de données différents. Il est peu probable que vous ayez besoin de compatibilité descendante. Pour plus d'informations sur votre plateforme, voir tableau 6.

Tableau 6. Informations sur la compatibilité des plateformes

Plateforme	Description
Windows	Les fichiers de message doivent être au même niveau que la bibliothèque (DLL). Le module TCA (Trusted Communication Agent module (dsmtca)) n'est pas utilisé.
UNIX ou Linux	La bibliothèque API, le module Trusted Communication Agent (dsmtca) et les fichiers de message doivent être au même niveau.

L'appel à **dsmQueryApiVersionEx** retourne la version de la bibliothèque API qui est installée sur la station de travail de l'utilisateur final. Vous pouvez, alors, comparer la valeur retournée avec la version de l'API qu'utilise l'application client.

Le numéro de version de l'API du client d'application se trouve dans le code objet compilé sous la forme d'un ensemble de quatre constantes définies dans le fichier dsmapitd.h:

```
DSM_API_VERSION
DSM_API_RELEASE
DSM_API_LEVEL
DSM_API_SUB_LEVEL
```

Voir Annexe B, «Fichiers source des définitions de type d'API», à la page 165.

La version de l'API de l'application client devrait être inférieure ou égale à celle de la bibliothèque API installée sur le système de l'utilisateur. Cependant, soyez attentif aux autres conditions. A tout moment vous pouvez utiliser la fonction **dsmQueryApiVersionEx**, que la session API soit démarrée ou non.

Les structures de données qu'utilise l'API contiennent également des informations de contrôle de version. Ces informations font partie de la première zone des structures de données. Au fur et à mesure d'améliorations apportées aux structures de données, le numéro de version est augmenté. Lors de l'initialisation de la zone de version, utilisez la valeur de version de la structure définie dans le fichier dsmapitd.h.

La figure 1, à la page 16 illustre la définition de type de la structure **dsmApiVersionEx** du fichier d'en-tête dsmapitd.h. L'exemple définit ensuite une variable globale appelée ***apiLibVer***. Il montre également comment utiliser cette variable dans un appel à **dsmQueryApiVersionEx** afin d'obtenir la version de la bibliothèque API de l'utilisateur. Et enfin, la valeur obtenue est comparée à la version de l'API de l'application client.

```
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);
```

Figure 1. Exemple d'obtention du niveau de version de l'API

Utilisation du traitement multitâche

L'API multiprocessus permet aux applications de créer plusieurs sessions avec le serveur IBM Spectrum Protect au sein du même processus. L'API est réentrante. Tous les appels peuvent s'exécuter en parallèle à l'intérieur de plusieurs unités d'exécutions différentes.

Conseil : Lorsque vous exécutez des applications qui supposent l'utilisation de l'API avec un traitement multitâche, faites un appel à **dsmQueryAPIVersionEx**.

Pour utiliser l'API en mode multiprocessus, définissez la valeur de *mtflag* sur DSM_MULTITHREAD lors de l'appel de **dsmSetUp**. L'appel à la fonction **dsmSetUp** doit être le premier appel après celui à **dsmQueryAPIVersionEx**. Cet appel doit être terminé avant qu'une unité d'exécution appelle la fonction **dsmInitEx**. A la fin du traitement de toutes les unités d'exécution, faites un appel à la fonction **dsmCleanUp**. Le processus principal ne doit pas s'arrêter avant que toutes les unités d'exécution n'aient terminé leur traitement. Voir callmt1.c dans le modèle d'application.

Restriction : le mode par défaut de l'API est le traitement monotâche. Si une application n'appelle pas la fonction **dsmSetUp** avec la valeur de *mtflag* à DSM_MULTITHREAD, l'API ne permet qu'une seule session par processus.

Une fois **dsmSetUp** terminé avec succès, l'application peut démarrer plusieurs unités d'exécution et faire plusieurs appels à **dsmInitEx**. Chaque appel à **dsmInitEx** retourne un descripteur pour la session. Tous les appels suivants sur cette unité d'exécution pour cette session doivent utiliser ce descripteur. Certaines valeurs sont des variables d'environnement accessibles à tout le processus (des valeurs définies au cours d'un appel à **dsmSetUp**). Chaque appel à **dsmInitEx** réanalyse les options.

Chaque unité d'exécution peut avoir des options différentes soit en précisant un fichier de remplacement, ou à l'aide d'une chaîne d'options lors de l'appel à **dsmInitEx**. Cela permet à des unités d'exécution différentes de se connecter à différents serveurs ou d'utiliser des noms de poste différent.

Recommandation : Sur les systèmes HP, définissez la taille de la pile sur 64 ko au minimum. La valeur par défaut (32 ko) risque de ne pas suffire.

Pour permettre aux utilisateurs de l'application de disposer d'une session hors réseau local, utilisez **dsmSetUp** *mtFlag DSM_MULTITHREAD* dans votre application. Ceci est requis même pour une application à traitement monotâche. Cet indicateur active le traitement multitâche nécessaire à l'interface hors réseau local de IBM Spectrum Protect.

Signaux et gestionnaires de signaux

Une application doit gérer les signaux provenant de l'utilisateur ou du système d'exploitation. Si l'utilisateur appuie sur **CTRL+C**, l'application doit intercepter ce signal et envoyer des appels à **dsmTerminate** pour chaque unité d'exécution active. Elle appelle ensuite **dsmCleanUp** pour quitter. Si des sessions ne sont pas correctement fermées, cela peut entraîner des résultats inattendus sur le serveur.

Une application doit établir des gestionnaires de signaux, tels que SIGPIPE et SIGUSR1, pour prendre en compte les signaux qui peuvent provoquer l'arrêt d'une application. L'application reçoit alors, le code retour de l'API. Par exemple, pour ignorer SIGPIPE ajoutez l'instruction suivante dans votre application : `signal(SIGPIPE, SIG_IGN)`. Une fois cette information ajoutée, l'application reçoit le code retour correspondant à un canal de communication rompu, au lieu de s'arrêter.

Vous pouvez utiliser le processus enfant, TCA (Trusted Communication Agent) si l'option `passwordaccess` est défini sur `generate`. Quand vous utilisez le processus TCA, IBM Spectrum Protect se sert du signal SIGCLD. Si votre application se sert du signal SIGCLD, sachez que des interférences potentielles peuvent provenir des processus IBM Spectrum Protect et de la façon dont SIGCLD est utilisé. Pour plus d'informations sur l'utilisation du TCA, voir «Sécurité de session», à la page 19.

Démarrage ou fermeture d'une session

IBM Spectrum Protect est un produit basé sur les sessions, et toutes les activités doivent être effectuées au sein d'une session IBM Spectrum Protect. Pour ouvrir une session, l'application initie l'appel **dsmInitEx**. Ceci doit se faire avant tout autre appel à l'API, hormis **dsmQueryApiVersionEx**, **dsmQueryCliOptions** ou **dsmSetUp**.

L'appel à la fonction **dsmQueryCliOptions** peut uniquement être effectué avant l'appel à **dsmInitEx**. La fonction renvoie les valeurs d'options importantes, telles que les fichiers d'options ainsi que les paramètres de compression et de communication. L'appel à **dsmInitEx** démarre une session avec le serveur comme indiqué dans les paramètres figurant dans l'appel ou définis dans les fichiers d'options.

Le nom du poste client, le nom du propriétaire et les paramètres relatifs au mot de passe sont transmis via l'appel à **dsmInitEx**. Majuscules et minuscules sont équivalentes pour le nom du propriétaire, mais tel n'est pas le cas pour le nom de poste et le mot de passe. Les postes client de l'application doivent être enregistrés auprès du serveur avant le lancement d'une session.

Chaque fois qu'une application client de l'API démarre une session avec le serveur, le type de l'application est enregistré auprès du serveur. Indiquez toujours une abréviation du système d'exploitation pour la valeur du type d'application car celle-ci est entrée dans la zone plateforme sur le serveur. La longueur maximale est DSM_MAX_PLATFORM_LENGTH.

L'appel à la fonction **dsmInitEx** entraîne le lancement de la session IBM Spectrum Protect à l'aide du fichier de configuration de l'API et du fichier d'options de l'application client. Ces deux fichiers peuvent être utilisés par l'application client pour définir certaines options de IBM Spectrum Protect. Ces valeurs remplacent les valeurs qui ont été définies dans les fichiers de configuration de l'utilisateur lors de l'installation. Les utilisateurs ne peuvent pas modifier les options définies par l'administrateur. Si l'application client ne possède pas un fichier de configuration ou un fichier d'options, vous pouvez attribuer la valeur NULL à ces paramètres. Pour plus d'informations concernant les fichiers de configuration, consultez la rubrique suivante :

«Analyse des fichiers de configuration et d'options», à la page 1

L'appel de fonction **dsmInitEx** établit la session IBM Spectrum Protect, à l'aide de paramètres permettant une vérification étendue.

Vérifiez l'appel à **dsmInitEx** ainsi que le code retour des informations relatives à **dsmInitExOut**. L'administrateur annule la dernière session si la valeur du code retour est okay (RC=ok) et que le code retour des informations (infoRC) est DSM_RC_REJECT_LASTSESS_CANCELED. Pour arrêter la session en cours, appelez la fonction **dsmTerminate**.

L'appel à la fonction **dsmQuerySessOptions** renvoie les mêmes zones que l'appel à **dsmQueryCliOptions**. L'appel peut être envoyé à l'intérieur d'une session uniquement. Les valeurs reflètent les options client qui sont valides au cours de cette session, options figurant dans les fichiers d'options et résultant de toute opération de substitution dans l'appel de fonction **dsmInitEx**.

Une fois la session lancée, l'application adresse un appel à **dsmQuerySessInfo** afin de déterminer les paramètres du serveur qui ont été définis pour cette session. Les éléments tels que le domaine de règles et les limites de transactions sont renvoyés à l'application à l'aide de cet appel.

Fermez les sessions à l'aide d'un appel à **dsmTerminate**. Toutes les connexions au serveur sont interrompues et toutes les ressources associées à cette session sont libérées.

Pour obtenir un exemple de démarrage et de fermeture d'une session, consultez la rubrique suivante :

figure 2, à la page 21

L'exemple définit un nombre de variables globales et locales qui sont utilisées dans les appels à **dsmInitEx** and **dsmTerminate**. L'appel **dsmInitEx** utilise un pointeur vers `dsmHandle` comme paramètre, tandis que l'appel **dsmTerminate** utilise `dsmHandle` comme paramètre. L'exemple dans la figure 3, à la page 21 affiche les informations relatives à la fonction **rcApiOut**. La fonction **rcApiOut** appelle la fonction **dsmRCMsg** de l'API, qui convertit un code retour en un message. L'appel de fonction **rcApiOut** imprime ensuite le message pour l'utilisateur. Une version de la fonction **rcApiOut** est incluse dans l'exemple d'application de l'API. La fonction

dsmApiVersion est une définition de type incluse dans le fichier d'en-tête dsmapitd.h.

Sécurité de session

Le système à base de session IBM Spectrum Protect est doté de composants de sécurité permettant aux applications de lancer des sessions de façon sécurisée. Ces mesures de sécurité empêchent tout accès non autorisé au serveur et permettent d'assurer l'intégrité du système.

Toutes les sessions lancées avec le serveur doivent effectuer une procédure de connexion. Cette procédure requiert un mot de passe. Si le mot de passe est associé au nom de poste du client, cela permet de disposer de l'autorisation appropriée lors de la connexion au serveur. L'application client fournit ce mot de passe à l'API afin de démarrer la session.

Il existe deux méthodes pour le traitement des mots de passe : *passwordaccess=prompt* ou *passwordaccess=generate*. Si vous utilisez l'option *passwordaccess=prompt*, vous devez inclure la valeur du mot de passe dans chaque appel à **dsmInitEx**. Vous pouvez également entrer le nom de poste ou le nom du propriétaire dans l'appel à **dsmInitEx**.

Les mots de passe sont dotés de délais d'expiration. Si un appel à **dsmInitEx** échoue en raison d'un code retour indiquant que le mot de passe est expiré (DSM_RC_REJECT_VERIFIER_EXPIRED), l'application client doit effectuer l'appel à **dsmChangePW** à l'aide du descripteur renvoyé par l'appel de fonction **dsmInitEx**. Cette opération met à jour le mot de passe avant que la connexion ne soit établie. L'exemple dans la figure 4, à la page 22 décrit la procédure utilisée pour modifier un mot de passe à l'aide de la fonction **dsmChangePW**. Le propriétaire de la connexion doit se servir d'un ID superutilisateur ou d'un ID utilisateur autorisé pour changer le mot de passe.

La seconde méthode, *passwordaccess=generate*, déchiffre et enregistre la valeur du mot de passe dans un fichier. Le nom de poste et le nom du propriétaire ne peuvent pas être inclus dans l'appel à **dsmInitEx**. Par conséquent, les valeurs par défaut du système sont utilisées. Ceci permet d'assurer la sécurité du fichier de mots de passe. Une fois le mot de passe expiré, le paramètre generate crée un autre mot de passe et met automatiquement à jour le fichier de mots de passe.

Conseils :

1. Si deux machines physiques sont dotées du même nom de poste IBM Spectrum Protect ou si plusieurs chemins d'accès ont été définis sur un seul poste à l'aide de diverses strophes de serveur, il se peut que la méthode *passwordaccess=generate* soit valide uniquement pour la première strophe utilisée après l'expiration du mot de passe. Au cours de la première communication client-serveur, l'utilisateur est invité à entrer le même mot de passe pour chaque strophe de serveur. De plus, une copie du mot de passe est stocké pour chaque strophe. Une fois le mot de passe expiré, un autre mot de passe est généré pour la strophe à l'origine de la connexion client-serveur. Toutes les tentatives suivantes de connexion via les autres strophes de serveur échouent car il n'existe aucun lien logique entre les différentes copies de l'ancien mot de passe et la copie mise à jour générée par la première strophe utilisée après l'expiration du mot de passe. Dans ce cas, vous devez mettre à jour les mots de passe avant l'expiration du mot de passe ou après l'expiration en tant que reprise après incident, comme suit :

a. Exécutez la commande **dsmadmc**, puis mettez à jour le mot de passe sur le serveur.
b. Exécutez la commande **dsmc -servername=stanza1**, puis utilisez le nouveau mot de passe pour générer une entrée appropriée.
c. Exécutez la commande **dsmc -servername=stanza2**, puis utilisez le nouveau mot de passe pour générer une entrée appropriée.

2. Pour UNIX ou Linux : seul le superutilisateur ou un utilisateur autorisé peut changer le mot de passe en utilisant *passwordaccess=prompt*. Seul le superutilisateur ou un utilisateur autorisé peut démarrer le fichier de mot de passe en utilisant *passwordaccess=generate*. Vous pouvez utiliser le processus enfant TCA pour le traitement des mots de passe. Une notification est envoyée à l'application à ce sujet car un processus enfant ainsi qu'un signal SIGCLD sont utilisés. TCA (Trusted Communication Agent) n'est pas utilisé dans les cas suivants :
   - L'option *passwordaccess* est définie sur *prompt*.
   - L'utilisateur est root.
   - L'appelant de la fonction doit être un utilisateur autorisé.

   Restriction : Les options `users` et `groups` ne sont pas reconnues.

Il est possible qu'une application limite l'accès utilisateur à l'aide d'autres moyens tels que le paramétrage des filtres d'accès.

Les applications disposant de plusieurs connexions IP vers un serveur IBM Spectrum Protect unique doivent utiliser le même nom de poste et le même mot de passe client IBM Spectrum Protect pour chaque session. Pour activer cette prise en charge, procédez comme suit :

1. Définissez une strophe de serveur IBM Spectrum Protect dans le fichier dsm.sys.
2. Pour les connexions n'utilisant pas d'adresse IP par défaut, indiquez les valeurs d'option pour l'adresse *TCPserver* et *TCPport* lors de l'appel à **dsmInitEx**.

Ces valeurs remplacent les informations de connexion IP. Cependant, le même poste de strophe (dsm.sys) ainsi que les mêmes informations de mot de passe sont utilisés par la session.

Remarque : Les postes d'une même grappe utilisent un mot de passe unique.

```
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);
}
```

Figure 2. Exemple de lancement et de fermeture de session

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

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

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

Figure 3. Détails de rcApiOut

```
printf("Enter your current password:");
gets(current_pw);
printf("Enter your new password:");
gets(new_pw1);
printf("Enter your new password again:");
gets(new_pw2);
/* If new password entries don't match, try again or exit. */
/* If they do match, call dsmChangePW.                      */

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

Figure 4. Exemple de modification de mot de passe

Définition de l'option `passwordaccess` sur `generate` sans TCA

Le processus enfant TCA (agent de communication autorisé) contrôle généralement l'accès au fichier de mots de passe sécurisé. Sur les systèmes UNIX et Linux, vous pouvez vous connecter en tant qu'utilisateur autorisé et définir l'option `passwordaccess` sur `generate` sans démarrer le processus TCA.

Procédez comme suit lorsque vous définissez l'option `passwordaccess` sur `generate` sans le processus TCA :

1. Effectuez une opération d'écriture dans l'application à l'aide d'un appel à **dsmSetUp**, entraînant ainsi le transfert de *argv[0]*. *argv[0]* comporte le nom de l'application appelant l'API. L'application est autorisée à exécuter un utilisateur autorisé ; toutefois, l'administrateur doit choisir le nom de connexion de l'utilisateur autorisé.
2. Définissez le bit ID groupe effectif (bit `S`) de l'exécutable de l'application sur `On`. Le propriétaire de ce fichier exécutable d'application peut devenir un utilisateur autorisé et peut créer un fichier de mots de passe, mettre à jour des mots de passe et exécuter des applications. Le propriétaire de l'exécutable de l'application et l'ID utilisateur qui exécute le programme doivent être identiques. Dans l'exemple suivant, *`Utilisateur`* est `utilisateur1`, le nom de l'exécutable de l'application est `applA` et `utilisateur1` est doté de droits d'accès en lecture/écriture sur le répertoire `/home/user1`. L'exécutable `applA` dispose des autorisations suivantes :

   ```
   -rwsr-xr-x user1    group1   applA
   ```
3. Demandez aux utilisateurs de l'application de se servir du nom d'utilisateur autorisé pour se connecter. IBM Spectrum Protect vérifie que l'ID de connexion correspond au propriétaire de l'exécutable de l'application avant d'autoriser l'accès au fichier de mots de passe sécurisé.
4. Définissez l'option `passworddir` option dans le fichier `dsm.sys` qui permet de pointer vers un répertoire dans lequel cet utilisateur dispose de droits en lecture/écriture. Par exemple, saisissez la ligne suivante dans la section serveur du fichier `dsm.sys` :

   ```
   passworddir /home/user1
   ```
5. Créez le fichier de mot de passe puis assurez-vous que l'utilisateur autorisé est le propriétaire du fichier.
6. Connectez-vous en tant que `user1` et exécutez `app1A`.
7. Appelez la fonction **dsmSetUp**, puis transmettez à *argv*.

Création d'un utilisateur administrateur possédant des droits de propriétés au client

Un administrateur possédant des droits propriétaires client peut définir les paramètres de l'appel de fonction **dsmInitEx** pour le lancement de sessions. La fonction d'«administrateur» peut être attribuée à cet utilisateur, lui conférant ainsi des droits de sauvegarde et de restauration pour les postes définis.

Pour recevoir des droits propriétaires client, procédez comme suit sur le serveur :

1. Définition de l'administrateur :

   ```
   REGister Admin nom_admin motdepasse
   ```

   Où :

   - *nom_admin* correspond au nom de l'administrateur.
   - *mot_de_passe* correspond au mot de passe de l'administrateur.
2. Définition du niveau d'autorité. Les utilisateurs disposant de droits système ou de droits de règles possèdent également des droits de propriétés au client.

   ```
   Grant Authority nom_admin classes droits node
   ```

   Où :

   - *nom_admin* correspond à l'administrateur.
   - *classes* correspond au poste.
   - *droits* dispose de l'un des niveaux d'autorité suivants :
     - `owner` : droits complets de sauvegarde et de restauration pour le poste
     - `node` : poste unique
     - `domain` : groupe de postes
3. Définition de l'accès à un seul poste.

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

   Où :

   - *node_name* correspond au poste utilisateur du client
   - *password* correspond au mot de passe du poste utilisateur du client
   - *user_id* correspond au nom de l'administrateur

Lorsque l'application utilise l'administrateur, la fonction **dsmInitEx** est appelée avec les paramètres `userName` et `userNamePswd`.

```
dsmInitEx
      clientNodeName = NULL
      clientOwnerName = NULL
      clientPassword = NULL
      userName = nom de 'l'administrateur'
      userNamePswd = mot de passe de 'l'administrateur'
```

Vous pouvez définir l'option Option `passwordaccess` pour `générer` ou `inviter`. L'utilisation de ces deux paramètres permet à la valeur `userNamePswd` de lancer la session. Lorsque la session est lancée, les procédures de sauvegarde ou de restauration peuvent être exécutées pour ce poste.

Noms et ID d'objet

Le serveur IBM Spectrum Protect est un serveur de stockage d'objets dont l'objectif principal est de stocker et de récupérer les objets nommés de manière efficace. L'ID objet est unique pour chaque objet et n'est pas doté d'un délai d'expiration, *sauf* lors de l'exportation ou de l'importation.

Pour satisfaire cette exigence, IBM Spectrum Protect dispose de deux zones de mémoire système, une base de données et un stockage de données.

- La base de données contient toutes les métadonnées, par exemple le nom ou les attributs, qui sont associées aux objets.
- L'emplacement de stockage de données contient les données objet. Le stockage de données est en fait une hiérarchie de stockage définie par l'administrateur système. Les données sont stockées et gérées de manière efficace sur les supports en ligne ou hors ligne en fonction du coût et des exigences relatives à l'accès.

Chaque objet stocké sur le serveur est doté d'un nom. Les principaux composants ci-après associés à ce nom sont contrôlés par le client :

- Nom d'espace fichier
- Nom de haut niveau
- Nom de bas niveau
- Type d'objet

Lors de l'attribution de noms aux objets relatifs à une application, il se peut qu'un nom externe doit être utilisé pour les noms d'objet complets de l'utilisateur final. Plus précisément, il se peut que l'utilisateur final doit indiquer l'objet dans une instruction d'inclusion ou d'exclusion lors de l'exécution de l'application. La syntaxe exacte du nom d'objet figurant dans ces instructions varie en fonction de la plateforme. Sous le système d'exploitation Windows, l'identificateur d'unité associé à l'espace fichier et non au nom d'espace fichier est utilisé dans l'instruction d'inclusion-exclusion.

Il se peut que la valeur de l'ID objet attribuée lors de la création d'un objet soit différente de la valeur attribuée lors d'une procédure de restauration. Les applications doivent être dotées du même nom d'objet. De plus, elles doivent obtenir l'ID objet en cours avant d'exécuter une procédure de restauration.

Nom d'espace fichier

Le nom d'espace fichier est l'un des composants de stockage principaux. Il peut s'agir du nom d'un système de fichiers, d'une unité de disque ou tout autre qualificatif de haut niveau regroupant les données correspondantes.

IBM Spectrum Protect utilise l'espace fichier pour identifier le système de fichiers ou le disque sur lequel les données sont stockées. Ainsi, des actions peuvent être exécutées sur toutes les entités figurant dans un espace fichier telles que l'interrogation de tous les objets dans un espace fichier défini. Etant donné que l'espace fichier est un composant très important de la convention d'attribution de noms de IBM Spectrum Protect, vous utilisez des appels spéciaux pour enregistrer, mettre à jour, interroger ou supprimer des espaces fichier.

Le serveur possède également des commandes administratives pour interroger les espaces fichier à propos des postes stockés dans IBM Spectrum Protect et les supprimer, si nécessaire. Toutes les données stockées par l'application client

doivent être dotées d'un nom d'espace fichier. Sélectionnez un nom approprié pour regrouper les données correspondantes dans le système.

Afin d'éviter toute interférence, une application client doit choisir des noms d'espaces fichier différents de ceux utilisés par un client de sauvegarde-archivage. Les noms d'espace fichier de l'application client doivent être accessibles de sorte que les utilisateurs finals puissent identifier les objets figurant dans les instructions d'inclusion-exclusion, le cas échéant.

Remarque : Sur les plateformes Windows, un identificateur d'unité est associé à un espace fichier. Lorsque vous enregistrez ou mettez à jour un espace fichier, vous devez indiquer l'identificateur d'unité. Comme la liste inclusive-exclusive se rapporte à l'identificateur d'unité, il est nécessaire de procéder à un suivi de chaque identificateur et de l'espace fichier correspondant. Dans l'exemple de programme dapismp, la valeur par défaut attribuée à l'identificateur d'unité est "G".

Pour plus d'informations sur les exemples de programme, voir Chapitre 2, «Génération et exécution du modèle d'application de l'API», à la page 5.

Noms de haut et de bas niveau

Le qualificateur de nom de haut niveau et le qualificateur de nom de bas niveau sont deux autres composants du nom de l'objet. Le qualificateur de nom de haut niveau correspond au répertoire dans lequel est stocké l'objet. Le qualificateur de nom de bas niveau correspond lui au nom réel de l'objet figurant dans ce répertoire.

Si le nom d'espace fichier, le nom de haut niveau et le nom de bas niveau sont concaténés, la syntaxe du nom constitué doit être correcte afin que celui-ci soit valide pour le système d'exploitation sous lequel est exécuté le client. Il n'est pas nécessaire que le nom existe en tant qu'objet sur le système ou qu'il soit identique aux données en cours sur le système de fichiers local. Cependant, le nom doit être conforme aux règles d'appellation standard afin d'être correctement traité par les appels à **dsmBindMC**. Consultez la section «Analyse des objets de sauvegarde et d'archivage», à la page 44 pour les considérations relatives à l'attribution de noms associées à la gestion de règles.

Type d'objet

Le type de l'objet permet d'identifier s'il s'agit d'un fichier ou d'un répertoire. Un fichier est un objet contenant des attributs et des données binaires, tandis qu'un répertoire est un objet contenant uniquement des attributs.

Le tableau 7 indique le code client d'application pour les noms d'objet par plateforme.

Tableau 7. Exemples de nom objet d'application par plateforme

Plateforme	Code client pour nom d'objet
UNIX ou Linux	/myfs/highlev/lowlev
Windows	"myvol\\highlev\\lowlev" **Remarque :** Sur une plateforme Windows, utilisez une double barre oblique inversée car la barre oblique est une caractère d'échappement. Les noms d'espace fichier sont précédés d'une barre oblique sur les plateformes UNIX ou Linux, mais ce n'est pas le cas sur les plateformes Windows.

Accès aux objets en tant que propriétaire de la session

Chaque objet est doté d'un nom de propriétaire. Les règles déterminant l'accès aux objets varient en fonction du nom de propriétaire utilisé lors du lancement d'une session. Utilisez la valeur relative au propriétaire de la session pour contrôler l'accès à l'objet.

Le propriétaire de la session est défini lors de l'appel à **dsmInitEx** dans le paramètre *clientOwnerNameP*. Si vous démarrez une session avec la valeur *NULL* attribuée au nom du propriétaire **dsmInitEx** et que vous utilisez l'option *passwordaccess=prompt*, le propriétaire de la session est géré via à une autorité de session (superutilisateur ou utilisateur autorisé). Cela s'applique également si vous vous connectez avec un ID super utilisateur ou utilisateur autorisé et que vous utilisez *passwordaccess= generate*. Lors d'une session démarrée à l'aide de cette méthode, vous pouvez exécuter une action sur les objets appartenant à ce poste, indépendamment du propriétaire de l'objet.

Si une session est lancée à l'aide d'un nom de propriétaire spécifique, elle permet d'exécuter uniquement des actions sur les objets portant ce nom. Les opérations de sauvegarde ou d'archivage exécutées sur le système doivent également être dotées d'un nom de propriétaire. Toutes les interrogations effectuées renvoient uniquement les valeurs associées au nom du propriétaire. La valeur relative au propriétaire de l'objet est définie lors d'un appel à **dsmSendObj** dans la zone **Owner** de la structure **ObjAttr**. La différenciation majuscules/minuscules s'applique au nom de propriétaire. Le tableau 8 récapitule les conditions d'accès d'un utilisateur à un objet.

Tableau 8. Récapitulatif de l'accès utilisateur aux objets

Propriétaire de la session	Propriétaire de l'objet	Accès utilisateur
NULL (root, propriétaire système)	« » (chaîne vide)	Oui
NULL	Nom spécifique	Oui
Nom spécifique	« » (chaîne vide)	Non
Nom spécifique	Même nom	Oui
Nom spécifique	Nom différent	Non

Accès aux objets sur plusieurs postes et propriétaires

Trois appels de fonction prennent en charge l'accès sur plusieurs postes et entre plusieurs propriétaires sur la même plateforme : **dsmSetAccess**, **dsmDeleteAccess**, et **dsmQueryAccess**. Ces fonctions, de même que les options de chaîne *-fromnode* et *-fromowner* qui sont transmises via **dsmInitEx**, autorisent une requête relative à plusieurs postes ainsi qu'un processus de restauration et de récupération via l'API.

Par exemple, l'utilisateur A sur le poste A utilise l'appel de fonction **dsmSetAccess** pour permettre à l'utilisateur B sur le poste B d'accéder aux sauvegardes figurant dans l'espace fichier `/db`. La règle d'accès est la suivante :

ID	Type	Poste	Utilisateur	Chemin d'accès
1	Sauvegarde	Poste B	Utilisateur B	`/db/*/*`

Lorsque l'Utilisateur B se connecte au Poste B, la chaîne d'options de la fonction **dsmInitEx** est la suivante :

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

Ces options sont définies pour cette session. Toutes les requêtes permettent d'accéder aux espaces fichier ainsi qu'aux fichiers du poste A. Les opérations de sauvegarde et d'archivage ne sont pas autorisées. Seuls les processus d'interrogation, de restauration et de récupération sont autorisés à partir des espaces fichier auxquels l'Utilisateur B peut accéder. Si l'application tente d'exécuter une opération à l'aide de **dsmBeginTxn** (par exemple, une opération de sauvegarde ou de mise à jour) au cours d'une connexion à un jeu d'options *-fromnode* ou *-fromowner*, puis **dsmBeginTxn** échoue avec un code retour DSM_RC_ABORT_NODE_NOT_AUTHORIZED. Pour plus d'informations, consultez les appels de fonction individuels et «**dsmInitEx**», à la page 124.

Conseil : Sous UNIX et Linux, vous pouvez indiquer *–fromowner=root* dans la chaîne d'options transmise sur l'appel de fonction **dsmInitEx**. Ceci permet aux utilisateurs ne disposant pas de droits de superutilisateur d'accéder aux fichiers appartenant au superutilisateur lorsque la permission set access a été accordée.

L'option *asnodename* sur la chaîne d'options **dsmInitEx** permet, à l'aide de la fonction appropriée, de sauvegarder, archiver, restaurer, récupérer, interroger, supprimer des données sous le nom du poste cible sur le serveur IBM Spectrum Protect. Pour plus d'informations sur l'activation de cette option, voir «Sauvegarde de postes multiples avec la prise en charge du proxy sur le poste client», à la page 85.

Gestion des espaces fichier

Etant donné que les espaces fichier sont importants pour le fonctionnement du système, un ensemble d'appels distinct est utilisé pour enregistrer, mettre à jour et supprimer les ID d'espace fichier. Avant de stocker les objets associés à un espace fichier sur le système, vous devez d'abord enregistrer l'espace fichier auprès de IBM Spectrum Protect.

Utilisez l'appel à **dsmRegisterFS** pour l'exécution de cette tâche. Pour plus d'informations sur les noms d'objet et les ID, voir «Noms et ID d'objet», à la page 24.

L'ID d'espace fichier est un qualificateur de haut niveau figurant dans une hiérarchie de noms divisée en trois parties. Le regroupement de données correspondantes dans un espace fichier facilite la gestion de ces donnée. Par exemple, l'application client ou l'administrateur du serveur IBM Spectrum Protect peut supprimer un espace fichier ainsi que tous les objets y figurant.

Les espaces fichier permettent également à l'application client de fournir des informations relatives à l'espace fichier au serveur pouvant être interrogé par l'administrateur. Ces informations sont renvoyées vers la requête dans la structure **qryRespFSData** et comprennent des données sur le système de fichiers suivant :

Type	Définition
fstype	Type de l'espace fichier. Cette zone est une chaîne de caractères définie par l'application client.
fsAttr[platform].fsInfo	Un champ d'informations sur le client utilisé pour les données spécifiques au client.
capacity	Quantité totale d'espace dans l'espace fichier.
occupancy	Quantité d'espace actuellement occupée dans l'espace fichier.

Type	Définition
backStartDate	Horodatage du début de la sauvegarde la plus récente (défini à l'aide d'un appel à **dsmUpdateFS**).
backCompleteDate	Horodatage de la fin de la sauvegarde la plus récente (défini à l'aide d'un appel à **dsmUpdateFS**).

L'utilisation des commandes capacity et occupancy varient en fonction de l'application client. Il se peut que certaines applications ne requièrent pas d'informations sur la taille de l'espace fichier. Dans ce cas la valeur par défaut de ces zones est `0`. Pour plus d'informations sur l'interrogation des espaces fichier, voir «Interrogation du système IBM Spectrum Protect», à la page 35.

Une fois l'espace fichier enregistré auprès du système, vous pouvez sauvegarder ou archiver des objets à n'importe quel moment. Pour mettre à jour les zones d'utilisation et de capacité de l'espace fichier après une opération de sauvegarde ou d'archivage, appelez **dsmUpdateFS**. Cet appel permet de garantir que les valeurs relatives à l'occupation et la capacité du système de fichiers soient celles en cours. Vous pouvez également mettre à jour les valeurs des zones **fsinfo**, **backupstart** et **backupcomplete**.

Si vous souhaitez surveiller les dates de sauvegarde les plus récentes, la fonction **dsmUpdateFS** doit être appelée le démarrage de la sauvegarde. Définissez l'action de mise à jour à DSM_FSUPD_BACKSTARTDATE. Cette opération contraint le serveur à ajuster la valeur de la zone **backStartDate** de l'espace fichier à l'heure en cours. Une fois la procédure de sauvegarde de l'espace fichier terminée, faites un appel à **dsmUpdateFS**, après avoir défini l'action de mise à jour à DSM_FSUPD_BACKCOMPLETEDATE. Cet appel crée un horodatage à la fin de la sauvegarde.

Si un espace fichier n'est plus requis, vous pouvez le supprimer à l'aide de la commande **dsmDeleteFS**. Sous un système d'exploitation UNIX ou Linux, seul un superutilisateur ou un utilisateur autorisé peut supprimer un espace fichier.

Les exemples dans la figure 5, à la page 29 décrivent l'utilisation des trois appels d'espaces fichier pour UNIX ou Linux. Pour un exemple d'utilisation des trois appels d'espaces fichier pour Windows, voir le code de l'exemple de programme installé sur votre système.

```
/*  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);
}
```

Figure 5. Exemple d'utilisation des espaces fichier (première partie)

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

dsmFSUpd     updFilespace;         /* for update FS */

updFilespace.stVersion = dsmFSUpdVersion;
updFilespace.fsType = 0;             /* no change */
updFilespace.occupancy.hi = 0;
updFilespace.occupancy.lo = 50;
updFilespace.capacity.hi = 0;
updFilespace.capacity.lo = 200;
strcpy(updFilespace.fsAttr.unixFSAttr.fsInfo,
       "My update for filespace") ;
updFilespace.fsAttr.unixFSAttr.fsInfoLength =
       strlen(updFilespace.fsAttr.unixFSAttr.fsInfo);

updAction = DSM_FSUPD_FSINFO |
            DSM_FSUPD_OCCUPANCY |
            DSM_FSUPD_CAPACITY;

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

Figure 6. Exemple d'utilisation des espaces fichier (deuxième partie)

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

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

Figure 7. Exemple d'utilisation des espaces fichier (troisième partie)

Association d'objets aux classes de gestion

Une des fonctions principales de IBM Spectrum Protect est l'utilisation de règles (classes de gestion) pour définir la méthode de stockage et de gestion des objets dans la mémoire IBM Spectrum Protect. Un objet est associé à une classe de gestion lorsque celui-ci est sauvegardé ou archivé.

Cette classe de gestion permet de déterminer les éléments suivants :
- Le nombre de versions de l'objet qui sont conservées en cas de sauvegarde
- Le délai de conservation des copies d'archivage
- L'emplacement d'insertion de l'objet dans la hiérarchie de stockage du serveur

Les classes de gestion sont composées de groupes de copies de sauvegarde et de groupes de copies d'archivage. Un groupe de copies est constitué d'un ensemble d'attributs qui définissent les règles de gestion relatives à un objet en cours de sauvegarde ou d'archivage. Les attributs du groupe de paramètres de sauvegarde sont valides au cours d'une opération de sauvegarde. Les attributs du groupe de copies d'archivage sont valides au cours d'une opération d'archivage.

La valeur attribuée au groupe de paramètres de sauvegarde ou d'archivage d'une classe de gestion spécifique peut être empty ou `NULL`. Si un objet est lié au groupe de paramètres de sauvegarde dont la valeur est `NULL`, cet objet ne peut pas être sauvegardé. Si un objet est lié à un groupe de paramètres d'archivage dont la valeur est `NULL`, cet objet ne peut pas être archivé.

L'utilisation d'une règle étant un composant très important de IBM Spectrum Protect, l'API requiert qu'une classe de gestion soit d'abord attribuée à tous les objets envoyés au serveur en utilisant l'appel **dsmBindMC**. Avec le logiciel IBM Spectrum Protect, vous pouvez utiliser une liste d'inclusion-exclusion pour affecter la liaison des classes de gestion. L'appel à **dsmBindMC** utilise la liste d'inclusion-exclusion en cours pour affecter les classes de gestion.

Les instructions d'inclusion peuvent associer une classe de gestion spécifique à un objet sauvegardé ou archivé. Les instructions d'exclusion autorisent l'archivage des objets mais peuvent empêcher la sauvegarde de ces objets.

L'API requiert qu'un appel à **dsmBindMC** soit fait avant la sauvegarde ou l'archivage d'un objet. L'appel de **dsmBindMC** renvoie une structure mcBindKey contenant des informations sur la classe de gestion et les groupes de copies qui sont associées à l'objet. Vérifiez la destination du groupe de copies avant de procéder à l'envoi. Si vous envoyez plusieurs objets au cours d'une seule transaction, la destination du groupe de copies de ces objets doit être la même. L'appel de fonction **dsmBindMC** renvoie les informations suivantes :

Tableau 9. Informations renvoyées suite à l'appel à dsmBindMC

Informations	Description
Classe de gestion	Nom de la classe de gestion associée à l'objet. L'application client peut faire un appel à **dsmBeginQuery** pour définir tous les attributs de cette classe de gestion.
Groupe de copies de sauvegarde	Indique s'il existe un groupe de copies de sauvegarde pour cette classe de gestion. Si une opération de sauvegarde est exécutée et que le groupe de copies de sauvegarde n'existe pas, cet objet ne peut pas être envoyé vers l'espace de stockage. Un code d'erreur est renvoyé si vous tentez d'envoyer l'objet à l'aide d'un appel à **dsmSendObj**.
Destination de la copie de sauvegarde	Cette zone identifie le pool de stockage vers lequel les données sont envoyées. Si vous exécutez une transaction de sauvegarde à objets multiples, toutes les destinations de copie figurant dans cette transaction doivent être identiques. Si un objet est doté d'une destination de copie différente de celle des autres objets de la transaction, arrêtez la transaction en cours, puis lancez une nouvelle transaction avant d'envoyer l'objet. Un code d'erreur est renvoyé si vous tentez d'envoyer les objets vers des destinations de copie différentes au cours de la même transaction.
Groupe de copies d'archivage	Indique s'il existe un groupe de copies d'archivage pour cette classe de gestion. Si une opération d'archivage est exécutée et que le groupe de copies d'archivage n'existe pas, cet objet ne peut pas être envoyé vers l'espace de stockage. Un code d'erreur est renvoyé si vous tentez d'envoyer l'objet en appelant **dsmSendObj**.
Destination de la copie d'archivage	Cette zone identifie le pool de stockage vers lequel les données sont envoyées. Si vous exécutez une transaction d'archivage à objets multiples, toutes les destinations de copie figurant dans cette transaction doivent être identiques. Si un objet est doté d'une destination de copie différente de celle des autres objets de la transaction, arrêtez la transaction en cours, puis lancez une nouvelle transaction avant d'envoyer l'objet. Un code d'erreur est renvoyé si vous tentez d'envoyer les objets vers des destinations de copie différentes au cours de la même transaction.

Vous pouvez réaffecter les copies de sauvegarde d'un objet à une classe de gestion différente si une opération de sauvegarde ultérieure avec le même nom d'objet est exécutée à l'aide d'une classe de gestion différente. Par exemple, si vous sauvegardez l'objetA et affectez cet objet à la classe de gestion1, puis sauvegardez ce même objet ultérieurement et l'affectez à la classe de gestion2, la sauvegarde la plus récente réaffecte toutes les copies inactives à la classe de gestion2. Toutes les copies seront alors contrôlées par les paramètres définis dans la classe de gestion2. Les données ne sont toutefois pas transférées si la destination est différente.

Vous pouvez également réattribuer les accès des copies de sauvegarde à une classe de gestion différente à l'aide de l'appel **dsmUpdateObj** ou **dsmUpdateObjEx** et de l'opération DSM_BACKUPD_MC.

Référence associée:

Option Deduplication

Interrogation des classes de gestion

Les applications peuvent interroger les classes de gestion afin d'identifier les classes de gestion pouvant être affectées à un poste spécifique et déterminer les attributs figurant dans la classe de gestion.

Vous pouvez affecter les objets à des classes de gestion uniquement à l'aide de l'appel à **dsmBindMC**. Vous pouvez également utiliser vos applications pour interroger les attributs de classes de gestion et les afficher pour les utilisateurs finals. Pour plus d'informations, voir «Interrogation du système IBM Spectrum Protect», à la page 35.

Dans l'exemple illustré dans la figure 8, une instruction de transfert est utilisée pour différencier les opérations de sauvegarde des opération d'archivage lors de l'appel à **dsmBindMC**. Les informations transmises via cet appel sont stockées dans la structure **MCBindKey**.

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

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

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

Figure 8. Exemple d'association d'une classe de gestion à un objet

Verrouillage et interruption des délais d'expiration/de suppression

Vous pouvez interrompre la suppression et l'expiration d'objets archivés spécifiques suite à une action mise en attente ou en cours d'exécution nécessitant la conservation de ces données. Si une action lancée nécessite l'accès à des données spécifiques, ces données doivent être disponibles jusqu'à la fin de l'exécution de l'action et qu'elles ne soient plus requises par le processus. Lorsque la mise en suspens n'est plus requise, le délai de suppression et d'expiration standard reprend, en respectant la durée de conservation d'origine.

Vérifiez la licence du serveur en effectuant un appel test **dsmRetentionEvent** :

1. Lancez une requête relative à l'objet que vous souhaitez mettre en suspens et recherchez l'ID de cet objet.
2. Faites des appels à **dsmBeginTxn**, **dsmRetentionEvent** avec l'option `Hold`, puis faites un appel à **dsmEndTxn**.
3. Si le serveur n'est pas sous licence, un vote d'abandon s'affiche avec le code de raison `DSM_RC_ABORT_LICENSE_VIOLATION`.

Restrictions :

1. Vous ne pouvez pas faire plusieurs appels à la fonction **dsmRetentionEvent** au cours d'une seule transaction.
2. Vous ne pouvez pas suspendre un objet qui l'est déjà.

1. Pour mettre des objets en attente, procédez comme suit.
   a. Interrogez le serveur pour rechercher tous les objets que vous souhaitez suspendre. Recherchez l'ID objet de chaque objet.

b. Faites un appel à **dsmBeginTxn**, puis à **dsmRetentionEvent** avec la liste d'objets, suivie d'un appel à **dsmEventType** : `eventHoldObj`. Si le nombre d'objets est supérieur à la valeur du paramètre `maxObjPerTxn`, utilisez plusieurs transactions.

c. Utilisez l'appel de fonction `qryRespArchiveData` sur l'appel de fonction **dsmGetNextQObj** pour confirmer que les objets sont mis en attente. Vérifiez la valeur de `objHeld` dans `qryRespArchiveData`.

2. Pour annuler la suspension des objets, procédez comme suit.

a. Interrogez le serveur pour rechercher tous les objets dont vous souhaitez désactiver la suspension. Recherchez l'ID objet de chaque objet.

b. Faites un appel à **dsmBeginTxn**, puis à **dsmRetentionEvent** avec la liste d'objets, suivie d'un appel à **dsmEventType** : `eventReleaseObj`. Si le nombre d'objets est supérieur à la valeur du paramètre `maxObjPerTxn`, utilisez plusieurs transactions.

c. Utilisez la réponse **qryRespArchiveData** sur l'appel de fonction `dsmGetNextQObj` pour confirmer si la suspension des objets a été désactivée. Vérifiez la valeur de `objHeld` dans **qryRespArchiveData**.

Protection de la conservation des données archivées

Les données qui sont sous le contrôle d'IBM Spectrum Protect ne peuvent être modifiées par des agents non autorisés, comme des individus ou un programme. Cette protection empêche la suppression de données, tels des objets archivés, par des agents avant l'expiration de la durée de conservation.

Ceci permet de garantir qu'aucun individu ou programme ne pourra supprimer des données contrôlées par IBM Spectrum Protect, de façon intentionnelle ou accidentelle. Un objet d'archivage envoyé à un serveur de protection des données archivées est protégé des suppressions accidentelles et que sa durée de conservation est appliquée. La protection des données archivées présente les restrictions suivantes :

- Seules les opérations d'archivage sont autorisées sur un serveur de protection de la conservation des archives.
- Les objets qui ne sont pas liés de manière explicite à une classe de gestion via une valeur dans l'appel de fonction **dsmBindMc** ou via les instructions d'inclusion-exclusion, sont liés au nom explicite de la classe de gestion par défaut. Par exemple, si la classe de gestion par défaut dans la politique du poste est MC1, l'objet est lié de manière explicite à MC1 plutôt qu'à DEFAULT. Suite à la réponse à une requête, l'objet s'affiche comme étant lié à MC1.
- Une fois la protection des données archivées activée, toute tentative de suppression d'un objet avant l'expiration de la durée de conservation renvoie le code `DSM_RC_ABORT_DELETE_NOT_ALLOWED` sur la transaction finale.

Reportez-vous à la documentation du serveur IBM Spectrum Protect pour des instructions sur la configuration de la protection des données archivées pour un objet d'archivage.

Pour définir la protection des données archivées, procédez comme suit.

1. Sur une nouvelle installation de serveur ne contenant aucunes données préalables, exécutez la commande **SET ARCHIVERETENTIONPROTECTION ON**.
2. Dans la chaîne d'option de l'API sur les appels de fonction **dsmInit** ou **dsmInitEx**, saisissez l'instruction suivantes :

```
-ENABLEARCHIVERETENTIONPROTECTION=yes
```

Vous pouvez également définir l'option enablearchiveretentionprotection dans votre fichier dsm.opt sur des systèmes autres que UNIX ou dans votre fichier dsm.sys sur des systèmes UNIX :

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

Pour plus d'informations sur cette option, voir «L'option enablearchiveretentionprotection».

3. Interrogez le serveur pour confirmer si la protection des données archivées est activée sur le serveur IBM Spectrum Protect. Vérifiez la valeur du champ archiveRetentionProtection dans la structure dsmQuerySessInfo.

L'option enablearchiveretentionprotection

L'option enablearchiveretentionprotection spécifie s'il est nécessaire d'activer la protection de la conservation des données pour les objets archivés sur le serveur IBM Spectrum Protect réservé à cet usage. Votre administrateur de serveur doit activer la protection de la conservation des données sur un nouveau serveur qui ne dispose pas déjà d'objets stockés (sauvegardés, archivés ou objets dont l'espace est géré). Un message d'erreur est généré si l'application API tente de stocker la version de sauvegarde d'un objet ou un objet dont l'espace est géré sur le serveur.

La note dans Chapitre 3, «Remarques concernant la conception d'une application», à la page 11 précise : «N'enregistrez aucune valeur objectID pour des restaurations futures. Leur validité au cours de la durée de vie de l'objet n'est pas garantie.» est flexible pour les applications d'Archive Manager étant donné que le serveur Archive Manager ne prend pas en charge les opérations d'exportation et d'importation. Les applications Archive Manager peuvent sauvegarder et utiliser l'ID objet pour améliorer les performances lors de la restauration des objets.

Si le serveur émet la commande **SET ARCHIVERETENTIONPROTECTION ON**, vous ne pouvez pas supprimer un objet archivé du serveur en utilisant la commande **delete filespace**, tant que les paramètres de politique du groupe des copies d'archivage ne sont pas satisfaits. Voir la documentation serveur appropriée pour plus d'informations sur la façon de configurer une classe de gestion.

Politique de conservation d'événements

Dans une politique de conservation en fonction d'un événement, la durée de conservation d'un objet d'archivage est initiée par un événement métier, tel que la fermeture d'un compte bancaire. Une politique de conservation d'événements aligne précisément la politique de conservation des données IBM Spectrum Protect avec les besoins métier. Lorsque l'événement se produit, l'application envoie un événement **eventRetentionActivate** pour cet objet au serveur pour initier la conservation.

Pour utiliser une politique de conservation en fonction d'un événement, procédez comme suit.

1. Sur le serveur, créez une classe de gestion avec une archive **copygroup** de type EVENT. Pour plus d'informations, consultez la documentation du serveur IBM Spectrum Protect.
2. Interrogez la classe de gestion pour confirmer qu'elle est basée sur un événement. Si la classe de gestion est basée sur un événement, le champ **retainInit** de la structure **archDetailCG** est ARCH_RETINIT_EVENT.

3. Affectez les objets à la classe de gestion basée sur un événement à l'aide des options include, **archmc** ou de manière explicite via l'attribut **mcNameP** de la structure **ObjAttr** dans l'appel de fonction **dsmSendObj**.
4. Lorsque vous souhaitez démarrer la conservation des objets, interrogez le serveur pour tous les objets affectés. Vérifiez s'ils sont dans un état EN ATTENTE et obtenez l'ID de l'objet. Dans un état en attente, le champ **retentionInitiated** de la structure **qryRespArchiveData** indique DSM_ARCH_RETINIT_PENDING.
5. Faites un appel à **dsmBeginTxn**, puis à **dsmRetentionEvent** avec la liste d'objets, suivie d'un appel à **dsmEventType** : eventRetentionActivate. Si le nombre d'objets est supérieur à la valeur du paramètre maxObjPerTxn, utilisez plusieurs transactions.

   Restriction : Vous ne pouvez émettre qu'un seul appel **dsmRetentionEvent** par transaction.
6. Interrogez les objets pour confirmer que la conservation est activée. Si la conservation est initiée, le champ **retentionInitiated** de la structure **qryRespArchiveData** possède une valeur I.

Interrogation du système IBM Spectrum Protect

L'API dispose de plusieurs requêtes pouvant être utilisées par l'application, telles les requêtes relatives aux classes de gestion.

Toutes les requêtes qui utilisent l'appel **dsmBeginQuery** suivent les étapes ci-après.

1. Faites l'appel à **dsmBeginQuery** avec le type de requête approprié :
   - Sauvegarde
   - Archivage
   - Objets sauvegardés actifs
   - Espace fichier
   - Classe de gestion

   L'appel **dsmBeginQuery** indique à l'API le format des données provenant du serveur. Les zones appropriées peuvent être placées dans les structures de données transmises via les appels à **dsmGetNextQObj**. L'appel Begin Query permet également à l'application client de définir l'étendue de la requête lorsque les paramètres de cet appel sont correctement définis.

   Restriction : Sur les systèmes UNIX ou Linux, seul le superutilisateur peut interroger les objets sauvegardés actifs. Ce type de requête est appelé "raccourci".
2. Faites un appel à **dsmGetNextQObj** pour obtenir tous les enregistrements à partir de la requête. Cet appel permet de d'accéder à une mémoire tampon suffisamment large pour contenir les données résultant de la requête. Chaque type de requête dispose d'une structure de données correspondante pour les données renvoyées. Par exemple, une requête de sauvegarde est associée à une structure **qryRespBackupData** complétée lorsqu'un appel est fait à **dsmGetNextQObj**.
3. L'appel à **dsmGetNextQObj** renvoie généralement un des codes suivants :
   - DSM_RC_MORE_DATA : émettez à nouveau l'appel à **dsmGetNextQObj**.
   - DSM_RC_FINISHED : il n'y a plus de données. Faites appel à la fonction **dsmEndQuery**.
4. Faites appel à la fonction **dsmEndQuery**. Lorsque toutes les données de requête ont été extraites ou si aucunes données supplémentaires ne sont nécessaires,

émettez l'appel **dsmEndQuery** pour mettre fin au traitement de requête. L'API supprime alors toutes les données restantes du flux de requêtes et libère les ressources utilisées pour la requête.

La figure 9 affiche le diagramme d'état relatif aux opérations de requête.

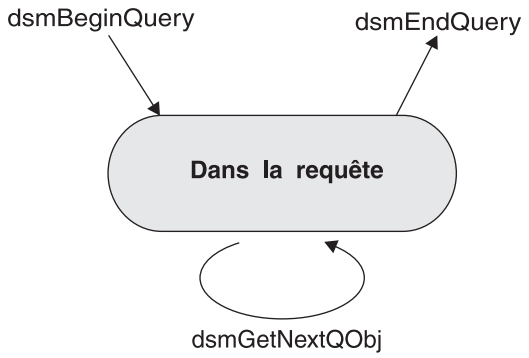


Figure 9. Diagramme d'état relatif aux requêtes générales

La figure 10 affiche l'organigramme relatif aux opérations de requête.

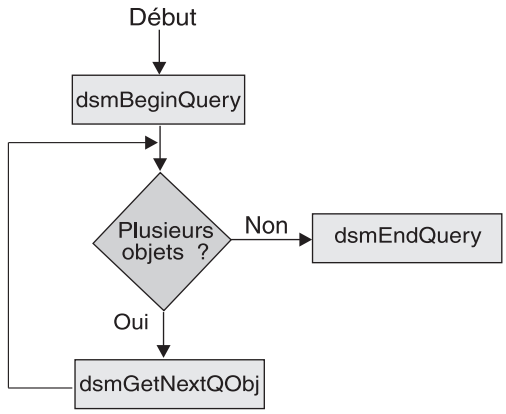


Figure 10. Organigramme relatif aux requêtes générales

Exemple d'interrogation du système

Dans cet exemple, une requête de classe de gestion imprime les valeurs figurant dans toutes les zones des groupes de copies de sauvegarde et d'archivage associés à une classe de gestion spécifique.

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

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

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

qRespMCData.stVersion = qryRespMCDetailDataVersion;
```

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

Figure 11. Exemple d'exécution d'une requête système

Efficacité du serveur

Utilisez ces directives lorsque vous récupérez ou envoyez des objets au serveur IBM Spectrum Protect.

- Lorsque vous récupérez des objets sur le serveur IBM Spectrum Protect, respectez les directives suivantes :

- Récupérez les données dans l'ordre de restauration fourni par le serveur IBM Spectrum Protect. L'ordre de restauration est particulièrement important pour les unités de bande, car la récupération de données dans un ordre inapproprié peut provoquer des rembobinages et des montages.
- Même lorsque les données sont stockées sur une unité de disque, vous pouvez gagner du temps en les récupérant dans le bon ordre.
- Effectuez autant d'opérations que possible au cours d'une session du serveur IBM Spectrum Protect.
- Evitez de lancer et d'interrompre de multiples sessions.

• Lorsque vous envoyez des objets au serveur IBM Spectrum Protect, respectez les directives suivantes :
  - Envoyez de multiples objets au cours d'une même transaction.
  - Evitez d'envoyer un seul objet par transaction, surtout lorsque les données sont envoyées directement à une unité de disque. Une partie de la transaction de l'unité de bande consiste à s'assurer que les données présentes dans les mémoires tampon RAM de la bande sont écrites sur des supports.

Concepts associés:

«Sélection et tri d'objets par ordre de restauration», à la page 71

Information associée:

«Démarrage ou fermeture d'une session», à la page 17

Envoi de données à un serveur

L'API permet aux applications client d'envoyer des données ou des objets nommés ainsi que leurs données correspondantes vers la mémoire du serveur IBM Spectrum Protect.

Conseil : Vous pouvez sauvegarder ou archiver les données. Effectuez toutes les opérations d'envoi d'une transaction.

Modèle de transaction

L'envoi des données à la mémoire IBM Spectrum Protect lors des opérations de sauvegarde ou d'archivage est effectué au cours d'une transaction. Un modèle de transaction fournit un degré d'intégrité des données élevé mais il impose certaines restrictions qu'un client d'application doit prendre en considération.

Lancez une transaction à l'aide d'un appel à **dsmBeginTxn** ou arrêtez une transaction à l'aide d'un appel à **dsmEndTxn**. Une seule transaction correspond à une action atomique. Les données envoyées dans le cadre d'une transaction sont validées dans le système à la fin de la transaction ou sont annulées si la transaction est interrompue.

Les transactions peuvent être constituées d'envois à objet unique ou d'envois à objets multiples. Pour améliorer vos performances système en diminuant le temps système, envoyez des objets de taille inférieure lors d'une transaction à plusieurs objets. Le client de l'application détermine si les transactions uniques ou les transactions multiples sont appropriées.

Envoyez tous les objets figurant dans une transaction à objets multiples vers la même destination de copie. Si vous devez envoyer un objet vers une destination différente de celle de l'objet précédent, arrêtez la transaction en cours, puis démarrez une nouvelle transaction. Au cours de la nouvelle transaction, vous pouvez envoyer l'objet vers la nouvelle destination de copie.

Remarque : La cohérence de la destination de copie des objets ne contenant pas de données binaires (*sizeEstimate=0*) n'est pas vérifiée.

IBM Spectrum Protect restreint le nombre d'objets pouvant être envoyés au cours d'une transaction à objets multiples. Pour obtenir cette limite, appelez **dsmQuerySessInfo** et examinez la zone **maxObjPerTxn**. Cette zone affiche la valeur de l'option *TXNGroupmax* définie sur votre serveur.

L'application client doit faire le suivi des objets envoyés au cours d'une transaction pour effectuer le traitement du processus de relance ou le traitement des erreurs si la transaction est interrompue. Une transaction peut être interrompue à n'importe quel moment par le serveur ou le client. L'application client doit être abilitée à gérer les interruptions des transactions qui n'ont pas été lancées par elle.

Agrégation de fichiers

Les serveurs IBM Spectrum Protect utilisent une fonction appelée file aggregation. Grâce à cette fonction, tous les objets envoyés au cours d'une seule transaction sont stockés dans le même emplacement, permettant ainsi d'améliorer la performance et d'économiser de l'espace. Vous pouvez poursuivre l'interrogation et la restauration des objets séparément.

Pour utiliser cette fonction, tous les objets figurant dans une transaction doivent être dotés du même nom d'espace fichier. Si ce nom est modifié au cours d'une transaction, le serveur arrête l'exécution de l'objet agrégé existant et lance un nouvel objet.

Transfert de données hors réseau local

L'API peut exécuter le transfert de données hors réseau local si l'option **dsmSetUp** relative au traitement multitâche est activée. L'API confirme l'existence d'une destination hors réseau local dans la structure de réponse **archDetailCG** de **Query Mgmt Class** ou dans la zone **bLanFreeDest** de **backupDetailCG** .

Vous pouvez exécuter des opérations hors réseau local sur les plateformes prises en charge par l'agent de stockage. La plateforme Macintosh est exclue.

Des informations hors réseau local sont disponibles dans les structures de résultat ci-dessous. La zone **totalLFBytesRecv** est comprise dans la structure de résultat (**dsmEndGetDataExOut_t**) de **dsmEndGetData**. Il s'agit du nombre total d'octets hors réseau local reçus. La zone **totalLFBytesSent** est incluse dans la structure de résultat (**dsmEndSendObjExOut_t**) de la fonction**dsmEndSendObjEx**. Il s'agit du nombre total d'octets hors réseau local qui ont été envoyés.

Information associée:

Transfert de données hors réseau local : présentation de l'agent de stockage

Opérations d'écriture simultanée

Vous pouvez configurer les pools de stockage du serveur IBM Spectrum Protect afin qu'ils exécutent des opérations d'écriture simultanément dans le pool de stockage principal et dans des pools de stockage de copie au cours d'une opération de sauvegarde ou d'archivage. Utilisez cette configuration pour créer plusieurs copies de l'objet.

En cas d'échec d'une opération d'écriture simultanée, le code retour de la fonction **dsmEndTxn** peut être DSM_RC_ABORT_STGPOOL_COPY_CONT_NO, indiquant que l'écriture dans l'un des pools de stockage de copie a échoué et que l'option de pool de stockage IBM Spectrum Protect COPYCONTINUE a été définie sur NO. L'application se termine et le problème doit être résolu par l'administrateur du serveur IBM Spectrum Protect.

Pour plus d'informations sur la configuration d'opérations d'écriture simultanée, voir la documentation sur le serveur IBM Spectrum Protect.

Amélioration des performances de l'API

Vous pouvez utiliser les options client tcpbuffsize et tcpnodelay, ainsi que le paramètre de l'API **DataBlk** pour améliorer les performances de l'API.

Le tableau 10 décrit les actions que vous pouvez effectuer pour améliorer les performances de l'API.

Tableau 10. Options de sauvegarde-archivage et paramètre de l'API améliorant les performances

Options client de sauvegarde-archivage	**Description**
tcpbuffsize	Indique la taille de la mémoire tampon TCP. La valeur par défaut est 31 Ko. Pour améliorer les performances, définissez cette valeur sur 32 Ko.
tcpnodelay	Indique si des mémoires tampon de taille inférieure doivent être envoyées au serveur plutôt que de les mettre en attente. Pour améliorer les performances, définissez cette option sur *yes* pour toutes les plateformes. Cette option est valide pour Windows et AIX uniquement.
Paramètre de l'API	**Description**
DataBlk	Ce paramètre est utilisé avec l'appel de fonction **dsmSendData** pour déterminer la taille de la mémoire tampon de l'application. Pour obtenir des résultats optimaux, définissez le paramètre comme un multiple de la valeur *tcpbuffsize* spécifiée à l'aide de tcpbuffsize moins 4 octets. Par exemple, définissez une valeur de 28 pour ce paramètre lorsque la valeur de tcpbuffsize est définie sur 32 Ko.

Chaque appel de **dsmSendData** est synchrone et ne renvoie pas de données tant que les données transférées vers l'API dans **dataBlkPtr** ne sont pas vidées sur le réseau. L'API ajoute un supplément de 4 octets à chaque mémoire tampon de transaction placée sur le réseau.

Par exemple, si la taille de mémoire tampon de transaction est de 32 Ko et la taille de la mémoire tampon **DataBlk** de l'application est de 31 Ko, alors chaque mémoire tampon **DataBlk** de l'application tient dans une mémoire tampon de communications et peut être vidée immédiatement. Cependant, si la mémoire tampon **DataBlk** de l'application est de 32 Ko exactement, et dans la mesure où

l'API ajoute 4 octets par mémoire tampon de transaction, deux vidages sont requis : un de 32 Ko et un de 4 octets. De plus, si vous définissez l'option `tcpnodelay` sur `no`, le vidage des 4 octets peut prendre jusqu'à 200 millisecondes.

Définissez l'API de sorte à envoyer les données de performances au moniteur de performances client

Le moniteur de performances client Tivoli Storage Manager est un composant du centre d'administration utilisé pour afficher les données de performances collectées par l'API. Le moniteur de performances client enregistre et affiche les données de performances pour les opérations de sauvegarde archivage et restauration du client.

Grâce à la surveillance des performances, vous pouvez afficher les données de performances collectées par l'API dans le centre d'administration Tivoli Storage Manager. Depuis la version 7.1, le composant de centre d'administration n'est plus inclus dans les distributions Tivoli Storage Manager ou IBM Spectrum Protect. Si vous disposez d'un centre d'administration qui a été installé avec une précédente version du serveur, vous pouvez continuer à l'utiliser pour afficher les données de performances. Si aucun centre d'administration n'est encore installé, vous pouvez télécharger la version précédemment publiée de ftp://public.dhe.ibm.com/storage/tivoli-storage-management/maintenance/admincenter/v6r3/. Pour plus d'informations sur l'utilisation du moniteur de performances, voir la documentation du serveur Tivoli Storage Manager version 6.3.

Configuration des options du moniteur de performances client

Vous permettez aux clients IBM Spectrum Protect d'utiliser le moniteur de performances en spécifiant les paramètres dans le fichier d'options client. Vous spécifiez ces options pour chaque client que vous souhaitez surveiller.

Lors de la surveillance des performances sur les ordinateurs UNIX et Linux, définissez la limite du descripteur de fichiers ouverts sur 1024 au minimum, en utilisant la commande suivante :

```
ulimit -n 1024
```

Pour configurer les options du moniteur de performances client, procédez comme suit.

1. Ouvrez le fichier d'options client pour chaque client que vous surveillez. Selon votre configuration, les options client se trouvent dans l'un des fichiers suivants :
   - `dsm.opt`
   - `dsm.sys`
2. Ajoutez les options suivantes au fichier d'options client :

   ```
   PERFMONTCPSERVERADDRESS
   PERFMONTCPPORT
   PERFMONCOMMTIMEOUT
   ```

PERFMONTCPSERVERADDRESS

L'option PERFMONTCPSERVERADDRESS indique le nom d'hôte ou l'adresse IP du système où le moniteur de performances client est installé.

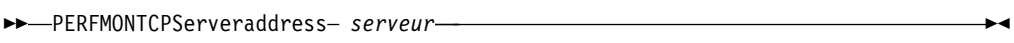
Clients pris en charge

Cette option peut être utilisée sur n'importe quelle plateforme et est prise en charge pour tous les clients.

Fichier d'options

Définissez cette option dans le fichier d'options client (`dsm.opt` ou `dsm.sys`).

Syntaxe



Paramètres

serveur
Nom d'hôte du serveur ou adresse IP du système sur lequel le moniteur de performances client est installé (il s'agit du même serveur qui exécute le centre d'administration).

Exemples

Fichier d'options :

```
PERFMONTCPSERVERADDRESS 131.222.10.5
```

Ligne de commande :
Cette option ne peut pas être définie à l'aide de la ligne de commande.

PERFMONTCPPORT

Numéro de port sur lequel le moniteur de performances client est en mode écoute pour les données de performances provenant des clients.

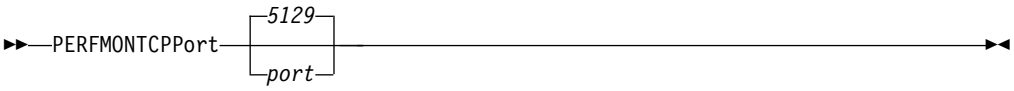
Clients pris en charge

Cette option peut être utilisée sur n'importe quelle plateforme et est prise en charge pour tous les clients.

Fichier d'options

Définissez cette option dans le fichier d'options client (`dsm.opt` ou `dsm.sys`).

Syntaxe



Paramètres

port
Port surveillé pour les données de performances du client. Le port 5129 est utilisé par défaut.

Exemples

Fichier d'options :

```
PERFMONTCPPPORT 5000
```

Ligne de commande :

Cette option ne peut pas être définie à l'aide de la ligne de commande.

PERFMONCOMMTIMEOUT

Indique la durée maximale, en secondes, pendant laquelle l'appel dsmTerminate attend que les données de performances arrivent après la fin d'une session.

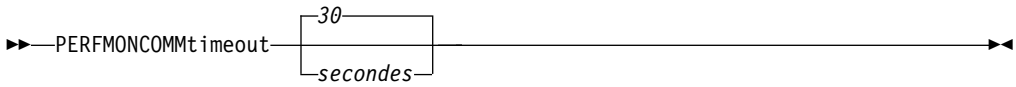
Clients pris en charge

Cette option peut être utilisée sur n'importe quelle plateforme et est prise en charge pour tous les clients.

Fichier d'options

Définissez cette option dans le fichier d'options client (`dsm.opt` ou `dsm.sys`).

Syntaxe



Paramètres

secondes

Durée d'attente avant que les données de performances restantes n'arrivent, avant la fin de la session.

Exemples

Fichier d'options :

```
PERFMONCOMMTIMEOUT 60
```

Ligne de commande :

Cette option ne peut pas être définie à l'aide de la ligne de commande.

Envoi d'objets au serveur

Les applications client peuvent envoyer des données ou des objets nommés ainsi que leurs données correspondantes vers la mémoire du serveur IBM Spectrum Protect à l'aide des fonctions de sauvegarde ou d'archivage de l'API. Les composants de sauvegarde et d'archivage du système permettent l'utilisation de plusieurs procédures de gestion pour les données envoyées vers l'espace de stockage.

L'attribut de taille estimée correspond à l'estimation de la taille totale de l'objet données à envoyer au serveur. Si la taille exacte de l'objet n'est pas connue de l'application, attribuez une valeur supérieure à la fonction *sizeEstimate*. Si l'estimation est inférieure à la taille en cours, le serveur IBM Spectrum Protect utilise des ressources supplémentaires pour gérer les allocations d'espace additionnelles.

Conseils :

- Soyez aussi précis que possible pour définir cette taille. Le serveur utilise cet attribut pour gérer efficacement l'allocation d'espace et le placement d'objet au sein de ses ressources de stockage.
- Si l'estimation est inférieure à la taille réelle, un serveur doté d'une fonction de mise en cache n'alloue pas d'espace supplémentaire et interrompt l'envoi.

Vous pouvez rencontrer des problèmes si la valeur de l'option *sizeEstimate* est trop importante. Il se peut que le serveur ne dispose pas de suffisamment d'espace pour la taille estimée mais détienne assez d'espace pour la taille effective, ou qu'il utilise des unités plus lentes.

Vous pouvez sauvegarder ou archiver des objets dont la taille est supérieure à 2 Go. Les objets peuvent être compressés ou non compressés.

Pour lancer une opération d'envoi, appelez **dsmSendObj**. Si le volume de données que vous devez envoyer en une seule opération est trop important, vous pouvez appeler **dsmSendData** plusieurs fois pour transférer les informations restantes. Appelez **dsmEndSendObj** pour terminer l'opération d'envoi.

Analyse des objets de sauvegarde et d'archivage

Le composant de sauvegarde du système IBM Spectrum Protect prend en charge plusieurs versions d'objets nommés qui sont stockés sur le serveur.

Un contrôle des versions est effectué pour chaque objet sauvegardé sur le serveur dont le nom correspond à celui d'un objet ayant déjà été sauvegardé par le client sur ce même serveur. Les objets sont actifs ou inactifs sur le serveur. La version active la plus récente d'un objet sur le serveur est considérée comme étant la version active. Tout autre objet doté du même nom, qu'il s'agisse d'une version antérieure ou d'une version désactivée, est considéré comme étant inactif. Les constructions de classe de gestion permettent de définir plusieurs critères de gestion. Ces critères sont attribués aux versions actives et inactives stockées sur le serveur.

Le tableau 11 répertorie les zones de groupe de copie s'appliquant aux états actifs et inactifs :

Tableau 11. Zones du groupe de paramètres de sauvegarde

Zone	Description
VEREXISTS	Indique le nombre de versions inactives si des versions actives existent.
VERDELETED	Indique le nombre de versions inactives s'il n'existe pas de versions actives.
RETEXTRA	Indique le délai de conservation des versions inactives.
RETONLY	Indique le délai de conservation des versions inactives les plus récentes s'il n'existe pas de versions actives.

Si les versions de sauvegarde sont dotées de noms uniques, par exemple l'utilisation d'un horodatage dans le nom, la gestion des versions n'est pas effectuée automatiquement : tous les objets sont actifs. Les versions actives n'ont pas de délai d'expiration. Par conséquent, une application peut désactiver ces versions à l'aide de l'appel à **dsmDeleteObj**. Dans ce cas, les versions désactivées

doivent arriver à expiration le plus tôt possible. L'utilisateur définit un groupe de copies de sauvegarde à l'aide des options VERDELETED et RETONLY définies sur 0.

Le composant d'archivage du système IBM Spectrum Protect autorise le stockage d'objets sur le serveur, les contrôles du délai de conservation ou d'expiration étant activés plutôt que les contrôles de versions. Chaque objet stocké est unique, et ce même si le nom d'un objet correspond au nom d'un objet déjà archivé. Les objets archivés sont dotés d'une zone de description associé aux métadonnées pouvant être utilisées au cours d'une requête pour identifier un objet spécifique.

Un ID objet unique est attribué à chaque objet stocké sur un serveur IBM Spectrum Protect. Il n'y a aucune garantie que la valeur d'origine reste valide pendant la durée de vie de l'objet (en particulier après une opération d'exportation ou d'importation). Par conséquent, une application ne doit pas interroger et sauvegarder l'ID objet original à des fins d'utilisation lors des opérations de restauration ultérieures. L'application devrait plutôt enregistrer le nom de l'objet et insérer une date. Vous pouvez utiliser ces informations au cours d'une restauration pour l'interrogation des objets et la vérification de la date entrée. L'ID objet en cours peut être alors utilisé pour restaurer l'objet.

Compression

Les options de configuration d'un poste donné et l'option **dsmSendObj** `objCompressed`, déterminent si IBM Spectrum Protect compresse l'objet au cours d'un envoi. De plus, les objets pour lesquels la valeur de la zone `sizeEstimate` est inférieure à celle de l'option `DSM_MIN_COMPRESS_SIZE` ne sont jamais compressés.

Un l'objet ne peut pas être compressé deux fois (`objCompressed=bTrue`). Si l'objet n'est pas compressé, IBM Spectrum Protect décide s'il doit l'être, en fonction des valeurs de l'option de compression définies par l'administrateur et configurées dans les sources de configuration de l'API.

L'administrateur peut changer les seuils de compression sur le serveur en utilisant la commande REGISTER NODE (`compression=yes`, `no`, ou `client-determined`). Si `client-determined` est sélectionné, le comportement de la compression est déterminé à l'aide de la valeur de l'option compression dans les sources de configuration.

Il se peut que la taille de certains types de données, par exemple les données compressées, augmente lorsque ces données sont traitées à l'aide des algorithmes de compression. Dans ce cas, le code retour DSM_RC_COMPRESS_GREW est généré. Si vous pensez que ceci peut se produire mais souhaitez poursuivre l'opération d'envoi, informez les utilisateurs finals qu'ils doivent définir l'option suivante dans leur fichier d'options :

```
COMPRESSAlways Yes
```

Si le code de retour DSM_RC_COMPRESS_GREW s'affiche au cours d'un appel de fonction **dsmSendData**, pour lequel l'option de compression a été activée, recommencez et envoyez l'objet sans le compresser. Pour appliquer cette règle, définissez **dsmSendObj** `ObjAttr.objCompressed` sur `bTrue`.

Les informations sur le comportement de compression en cours pendant un appel de fonction **dsmSendObj** sont transmises via l'appel à **dsmEndSendObjEx**. L'option `objCompressed` indique si la compression est terminée. `totalBytesSent` correspond au nombre d'octets envoyés par l'application. `totalCompressedSize` indique le

nombre d'octets après la compression. L'appel à **dsmEndSendObjEx** comporte également une zone `totalLFBytesSent` contenant le nombre total d'octets envoyés hors réseau local.

Avertissement : Si votre application prévoie d'utiliser une restauration ou une récupération d'objet partiel, vous ne pouvez pas compresser les données lors de leur envoi. Pour appliquer cette règle, définissez **dsmSendObj** `ObjAttr.objCompressed` sur `bTrue`.

Type de compression

Le type de compression utilisé par le client est déterminé par la combinaison de la compression et du dédoublonnage de données côté client utilisé durant la sauvegarde ou l'archivage.

L'algorithme de compression qui est utilisé par le client est retourné dans un rapport par l'API dans une nouvelle zone des structures **qryRespArchiveData** et **qryRespBackupData** :

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

Les types de compression suivants font l'objet d'un rapport :

LZ4 Méthode de compression plus rapide et plus efficace que le client utilise quand un objet dédoublonné par le client est envoyé à un pool de stockage de conteneur compatible avec LZ4 sur le serveur IBM Spectrum Protect. Le serveur doit avoir la version 7.1.5 ou une version ultérieure et doit utiliser les pool de stockage de conteneur. La compression LZ4 côté client est uniquement utilisée le dédoublonnage de données côté client est activé.

LZW Type de compression classique utilisé par le client dans l'une des situations suivantes :

- Des objets dédoublonnés par le client sont envoyés vers des pools de stockage classiques (qui ne sont pas de type conteneur) sur le serveur.
- L'objet client ne subit pas de dédoublonnage de données côté client.
- L'objet client n'est soumis qu'à un dédoublonnage de données côté serveur classique.

Zone vide
L'objet n'est pas compressé par le client. L'objet n'est pas compressé car l'option `compression` est définie sur *no* ou l'option n'est pas spécifié au cours du processus de sauvegarde ou d'archivage. Bien que l'objet ne soit pas compressé par le client, il peut être compressé par le serveur.

Le type de compression n'est pas configurable. Il est déterminé par le client de sauvegarde-archivage lors du traitement de la sauvegarde ou de l'archivage.

Exemple

L'exemple suivant montre la zone relative au type de compression dans la sortie des requêtes de sauvegarde et d'archivage depuis l'application exemple 64 bits **dapismp** :

```
Enter selection ==>1
                       Filespace:\fs1
                       Highlevel:\hl
                        Lowlevel:\ll
             Object Type(D/F/A):f
   Active(A),Inactive(I),Both(B):a
If root, query all owners? (Y/N):
              Object Owner Name:
```

```
   point in time date (MMDDYYYY):
   point in time time     (hhmm):
     Show detailed output? (Y/N):y
On Restore, Wait for mount?(Y/N):
Are the above responses correct (y/n/q)?


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

Suppression de la copie de la mémoire tampon

La fonction de suppression de la copie de la mémoire tampon supprime la copie des mémoires tampon de données entre une application et le serveur IBM Spectrum Protect, ce qui donne lieu à une meilleure utilisation du processeur. Pour un effet optimal, utilisez cette approche dans un environnement hors réseau local.

Les mémoires tampon pour le transfert de données sont attribuées par IBM Spectrum Protect et un indicateur est renvoyé à l'application. L'application place les données dans la mémoire tampon fournie et cette mémoire est transmise des niveaux de communication à l'agent de stockage à l'aide d'une mémoire partagée. Les données sont ensuite transférées vers l'unité de bande, qui supprime les copies de données. Cette fonction peut être utilisée lors des opérations de sauvegarde ou d'archivage.

Avertissement : Lorsque cette méthode est utilisée, soyez particulièrement vigilants quant à la taille des mémoires tampon et la gestion de ces mémoires. Les mémoires tampon sont partagées entre les composants et tout remplacement de la mémoire suite à une erreur de programmation génère des erreurs fatales.

La séquence globale des appels pour la sauvegarde-archivage est la suivante :

```
dsmInitEx (UseTsmBuffers = True, numTsmBuffers = [how many IBM Spectrum Protect
           -allocated buffers the application needs to allocate])
dsmBeginTxn
pour chaque objet dans la txn
  dsmBindMC
    dsmSendObject
      dsmRequestBuffer
      dsmSendBufferData (sends and release the buffer used)
    dsmEndSendObjEx
dsmEndTxn
pour chaque tampon encore utilisé
      dsmReleaseBuffer
dsmTerminate
```

La fonction **dsmRequestBuffer** peut être appelée plusieurs fois, selon la valeur spécifiée par l'option numTsmBuffers. Une application peut être dotée de deux unités d'exécution : une unité d'exécution fournisseur qui fournit des données aux mémoires tampon et une unité d'exécution consommateur qui envoie ces mémoires tampon à IBM Spectrum Protect à l'aide de l'appel à **dsmSendBufferData**. Lorsque la fonction **dsmRequestBuffer** est appelée et que la valeur de **numTsmBuffers** est atteinte, l'appel de fonction **dsmRequestBuffer** est bloqué tant qu'une mémoire tampon n'est pas libérée. Il est possible de libérer la mémoire tampon en appelant la fonction **dsmSendBufferData**, qui envoie et libère une mémoire tampon ou en appelant la fonction **dsmReleaseBuffer**. Pour plus d'informations, voir callbuff.c dans le répertoire des exemples d'API.

Si une erreur se produit au cours de l'envoi, l'application doit libérer toutes les mémoires tampon et fermer la session. Par exemple :

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

Si une application appelle **dsmTerminate** et qu'elle détient une mémoire tampon, l'API n'existe pas. Le code suivant est renvoyé : DSM_RC_CANNOT_EXIT_MUST_RELEASE_BUFFER. Si l'application ne peut pas libérer la mémoire tampon, l'application doit arrêter le processus afin de déclencher le nettoyage.

Elimination, restauration et récupération de la copie de la mémoire tampon

Le serveur IBM Spectrum Protect contrôle le volume de données à placer dans la mémoire tampon, en fonction de l'optimisation de l'accès aux bandes sur lesquelles la restauration et la récupération sont activées. Cette méthode n'est pas aussi avantageuse que la méthode standard d'obtention de données. Lors du prototypage, vérifiez les performances de la méthode de suppression de la copie de la mémoire tampon et utilisez cette méthode uniquement si vous constatez une amélioration significative.

Le volume maximal de données contenues dans une mémoire tampon unique qui est renvoyé par le serveur IBM Spectrum Protect est 256 Ko – supplément d'en-tête). Par conséquent, seules les applications effectuant des écritures de mémoire tampon minimes bénéficient de ce mécanisme d'extraction de données. Le nombre d'octets dans la mémoire tampon fournie doit être contrôlé par l'application, en fonction de la taille de l'objet, du réseau ainsi que d'autres restrictions. Dans certains cas, l'utilisation de la fonction de suppression de la copie de la mémoire tampon ne s'avère pas aussi avantageuse qu'une opération de restauration standard. L'API conserve généralement les données en cache et les enregistrements renvoyés à l'application sont de longueur fixe. Cela permet ainsi à l'application de contrôler le nombre d'écritures de données sur le disque.

Si la suppression de la copie de la mémoire tampon est utilisée, un mécanisme de mise en cache doit être créé pour les mémoires tampon dont la taille est inférieure à la taille de mémoire tampon en écriture définie. Ainsi, si une application écrit sur disque des blocs de données de 64 Ko, elle doit effectuer les actions suivantes :

1. Appeler **dsmGetBufferData**.
2. Ecrire dans des blocs de 64 Ko.
3. Une fois arrivé au dernier bloc, copier les données dans un **tempBuff**, effectuer un autre appel **dsmGetBufferData** et remplir le **tempBuff** à l'aide des données restantes.

4. Poursuivre l'écriture dans des blocs de 64 Ko :

```
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
```

Dans cet exemple, six écritures sur disque sont directes et 1 est mise en cache.

La séquence globale des appels pour la restauration et l'extraction est la suivante :

dsmInitEx (UseTsmBuffers = True numTsmBuffers = nombre de mémoires tampon attribuées par l'application).

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

Chaque appel à **dsmGetBufferData** est suivi d'un appel à **dsmReleaseBuffer**. Il n'est pas nécessaire que l'appel à **dsmGetBufferData** et l'appel à **dsmReleaseBuffer** soient des appels consécutifs. Une application peut passer plusieurs appels **dsmGetBufferData** pour obtenir diverses mémoires tampon, puis effectuer les appels à **dsmReleaseBuffer** ultérieurement. Pour des informations sur les codes exemple utilisant cette fonction, reportez-vous à `callbuff.c` dans le répertoire des exemples d'API.

Restriction : Etant donné que la mémoire tampon est fournie par l'API et que l'utilisation du processeur doit être minimisée, un traitement supplémentaire des données de la mémoire tampon n'est pas autorisé. L'application ne peut pas utiliser le chiffrement et la compression lors de la suppression de la copie de la mémoire tampon car ces opérations nécessitent des copies ainsi que le traitement des données.

Implémentez à la fois le chemin de transfert de données standard ainsi que la suppression de la copie de la mémoire tampon afin de permettre aux utilisateurs de basculer d'un chemin à l'autre, en fonction de leurs besoins. Pour compresser ou chiffrer des données, l'utilisateur doit utiliser le mécanisme existant. S'il existe une restriction au niveau du processeur, servez-vous du nouveau mécanisme. Ces mécanismes sont complémentaires et n'ont pas été conçus pour se remplacer l'un l'autre.

Fonction de chiffrement d'API

Deux méthodes sont disponibles pour chiffrer les données : chiffrement géré par l'application et chiffrement du client IBM Spectrum Protect.

Sélectionnez et utilisez une seule de ces méthodes pour chiffrer des données. Les méthodes sont mutuellement exclusives et si vous chiffrez des données en utilisant les deux méthodes, vous ne pourrez pas restaurer ou récupérer certaines données. Par exemple, supposons qu'une application utilise le chiffrement géré par l'application pour chiffrer l'objet A, puis le chiffrement du client IBM Spectrum Protect pour chiffrer l'objet B. Lors d'une opération de restauration, si l'application

a défini l'option d'utiliser le chiffrement du client IBM Spectrum Protect et tente de restaurer les deux objets, seul l'objet B pourra être restauré. L'objet A ne pourra pas être restauré car il a été chiffré par l'application, et non par le client.

Quelle que soit la méthode de chiffrement utilisée, le IBM Spectrum Protect doit permettre l'authentification par mot de passe. Par défaut, le serveur utilise `SET AUTHENTICATION ON`.

L'API utilise soit le chiffrement AES 128 bits, soit le chiffrement AES 256 bits. Le chiffrement AES 256 bits offre un chiffrement de données supérieur à celui du chiffrement AES 128 bits. Les fichiers sauvegardés à l'aide du chiffrement AES 256 bits ne peuvent pas être restaurés avec un client de niveau antérieur. Vous pouvez exécuter le chiffrement, que la compression soit activée ou pas. Si vous utilisez le chiffrement, les fonctions de restauration et récupération partielles d'objets et les fonctions d'élimination de la copie de mémoire tampon ne peuvent pas être utilisées.

Chiffrement géré par l'application

Lorsque le chiffrement géré par l'application est utilisé, l'application fournit le mot de passe de la clé à l'API (à l'aide de la clé DSM_ENCRYPT_USER) et le gère.

Avertissement : Si la clé de chiffrement n'est pas sauvegardée et que vous l'avez oubliée, vos données ne peuvent pas être récupérées.

L'application indique le mot de passe de la clé lors de l'appel à **dsmInitEx** et doit fournir le mot de passe correct au cours de la restauration.

Avertissement : Si le mot de passe de la clé est perdu, il n'existe aucun moyen de restaurer les données.

Le même mot de passe doit être utilisé pour les opérations de sauvegarde et de restauration (ou d'archivage et d'extraction) pour le même objet. Cette méthode n'a pas de lien de dépendance par rapport au niveau du serveur IBM Spectrum Protect. Pour configurer cette méthode, procédez comme suit.

1. Définissez la variable `bEncryptKeyEnabled` sur `bTrue` lors de l'appel à **dsmInitEx** et définissez la variable `encryptionPasswordP` afin que celle-ci pointe vers la chaîne dotée du mot de passe de la clé de chiffrement.
2. Définissez l'option `include.encrypt` pour les objets à `chiffrer`. Par exemple, pour chiffrer toutes les données, définissez les options :

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

   et

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

   Pour chiffrer l'objet `/FS1/DB2/FULL`, définissez :

   ```
   include.encrypt /FS1/DB2/FULL
   ```
3. Définissez `ENCRYPTKEY=PROMPT|SAVE` dans la chaîne d'option transmise à l'API dans l'appel de **dsmInitEx** sur Windows. Cette option peut également être définie dans `dsm.opt` (Windows) ou dans `dsm.sys` (UNIX ou Linux).

Par défaut, l'option `encryptkey` est définie sur `prompt`. Ce paramètre permet d'éviter le stockage automatique de la clé. Si la sauvegarde `encryptkey` est indiquée, la clé est stockée par IBM Spectrum Protect sur la machine locale mais une seule clé est valide pour toutes les opérations IBM Spectrum Protect dotées du même nom de poste.

Une fois l'objet envoyé, l'appel de **dsmEndSendObjEx** indique si un objet a été chiffré ainsi que la méthode utilisée. Les valeurs possibles dans la zone *encryptionType* sont :

- DSM_ENCRYPT_NO
- DSM_ENCRYPT_USER
- DSM_ENCRYPT_CLIENTENCRKEY

La table suivante répertorie les types de fonctions de chiffrement d'API, les prérequis ainsi que les fonctions disponibles.

Tableau 12. Types de fonctions de chiffrement d'API, prérequis et fonctions disponibles

Type	Prérequis	Fonction disponible
ENCRYPTIONTYPE	Aucune	Définissez ENCRYPTIONTYPE dans la chaîne d'option transmise à l'API dans l'appel de **dsmInitEx** sur Windows. La valeur par défaut est ENCRYPTIONTYPE=AES128.
EncryptKey=save	Aucune	API et sauvegarde-archivage
EncryptKey=prompt	Aucune	API et sauvegarde-archivage
EncryptKey=generate	Aucune	API et sauvegarde-archivage
EnableClientEncryptKey	Aucune	API uniquement

Remarque : Il est recommandé de définir l'authentification du serveur sur ON. Si l'authentification est désactivée, la clé n'est pas chiffrée mais les données le sont. Toutefois, ceci n'est pas recommandé.

Le tableau 13 indique la façon dont les utilisateurs autorisés et non autorisés peuvent chiffrer ou déchiffrer des données au cours d'une opération de sauvegarde ou de restauration en fonction de la valeur indiquée pour l'option passwordaccess. Le fichier TSM.PWD doit exister pour exécuter les opérations d'utilisateur autorisé et d'utilisateur non autorisé suivantes. L'utilisateur autorisé crée le fichier TSM.PWD et définit l'option encryptkey à sauvegarder et l'option passwordaccess à générer.

Tableau 13. Chiffrement ou déchiffrement des données à l'aide d'une clé gérée sous UNIX ou Linux

Opération	Option passwordaccess	Option encryptkey	Résultat
Sauvegarde utilisateur autorisé	generate	save	Données chiffrées.
	generate	prompt	Données chiffrées si encryptionPasswordP contient un mot de passe de chiffrement.
	prompt	save	Données chiffrées si encryptionPasswordP contient un mot de passe de chiffrement.
	prompt	prompt	Données chiffrées si encryptionPasswordP contient un mot de passe de chiffrement.
Restauration utilisateur autorisé	generate	save	Données chiffrées.
	generate	prompt	Données chiffrées si encryptionPasswordP contient un mot de passe de chiffrement.
	prompt	save	Données chiffrées si encryptionPasswordP contient un mot de passe de chiffrement.
	prompt	prompt	Données chiffrées si encryptionPasswordP contient un mot de passe de chiffrement.

Tableau 13. Chiffrement ou déchiffrement des données à l'aide d'une clé gérée sous UNIX ou Linux (suite)

Opération	Option passwordaccess	Option encryptkey	Résultat
Sauvegarde de l'utilisateur non autorisé	generate	save	Données chiffrées.
	generate	prompt	Données chiffrées si encryptionPasswordP contient un mot de passe de chiffrement.
	prompt	save	Données chiffrées si encryptionPasswordP contient un mot de passe de chiffrement.
	prompt	prompt	Données chiffrées si encryptionPasswordP contient un mot de passe de chiffrement.
Restauration utilisateur non autorisé	generate	save	Données chiffrées.
	generate	prompt	Données chiffrées si encryptionPasswordP contient un mot de passe de chiffrement.
	prompt	save	Données chiffrées si encryptionPasswordP contient un mot de passe de chiffrement.
	prompt	prompt	Données chiffrées si encryptionPasswordP contient un mot de passe de chiffrement.

Chiffrement du client IBM Spectrum Protect

Le chiffrement du client IBM Spectrum Protect utilise la clé gérée par la valeur DSM_ENCRYPT_CLIENTENCRKEY pour protéger vos données. Le chiffrement du client est transparent pour l'application qui utilise l'API, mais les opérations de restauration et d'extraction partielles d'objets ne sont pas possibles pour les objets chiffrés ou compressés.

Pour le chiffrement du client IBM Spectrum Protect comme pour le chiffrement géré par l'application, le mot de passe de chiffrement fait référence à une valeur de chaîne utilisée pour générer la clé de chiffrement actuelle. La valeur de l'option de mot de passe de chiffrement a une longueur comprise entre 1 et 63 caractères, mais la clé générée est toujours de 8 octets pour 56 DES, 16 octets pour 128 AES et 32 octets pour 256 AES.

Avertissement : Si la clé de chiffrement n'est pas disponible, les données ne peuvent être ni restaurées, ni récupérées. Lorsque vous utilisez ENABLECLIENTENCRYPTKEY pour le chiffrement, la clé est stockée sur la base de données du serveur. Par conséquent, la base de données du serveur doit être existante et dotée de valeurs appropriées afin que la restauration des objets utilisant cette méthode soit correcte. Vérifiez qu'une sauvegarde régulière de la base de données du serveur soit effectuée afin d'empêcher toute perte de données.

Cette méthode est la plus facile à implémenter. Une clé de chiffrement aléatoire est générée par session et celle-ci est stockée sur le serveur IBM Spectrum Protect, avec l'objet dans la base de données du serveur. Pendant la restauration, la clé stockée est utilisée pour le déchiffrement. Si cette méthode est utilisée, seul IBM Spectrum Protect gère la clé. Celle-ci n'est en aucun cas utilisée par l'application. Puisque la clé est stockée dans la base de données du serveur, vous devez disposer d'une base de données IBM Spectrum Protect valide pour une opération de restauration d'un objet chiffré. La clé est également chiffrée lorsqu'elle est transmise de l'API au serveur. La transmission de la clé est un processus sécurisé et lorsque la clé est stockée dans la base de données du serveur IBM Spectrum Protect, elle est chiffrée. La seule fois où la clé n'est pas chiffrée est lors d'une exportation de flux de données car les données d'un poste sont exportées d'un serveur à l'autre.

Pour activer le chiffrement du client IBM Spectrum Protect, procédez comme suit :

1. Spécifiez `-ENABLECLIENTENCRYPTKEY=YES` dans la chaîne d'option transmise à l'API sur l'appel dsmInitEx ou définissez l'option sur le fichier d'options du système `dsm.opt` (Windows) ou `dsm.sys` (UNIX ou Linux).
2. Définissez l'option `include.encrypt` pour les objets à chiffrer. Par exemple, pour chiffrer toutes les données, définissez les options :

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

   et

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

   Pour chiffrer l'objet `/FS1/DB2/FULL`, définissez :

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

Dédoublonnage de données

Le dédoublonnage de données est une méthode permettant de réduire les besoins en termes de stockage en éliminant les données redondantes.

Présentation

Deux types de dédoublonnage de données sont disponibles sur IBM Spectrum Protect : le *dédoublonnage de données côté client* et le *dédoublonnage de données côté serveur*.

Le *dédoublonnage de données coté client* est une technique de dédoublonnage de données utilisée sur le client de sauvegarde-archivage pour supprimer les données redondantes lors du processus de sauvegarde et d'archivage, avant que les données soient transférées sur le serveur IBM Spectrum Protect. L'utilisation du dédoublonnage côté client permet de réduire la quantité de données qui sont envoyées sur un réseau local.

Le *dédoublonnage côté serveur* est une technique réalisée par le serveur. L'administrateur IBM Spectrum Protect peut spécifier l'emplacement du dédoublonnage (client ou serveur) à utiliser avec le paramètre **DEDUP** sur la commande du serveur **REGISTER NODE** ou **UPDATE NODE**.

Améliorations

Le dédoublonnage de données côté client permet les opérations suivantes :

- Exclure des fichiers spécifiques d'un client du dédoublonnage de données.
- Activer un cache de dédoublonnage de données pour réduire le trafic réseau entre le client et le serveur. Le cache contient les extensions envoyées au serveur lors d'opérations de sauvegarde incrémentielle précédentes. Au lieu d'interroger le serveur pour savoir si une extension existe, le client interroge son cache.

  Indiquez une taille et un emplacement pour un cache client. En cas d'incohérence entre le serveur et le cache local, le cache local est supprimé et reconstruit.

  Remarque : Pour les applications utilisant l'interface de programme d'application IBM Spectrum Protect, le cache de dédoublonnage de données ne doit pas être utilisé en raison du risque d'échec de sauvegarde, car il n'est pas synchronisé avec le serveur IBM Spectrum Protect. Si plusieurs sessions de clients de sauvegarde-archivages simultanées sont configurées, un cache séparé doit être configuré pour chaque session.
- Activez le dédoublonnage de données côté client et la compression pour réduire la quantité de données stockées par le serveur. Chaque extension est compressée

avant d'être envoyée au serveur. Il s'agit d'un compromis entre économie en espace de stockage et puissance de traitement utilisée pour la compression des données client. En règle générale, si vous compressez et dédoublonnez les données sur le système client, vous utilisez environ deux fois plus de puissance de traitement que pour le dédoublonnage seul.

Le serveur peut utiliser des données compressées et dédoublonnées. De plus, les clients de sauvegarde-archivage antérieurs à la version 6.2 peuvent restaurer des données dédoublonnées et compressées.

Le dédoublonnage côté client utilise le processus suivant :

- Le client crée des extensions. Les *extensions* sont des éléments de fichier qui sont comparés à d'autres extensions de fichier pour identifier les doublons.
- Le client et le serveur fonctionnent ensemble pour identifier les extensions en doublon. Le client envoie les extensions non dupliquées vers le serveur.
- Les opérations ultérieures de dédoublonnage créent de nouvelles extensions. Certaines, voire toutes ces extensions peuvent correspondre à des extensions créées lors d'opérations de dédoublonnage antérieures et envoyées au serveur. Les extensions correspondantes ne sont pas renvoyées au serveur.

Avantages

Le dédoublonnage de données côté client présente plusieurs avantages :

- Il permet de réduire la quantité de données envoyées sur le réseau local.
- La puissance de traitement nécessaire pour identifier les données en double est déchargée du serveur aux postes client. Le dédoublonnage de données côté serveur est toujours activé pour les pools de stockage activés pour le dédoublonnage. Toutefois, les fichiers se trouvant dans les pools de stockage activés pour le dédoublonnage et ayant été dédoublonnés par le client ne nécessitent pas de traitement supplémentaire.
- La puissance de traitement nécessaire pour supprimer les données en double sur le serveur est éliminée, ce qui permet d'économiser immédiatement de l'espace sur le serveur.

Le dédoublonnage de données coté client peut présenter un inconvénient. Le serveur ne dispose pas d'exemplaires complets des fichiers client *tant que* vous n'avez pas sauvegardé les pools de stockage principaux qui contiennent les extensions client sur un pool de stockage de copie non dédoublonné. (Les *extensions* font partie d'un fichier créé lors du processus de dédoublonnage des données.) Au cours de la sauvegarde du pool de stockage dans un pool de stockage non dédoublonné, les extensions client sont regroupées dans des fichiers contigus.

Par défaut, les pools de stockage à accès séquentiel principaux configurés pour le dédoublonnage de données doivent être sauvegardés sur des pools de stockage de copie non dédoublonnés avant de pouvoir être récupérés et avant que des données en double ne puissent être supprimées. La valeur par défaut assure que le serveur dispose de copies de tous les fichiers à la fois, soit dans un pool de stockage principal, soit dans un pool de stockage de copie.

Important : Pour réduire encore les données, vous pouvez activer à la fois le dédoublonnage côté client et la compression. Chaque extension est compressée avant d'être envoyée au serveur. La compression permet d'économiser de l'espace, mais augmente la durée de traitement sur le poste de travail du client.

Les options suivantes concernent le dédoublonnage de données :

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

Dédoublonnage de données côté client d'API

Le *dédoublonnage côté client* est utilisé par l'API sur le client de sauvegarde-archivage, pour éliminer les données redondantes au cours des opérations de sauvegarde et d'archivage avant que les données ne soient transférées au serveur IBM Spectrum Protect.

Le dédoublonnage côté client est utilisé par l'API afin d'éliminer les données redondantes au cours des opérations de sauvegarde et d'archivage avant que les données ne soient transférée au serveur IBM Spectrum Protect. L'utilisation du dédoublonnage côté client permet de réduire la quantité de données qui sont envoyées sur un réseau local. L'utilisation du dédoublonnage côté client peut aussi réduire l'espace de stockage sur le serveur IBM Spectrum Protect.

Lorsque le client est activé pour le dédoublonnage côté client et que vous effectuez une opération de sauvegarde ou d'archivage, les données sont envoyées au serveur en tant qu'extensions. La prochaine fois qu'une opération de sauvegarde ou d'archivage sera réalisée, le client et le serveur identifieront les extensions de données qui ont déjà été sauvegardées ou archivées, et enverront uniquement les extensions uniques des données au serveur.

Pour le dédoublonnage de données côté client, le serveur et l'API doivent correspondre à la version 6.2 ou à une version ultérieure.

Avant d'utiliser le dédoublonnage de données côté client pour sauvegarder ou archiver vos données, le système doit répondre aux exigences suivantes :

- L'option `deduplication` doit être activée sur le client.
- Le serveur doit activer le client pour le dédoublonnage côté client avec le paramètre **DEDUP=CLIENTORSERVER** sur la commande **REGISTER NODE** ou **UPDATE NODE**.
- Le pool de stockage cible pour les données doit être activé pour le dédoublonnage. Le pool de stockage activé pour le dédoublonnage est uniquement de type dispositif de fichiers.
- Assurez-vous que les fichiers sont liés à la classe de gestion correcte.
- Un fichier peut être exclu du dédoublonnage côté client. Par défaut, tous les fichiers sont inclus.
- Les fichiers doivent être supérieurs à 2 Ko.
- Le serveur peut limiter la taille maximale de transaction pour le dédoublonnage en définissant l'option `CLIENTDEDUPTXNLIMIT` sur le serveur. Consultez la documentation du serveur pour plus d'informations sur cette option.

Si certaines de ces exigences ne sont pas satisfaites, les données sont traitées normalement, sans dédoublonnage côté client.

Voici quelques exemples de restrictions au dédoublonnage :

- Le déplacement de données hors LAN et le dédoublonnage côté client s'excluent mutuellement. Si vous activez le déplacement de données hors LAN et le dédoublonnage côté client, les opérations de déplacement de données hors LAN et de dédoublonnage côté client sont ignorées.
- Le chiffrement et le dédoublonnage côté client s'excluent mutuellement. Si vou activez à la fois le chiffrement et le dédoublonnage côté client, les opérations de chiffrement se terminent et le dédoublonnage côté client est ignoré. Les fichiers chiffrés ainsi que les fichiers éligibles pour le dédoublonnage côté client peuvent être traités au cours d'une même opération, mais sont traités lors de transactions distinctes.

  Exigences :

  1. Dans toute transaction, tous les fichiers doivent être inclus ou exclus pour le dédoublonnage. Si la transaction concerne des fichiers mixtes, elle échoue et un code retour `DSM_RC_NEEDTO_ENDTXN` est envoyé par l'API.
  2. Utilisez le chiffrement des unités de stockage en même temps que le dédoublonnage côté client. Dans la mesure où le chiffrement SSL est utilisé conjointement avec le dédoublonnage côté client, le chiffrement du client n'est pas nécessaire.
- Les fonctions suivantes ne sont pas disponibles pour le dédoublonnage côté client :
  - Client HSM (IBM Spectrum Protect for Space Management)
  - Tampon API partagé
  - NAS
  - Sauvegarde de sous-fichier
- L'élimination de copie dans la mémoire tampon ne peut pas être utilisée avec les transformations de données telles que la compression, le chiffrement et le dédoublonnage.
- Si vous utilisez le dédoublonnage côté client, l'API détecte et fait échouer (with RC=254) les sauvegardes d'extensions de fichier marqués comme expirés sur le serveur au cours de l'envoi de données au serveur. Si vous souhaitez renouveler l'opération, vous devez inclure cette programmation à l'application appelante.
- Les opérations d'écriture simultanée sur le serveur prévalent sur le dédoublonnage côté client. Si les opérations d'écriture simultanée sont activées, le dédoublonnage côté client n'est pas effectué.

  Restriction : Lorsque le dédoublonnage côté client est activé, l'API ne peut pas récupérer d'un état dans lequel le serveur manque d'espace de stockage sur le pool cible, même si un pool suivant est défini. Un code raison d'arrêt DSM_RS_ABORT_DESTINATION_POOL_CHANGED est retourné et l'opération échoue. Deux méthodes permettent de récupérer de cette situation :

  1. Demandez à l'administrateur d'ajouter des volumes scratch au pool de fichiers original.
  2. Renouvelez l'opération après avoir désactivé le dédoublonnage de données.

Pour économiser davantage encore la bande passante, vous pouvez activer une mémoire cache local pour le dédoublonnage. La mémoire cache locale évite aux requêtes d'être transférées au serveur IBM Spectrum Protect. La valeur par défaut pour ENABLEDEDUPCACHE est NO, de sorte que la mémoire cache n'est pas désynchronisée par rapport au serveur. Si la mémoire cache est désynchronisée avec le serveur, l'application renvoie toutes les données. Si votre application peut renouveler une opération ayant échoué et que vous souhaitez utiliser la mémoire cache locale, définissez l'option ENABLEDEDUPCACHE sur YES dans le fichier dsm.opt (Windows) ou dsm.sys (UNIX).

A la fin d'une opération de restauration, si *toutes* les données ont été restaurées via l'API et que l'objet a été dédoublonné par le client, un condensé de bout en bout est calculé et comparé à la valeur calculée au moment de la sauvegarde. Si ces valeurs ne correspondent pas, une erreur DSM_RC_DIGEST_VALIDATION_ERROR est retournée. Si une application reçoit cette erreur, les données sont corrompues. Cette erreur peut aussi résulter d'une erreur transitoire sur le réseau : vous pouvez donc renouveler l'opération de restauration ou de récupération.

Voici un exemple de commande de session de requête présentant les informations de dédoublonnage de données :

```
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:
```

Voici un exemple de la commande de classe de gestion de requête présentant les informations relatives au dédoublonnage :

```
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
```

Référence associée:

Option Deduplication

Exclusion de fichiers du dédoublonnage de données

Vous pouvez choisir d'exclure certains fichiers de sauvegarde ou d'archivage du dédoublonnage de données.

Pour exclure des fichiers du traitement de dédoublonnage, procédez comme suit.

1. Définissez l'option `exclude.dedup` pour les objets à exclure.

   Ainsi, pour exclure toutes les données de dédoublonnage pour les systèmes UNIX, définissez :

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

2. Pour exclure toutes les données du dédoublonnage pour les systèmes Windows, définissez :

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

Important : Si un objet est envoyé à un pool de dédoublonnage, l'opération de dédoublonnage se produit sur le serveur, même si l'objet est exclu du dédoublonnage de données côté client.

Inclusion de fichiers au dédoublonnage de données

Vous pouvez choisir d'inclure certains fichiers de sauvegarde ou d'archivage dans le dédoublonnage de données.

Pour affiner la liste de fichiers à inclure, l'option `include.dedup` peut être utilisée conjointement avec l'option `exclude.dedup`.

Par défaut, tous les objets éligibles sont inclus pour le dédoublonnage de données.

Voici quelques exemples pour UNIX et Linux :

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

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

Voici quelques exemples pour Windows :

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

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

Dédoublonnage de données côté serveur

Le dédoublonnage de données côté serveur est réalisé par le serveur.

L'administrateur IBM Spectrum Protect peut spécifier l'emplacement du dédoublonnage (client ou serveur) à utiliser avec le paramètre **DEDUP** sur la commande du serveur **REGISTER NODE** ou **UPDATE NODE**.

Dans un pool de stockage activé pour le dédoublonnage (pool de fichiers), une seule instance de l'extension de données est conservée. Les autres instances de la même extension de données sont remplacées par un pointeur vers l'instance conservée.

Pour plus d'informations sur le dédoublonnage de données côté serveur, voir la documentation sur le serveur IBM Spectrum Protect.

Reprise d'application

Lorsque le serveur IBM Spectrum Protect devient indisponible en raison d'une indisponibilité, les applications qui utilisent l'API peuvent automatiquement basculer sur un serveur secondaire pour la récupération des données.

Le serveur IBM Spectrum Protect auquel le client et l'API se connectent pendant des processus de production normaux s'appelle le *serveur principal*. Lorsque le serveur principal est configuré pour la réplication de poste, ce serveur est également appelé *serveur de réplication source*. Les données de poste client sur le serveur de réplication source peuvent être répliquées vers le *serveur de réplication cible*. Ce serveur, également appelé *serveur secondaire*, est le serveur sur lequel le client bascule automatiquement lorsque le serveur principal échoue.

Le client et l'API doivent être configurés pour la reprise en ligne du client automatisée et doivent se connecter à un serveur version 7.1 (ou ultérieure) qui réplique les données de poste client. La configuration de l'API est la même que la configuration du client de sauvegarde-archivage.

Pendant le fonctionnement normal, les informations de connexion du serveur secondaire sont automatiquement envoyées au client du serveur principal pendant le processus de connexion. Les informations du serveur secondaire sont automatiquement sauvegardées dans le fichier d'options client.

Chaque fois que l'application client se connecte au serveur IBM Spectrum Protect, elle tente de contacter le serveur principal. Si le serveur principal est indisponible, l'application bascule automatiquement sur le serveur secondaire en utilisant les informations du serveur secondaire dans le fichier d'options client. En mode de basculement, l'application peut interroger le serveur secondaire et restaurer ou récupérer les données répliquées.

Vous devez sauvegarder l'application au moins une fois sur le serveur principal. L'API peut basculer sur le serveur secondaire pour récupérer des données, uniquement si les données du poste client ont été répliquées du serveur principal vers le serveur secondaire.

Concepts associés:

Configuration et utilisation de la reprise en ligne de client automatisée

Informations sur l'état de la reprise en ligne

L'API fournit des informations d'état que les applications peuvent utiliser pour déterminer l'état de reprise en ligne et l'état des données client répliquées sur le serveur secondaire.

Le statut de réplication indique si la dernière sauvegarde a été répliquée sur le serveur secondaire. Si l'horodatage de la dernière opération de sauvegarde sur l'API correspond à l'horodatage de la sauvegarde sur le serveur secondaire, le statut de réplication est en cours. Si les deux horodatages ne correspondent pas, le statut de réplication n'est pas en cours et les données répliquées peuvent être obsolètes.

Les informations suivantes sur le statut de réplication sont renvoyées sur la réponse **query filespace** sur l'appel de fonction **dsmGetNextQObj** de la structure **qryRespFSData** :

Tableau 14. Informations sur le statut de réplication reportées par l'API

Informations sur le statut	Type	Définition
Début de la dernière réplication	**lastReplStartDate**	La dernière fois que la réplication a été lancée.
Fin de la dernière réplication	**lastReplCmpltDate**	La dernière fois que la réplication a été effectuée, même si un incident s'est produit.
Date de stockage de la dernière sauvegarde (Serveur)	**lastBackOpDateFromServer**	Le dernier horodatage de stockage sauvegardé sur le serveur.
Date de stockage de la dernière sauvegarde (Local)	**lastBackOpDateFromLocal**	Le dernier horodatage de stockage sauvegardé sur le client.

L'état de la reprise en ligne est signalé par le champ **bIsFailOverMode** de la structure **dsmInitExOut_t**.

Voir Annexe B, «Fichiers source des définitions de type d'API», à la page 165 pour la structure et les définitions de type de l'API.

Le code de retour DSM_RC_SIGNON_FAILOVER_MODE indique que le client et l'API ont basculé vers le serveur secondaire et s'exécutent en mode de reprise en ligne.

Exemple de connexion pendant une reprise en ligne

L'exemple suivant illustre une signature sur le serveur pendant une reprise en ligne :

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

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

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

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

Exemple de commande query session

L'exemple suivant illustre la commande **query session** qui affiche les informations du serveur secondaire (de réplication) :

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

Exemple de commande query filespace

L'exemple suivant illustre la commande **query filespace** qui affiche l'état de réplication d'un espace fichier sur le serveur secondaire :

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

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

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

Référence associée:

«dsmGetNextQObj», à la page 115

Exemples de diagramme de flux pour la sauvegarde et l'archivage

L'API est conçue pour les flux de données logiques directs et les transitions claires entre les divers états de l'application client. Ces transitions permettent d'identifier les erreurs dans la logique ainsi que les erreurs de programme au début du cycle de développement, améliorant considérablement la qualité et la fiabilité du système.

Par exemple, la fonction **dsmSendObj** peut être appelée uniquement lorsqu'une transaction est lancée et qu'un appel à **dsmBindMC** a été effectué pour l'objet que vous sauvegardez.

La figure 12 affiche l'organigramme relatif aux opérations de sauvegarde ou d'archivage au cours d'une transaction. La flèche pointant de «In Send Object» vers **dsmEndTxn** indique que vous pouvez faire appel à la fonction **dsmEndTxn** suite à un appel à **dsmSendObj** ou **dsmSendData**. Cette opération peut s'avérer nécessaire si une condition d'erreur a été générée lors de l'envoi d'un objet et que vous souhaitez mettre fin à l'opération. Dans ce cas, vous devez utiliser le vote DSM_VOTE_ABORT. Toutefois, en toute autre circonstance, faites appel à **dsmEndSendObj** avant de mettre fin à la transaction.

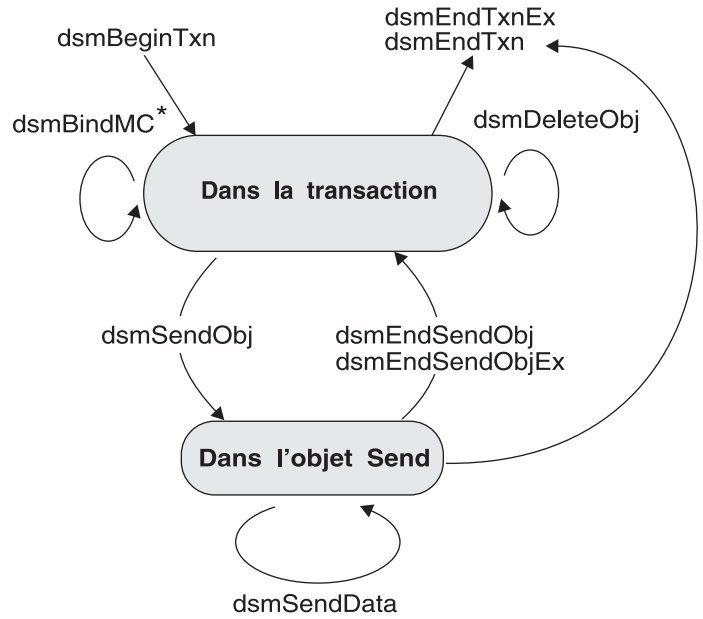


Figure 12. Organigramme relatif aux opérations de sauvegarde et d'archivage

La figure 13, à la page 63 affiche l'organigramme relatif à l'exécution des opérations de sauvegarde et d'archivage au cours d'une transaction.

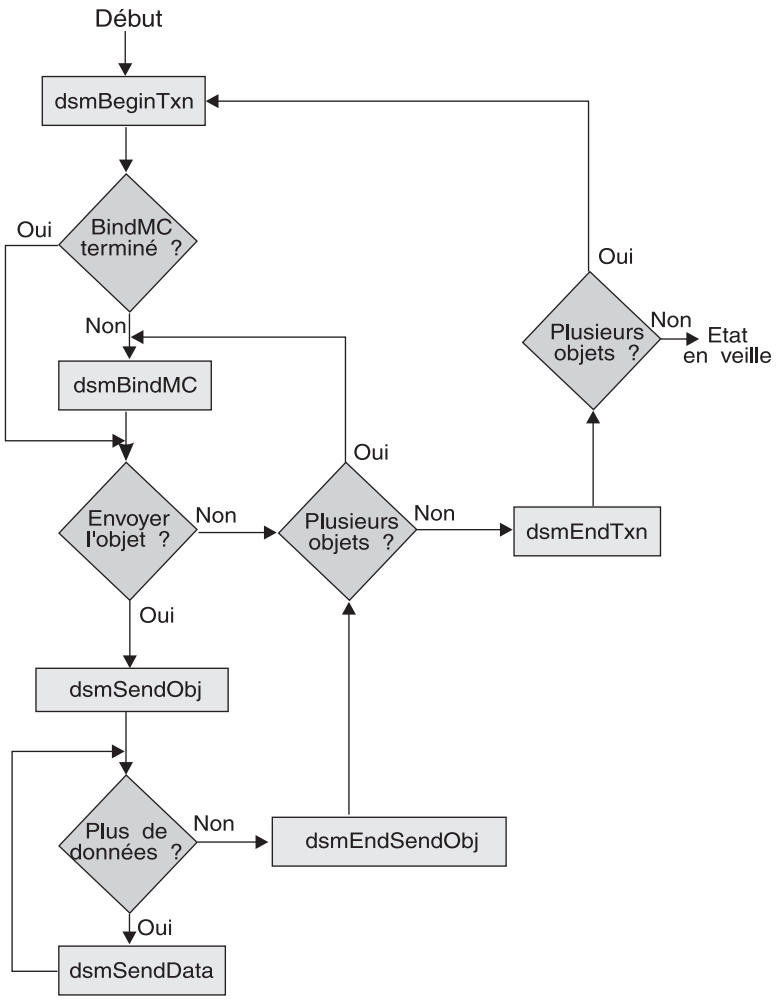


Figure 13. Organigramme relatif aux opérations de sauvegarde et d'archivage

L'élément principal de ces deux diagrammes est la boucle créée entre les appels ci-dessous à l'API effectués au cours d'une transaction :

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

L'appel de fonction **dsmBindMC** est unique car vous pouvez l'utiliser lors d'une transaction ou hors d'une transaction. Vous pouvez également appeler cette fonction pendant une autre transaction, si nécessaire. La seule exigence à respecter pour une utilisation de l'appel de fonction **dsmBindMC** est que l'appel doit être fait avant toute opération de sauvegarde ou d'archivage d'un objet. Si l'objet que vous sauvegardez ou archivez n'est pas associé à une classe de gestion, un code d'erreur est renvoyé par **dsmSendObj**. Si cela se produit, la transaction est interrompue par un appel à **dsmEndTxn** (cette condition d'erreur n'est pas incluse dans l'organigramme).

L'organigramme montre comment une application utilise les transactions à objets multiples. Il montre également où placer les points de décision afin de déterminer

si l'objet envoyé est adapté à la transaction ou si le lancement d'une nouvelle transaction est nécessaire.

Exemple de code pour les fonctions API exécutant l'envoi de données à la mémoire IBM Spectrum Protect

Cet exemple explique comment utiliser les fonctions API relatives à l'envoi de données à la mémoire IBM Spectrum Protect. L'appel de fonction **dsmSendObj** apparaît dans une instruction de transfert afin de permettre l'appel de différents paramètres selon l'opération effectuée (opération de sauvegarde ou de récupération).

La fonction **dsmSendData** est appelée de l'intérieur d'une boucle qui envoie des données de manière répétitive jusqu'à ce qu'un indicateur soit défini pour permettre à l'exécution de programme de mettre fin à la boucle. L'intégralité de l'opération d'envoi est effectuée lors de la transaction.

Le troisième paramètre de l'appel de fonction **dsmSendObj** est une mémoire tampon qui contient la description de l'archive. Comme les objets sauvegardés n'ont aucune description, la valeur de ce paramètre est NULL lors de la sauvegarde d'un objet.

La figure 8, à la page 32 décrit l'utilisation de l'appel de fonction **dsmBindMC**.

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

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

Figure 14. Exemple d'envoi de données à un serveur

Regroupement de fichiers

L'API IBM Spectrum Protect dispose d'un protocole de regroupement de fichiers logique associant plusieurs objets individuels. Vous pouvez faire référence à ces groupes et les gérer en tant que groupe logique sur le serveur. Tous les membres du groupe ainsi que le chef d'un groupe logique doivent appartenir au même poste et au même espace fichier sur le serveur.

Chaque groupe logique est doté d'un chef de groupe. Si ce dernier est supprimé, le groupe sera également supprimé. Vous ne pouvez pas supprimer le membre d'un groupe. L'expiration des membres d'un groupe varie en fonction du chef de groupe. Par exemple, si un membre est signalé pour expiration, il ne peut expirer que si le chef du groupe expire. Cependant, si un membre n'est pas marqué comme devant expirer et le chef du groupe est arrivé à expiration, tous les membres sont expirés.

Les groupes de fichiers contiennent des données de sauvegarde uniquement, et ne peuvent contenir de données d'archivage. Pour les objets archivés, vous pouvez utiliser la zone **Archive Description** afin de faciliter un type de regroupement, si nécessaire pour une application.

L'appel de fonction **dsmGroupHandler** regroupe les opérations. La fonction **dsmGroupHandler** doit être appelée à partir d'une transaction. La plupart des conditions d'erreur relatives au groupe sont détectées suite aux appels à **dsmEndTxn** ou **dsmEndTxnEx**.

La structure out figurant dans un appel à **dsmEndTxnEx** une nouvelle zone, **groupLeaderObjId**. Cette zone contient l'ID objet du chef de groupe si un groupe était ouvert au cours de cette transaction. Vous pouvez créer un groupe sur plusieurs transactions. Un groupe ne peut pas être validé ou enregistré sur le serveur tant qu'une fermeture n'a pas été effectuée. La fonction **dsmGroupHandler** est une interface pouvant accepter cinq opérations différentes. Les opérations sont les suivantes :

- DSM_GROUP_ACTION_OPEN
- DSM_GROUP_ACTION_CLOSE
- DSM_GROUP_ACTION_ADD
- DSM_GROUP_ACTION_ASSIGNTO
- DSM_GROUP_ACTION_REMOVE

Le tableau 15 répertorie les actions d'appel de fonction **dsmGroupHandler** :

Tableau 15. fonctions dsmGroupHanlder

Action	Description
OPEN	L'action OPEN permet de créer un groupe. Le prochain objet envoyé sera le chef de groupe. Le chef de groupe ne peut avoir un contenu. Tous les objets figurant après le premier objet deviennent des membres du groupe. Pour créer un groupe, ouvrez un groupe puis transmettez une chaîne unique pour identifier ce groupe. Cet identificateur unique permet d'ouvrir plusieurs groupes dotés du même nom. Une fois le groupe ouvert, l'objet suivant envoyé est le chef de groupe. Tous les autres objets envoyés sont les membres du groupe.

Tableau 15. fonctions dsmGroupHanlder (suite)

Action	Description
CLOSE	L'action CLOSE permet de valider et de sauvegarder un groupe ouvert. Pour fermer le groupe, transmettez le nom de l'objet ainsi que la chaîne unique utilisée au cours de l'opération d'ouverture. L'application doit vérifier si des groupes ouverts existent, puis les fermer ou les supprimer le cas échéant. Un groupe ne peut pas être validé ou enregistré tant que le groupe n'est pas fermé. L'action CLOSE échoue dans les cas suivants : • Le groupe que vous essayez de fermer porte le même nom qu'un groupe ouvert déjà existant. • Il existe une incompatibilité au niveau de la classe de gestion entre le groupe fermé actuel et le nouveau groupe à fermer portant le même nom. Dans ce cas, procédez comme suit. 1. Interrogez le précédent groupe fermé. 2. Si la classe de gestion du groupe fermé existant est différente de celle associée au groupe ouvert actuel, appelez **dsmUpdateObject** de type DSM_BACKUPD_MC. Cette commande permet de mettre à jour le groupe existant vers la nouvelle classe de gestion. 3. Exécutez l'action CLOSE.
ADD	L'action ADD permet d'ajouter un objet à un groupe. Tous les objets envoyés après l'exécution de l'action ADD sont attribués au groupe.
ASSIGNTO	L'action ASSIGNTO permet au client d'attribuer des objets existant sur le serveur à un groupe homologue défini. Cette transaction génère la relation de groupe PEER. L'action ASSIGNTO est similaire à l'action ADD, à l'exception des éléments suivants : • L'action ADD s'applique aux objets d'une transaction en cours. • L'action ASSIGNTO s'applique à un objet situé sur le serveur.
REMOVE	L'action REMOVE permet de supprimer un membre ou une liste de membres d'un groupe. Un chef de groupe ne peut pas être supprimé d'un groupe. Un membre de groupe doit être retiré du groupe avant d'être supprimé.

Utilisez les types de requêtes suivants pour la prise en charge de groupe :
- **qtBackupGroups**
- **qtOpenGroups**

La requête **qtBackupGroups** interroge les groupes fermés tandis que la requête **qtOpenGroups** interroge les groupes ouverts. La mémoire tampon de la requête des nouveaux types comporte des zones pour les options **groupLeaderObjId** et **objType**. La requête s'exécute différemment, en fonction des valeurs définies pour ces deux zones. Des exemples de requête sont inclus dans le tableau suivant :

Tableau 16. Exemples de requêtes

groupLeaderObjId.hi	groupLeaderObjId.lo	objType	Résultat
0	0	NULL	Renvoie la liste de tous les chefs de groupe
grpLdrObjId.hi	grpLdrObjId.lo	0	Renvoie une liste de tous les membres de groupe affectés au chef de groupe indiqué (**grpLdrObjId**).
grpLdrObjId.hi	grpLdrObjId.lo	objType	Renvoie une liste à l'aide de **BackQryRespEnhanced3**, pour chaque membre de groupe affecté à un chef de groupe défini (**grpLdrObjId**), et correspondant au type de l'objet (**objType**).

La structure de réponse (**qryRespBackupData**) suite à un appel **dsmGetNextQObj** comporte deux zones pour la prise en charge du groupe :

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

Ces zones correspondent à des indicateurs booléens. L'exemple suivant décrit la création du groupe, le processus d'ajout de membres au groupe et la fermeture du groupe afin de le valider sur le serveur IBM Spectrum Protect.

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

Figure 15. Exemple de pseudo-code utilisé pour la création d'un groupe

Pour obtenir un exemple de code, reportez-vous à l'exemple de programme de groupe `dsmgrp.c` se trouvent dans le répertoire `sampsrc` de l'API.

Réception de données du serveur

Les applications client peuvent recevoir des données ou des objets nommés ainsi que leurs données correspondantes depuis l'espace de stockage IBM Spectrum Protect en utilisant les fonctions de restauration et de récupération. La fonction de restauration permet d'accéder aux objets précédemment sauvegardés, tandis que la fonction de récupération permet d'accéder aux objets précédemment archivés.

Restriction : L'API peut uniquement restaurer ou récupérer les objets sauvegardés ou archivés à l'aide des appels d'API.

Les fonctions de restauration et de récupération commencent toutes deux par une opération de requête. La requête renvoie différentes informations, selon si l'objet était sauvegardé ou archivé à l'origine. Par exemple, une requête portant sur des objets sauvegardés renvoie des informations qui indiquent l'état actif ou inactif d'un objet, tandis qu'une requête portant sur des objets archivés renvoie des informations telles que les descriptions de l'objet. Les deux types de requête renvoient des ID objet qui sont utilisés pour identifier de façon unique un objet sur le serveur.

Restauration ou récupération partielle de l'objet

L'application client peut uniquement recevoir une partie de l'objet. Ceci se nomme une restauration partielle d'objet ou une extraction partielle d'objet.

Avertissement : Si une opération de restauration ou de récupération partielle est effectuée sur un objet compressé ou chiffré, vous risquez d'obtenir des résultats non souhaités.

Remarque : Si vous codez votre application de manière à ce qu'elle utilise une restauration ou une récupération partielle d'objet, vous ne pouvez pas compresser les données lors de leur envoi. Pour appliquer cette règle, définissez *ObjAttr.objCompressed* sur *bTrue*.

Pour effectuer une opération de restauration ou de récupération partielle d'objet, associez les deux zones de données suivantes à chaque entrée **GetList** d'objet :

offset Le décalage d'octets dans l'objet à l'origine du premier renvoi de données.

length Le nombre d'octets à renvoyer de l'objet.

Utilisez la fonction DSM_MAX_PARTIAL_GET_OBJ pour indiquer le nombre maximal d'objets pour lesquels une opération de restauration ou de récupération partielle d'objet peut être effectuée par rapport à une liste **dsmBeginGetData** spécifique.

Les zones de données suivantes, utilisées lors d'un appel à **dsmBeginGetData** , déterminent la partie de l'objet à restaurer ou à récupérer :

- Si la valeur du décalage ainsi que celle de la longueur est égale à zéro, l'intégralité de l'objet est restaurée ou récupérée de la mémoire IBM Spectrum Protect.
- Si la valeur du décalage est supérieure à zéro, mais que la longueur est définie sur zéro, l'objet est restauré ou récupéré à partir du point de décalage jusqu'à la fin.
- Si la valeur de la longueur est supérieure à zéro, seule la partie de l'objet commençant à partir du décalage et correspondant à la longueur spécifiée est restaurée ou récupérée.

Restauration ou récupération de données

Une fois la requête effectuée et une session établie avec le serveur IBM Spectrum Protect, vous pouvez exécuter une procédure pour restaurer ou récupérer des données.

Pour restaurer ou récupérer des données, procédez comme suit.

1. Interrogez le serveur IBM Spectrum Protect dans le but d'obtenir des données sauvegardées ou archivées.
2. Déterminez les objets à restaurer ou à récupérer du serveur.
3. Triez les objets du champ **Restore Order**.
4. Faites appel à la fonction **dsmBeginGetData** avec la liste des objets à accéder.
5. Faites un appel à la fonction **dsmGetObj** pour obtenir chaque objet à partir du système. Plusieurs appels à **dsmGetData** peuvent être nécessaires pour chaque objet afin d'obtenir toutes les données d'objet associées. Une fois toutes les données relatives à un objet obtenues, utilisez l'appel de fonction **dsmEndGetObj** .

6. Faites un appel à la fonction **dsmEndGetData** après obtention de toutes les données pour tous les objets ou pour mettre fin à l'opération de réception.

Interrogation du serveur

Avant de lancer toute opération de restauration ou de récupération, interrogez d'abord le serveur IBM Spectrum Protect afin de déterminer la nature des objets retournés par la mémoire.

Pour envoyer la requête, l'application doit fournir la liste de paramètres ainsi que les structures pour l'appel de fonction **dsmBeginQuery**. La structure doit inclure l'espace fichier qui est examiné par la requête et les entrées correspondant à des modèles pour les zones destinées aux noms de haut niveau et de bas niveau. Si lors de l'initialisation de la session la valeur attribuée au nom du propriétaire était NULL, il n'est pas nécessaire de renseigner la zone Propriétaire. Toutefois, si la session a été initialisée sous un nom de propriétaire explicite, seuls les objets associés à ce nom de propriétaire sont renvoyés.

La requête ponctuelle **BackupQuery** fournit une image instantanée du système correspondant à un moment précis. Si vous spécifiez une date valide, vous pouvez interroger tous les fichiers sauvegardés à cette date. Même si une version de sauvegarde active de date ultérieure existe pour un objet, la fonction remplace l'état de l'objet de sorte à renvoyer la version inactive précédente de celui-ci. Pour plus d'informations, consultez l'exemple suivant : pitDate.

Une requête renvoie toutes les informations stockées avec l'objet, en plus des informations de la table suivante.

Tableau 17. Informations de retour sur la requête au serveur

Zone	Description
`copyId`	Les valeurs `copyIdHi` et `copyIdLo` fournissent un nombre de 8 octets qui identifie de manière unique cet objet pour ce poste dans l'espace de stockage IBM Spectrum Protect. Utilisez cet ID pour demander un objet spécifique se trouvant dans l'espace de stockage dans le but d'exécuter une opération de restauration ou de récupération.
`restoreOrderExt`	La valeur `restoreOrderExt` fournit un mécanisme permettant de recevoir des objets du stockage IBM Spectrum Protect de la manière la plus efficace possible. Cette valeur permet de trier les objets à restaurer et donc, de garantir une opération de montage de bandes unique ainsi qu'une lecture intégrale de celles-ci.

Vous devez reporter toutes ou certaines des informations relatives à la requête pour traitement ultérieur. Conservez les zones `copyId` et `restoreOrderExt` car elles sont nécessaires à l'opération de restauration. Vous devez également conserver toute autre information requise pour pouvoir ouvrir un fichier de données ou identifier une destination.

Faites appel à **dsmEndQuery** pour terminer l'opération d'interrogation.

Sélection et tri d'objets par ordre de restauration

Une fois la requête de sauvegarde ou d'archivage réalisée, le client d'application doit déterminer quels objets doivent éventuellement être restaurés ou récupérés.

Vous devez ensuite trier les objets par ordre croissant (du plus petit au plus grand). Cette opération de triage est essentielle pour une performance optimale de l'opération de restauration. Le tri des objets dans les zones **restoreOrderExt** garantit que les données sont lues depuis le serveur dans l'ordre le plus efficace.

Toutes les données sur disque sont restaurées en priorité. Ensuite, la restauration des données figurant sur les classes de support nécessitant des montages de volumes (telles une bande) est effectuée. La zone **restoreOrderExt** permet également de garantir une lecture ordonnée des données se trouvant sur la bande, le traitement commençant par le début de la bande et se poursuivant jusqu'à la fin de celle-ci.

Une opération de triage correctement effectuée à l'aide de la zone **restoreOrderExt** signifie qu'aucun montage de bande en double et rembobinage de bande inutile ne se produira.

Si une valeur autre que zéro est attribuée à la zone **restoreOrderExt**.top, celle-ci est en corrélation avec une seule unité d'accès séquentiel sur le serveur IBM Spectrum Protect. Comme une unité d'accès séquentiel ne peut être utilisée que par une session/un point de montage à la fois, l'application doit s'assurer, en cas d'utilisation de plusieurs sessions, qu'il n'y a pas d'autres opérations de restauration effectuées simultanément et avec une valeur identique de **restoreOrderExt** .top. Dans la cas contraire, la première session est en mesure d'accéder aux objets, mais les autres, sessions patientent jusqu'à ce que le première session aboutisse et que l'unité soit disponible.

L'exemple suivant illustre le tri des objets à 'aide des zones **Restore Order**.

Figure 16. Tri des objets à l'aide des zones restore order

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

}  SortOrder;               /* struct used for sorting */

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

```
   qDataBlkArea.bufferPtf = (char*) &qbDataArea;

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

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

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

      i++;
      qry_item++;

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

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

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

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

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

Démarrage de l'appel à dsmBeginGetData

Une fois les objets à recevoir sélectionnés et triés, soumettez-les à IBM Spectrum Protect pour une opération de restauration ou d'extraction. L'appel à **dsmBeginGetData** lance une opération de restauration et de récupération. Les objets sont renvoyés à l'application client dans l'ordre demandé.

Fournissez des informations à ces deux paramètres lors de ces appels :

mountWait
: Ce paramètre indique au serveur si l'application client attend ou non que les supports déconnectés soient montés afin d'obtenir des données relatives à un objet ou si cet objet doit être ignoré lors de l'opération de restauration ou de récupération.

dsmGetObjListP
: Ce paramètre est une structure de données contenant la zone **objId** qui est une liste de tous les ID objet restaurés ou récupérés. Chaque **objId** est associé à une structure **partialObjData** décrivant si l'**objId** entier ou une section particulière uniquement de l'objet sera récupéré.

  Chaque **objId** est composé de huit octets, donc une requête de restauration ou de récupération peut, à elle seule, contenir des milliers d'objets. Le nombre d'objets pouvant être demandés lors d'un appel est limité à la valeur de DSM_MAX_GET_OBJ ou DSM_MAX_PARTIAL_GET_OBJ.

Réception de chaque objet à restaurer ou à récupérer

Une fois l'appel à la commande **dsmBeginGetData** effectué, suivez une procédure permettant de recevoir chaque objet envoyé par le serveur.

Le code retour DSM_RC_MORE_DATA signifie qu'une mémoire tampon a été renvoyée et que vous devez à nouveau appeler **dsmGetData**. Vérifiez **DataBlk.numBytes** pour connaître le nombre réel d'octets renvoyés.

Une fois toutes les données relatives à un objet obtenues, vous devez faire appel à la fonction **dsmEndGetObj** . Si d'autres objets sont reçus, envoyez à nouveau **dsmGetObj**.

Si vous devez arrêter le processus, par exemple pour ignorer toute donnée restante dans le flux de restauration pour tous les objets qui n'ont pas encore été reçus, appelez la commande **dsmEndGetData** . Ceci provoque l'envoi des données du serveur au client. Toutefois, cette méthode n'est pas très rapide. Si vous souhaitez mettre fin à une opération de restauration, utilisez **dsmTerminate** pour fermer la session.

1. Appelez la commande **dsmGetObj** pour identifier l'objet demandé dans le flux de données et pour obtenir le premier bloc de données associé à l'objet.
2. Effectuez plusieurs appels à la commande **dsmGetData** si nécessaire, pour obtenir les données objet restantes.

Exemples de diagrammes de flux pour la restauration et la récupération

Un diagramme d'état et un organigramme peuvent être utilisés pour visualiser des exemples d'opérations de restauration ou de récupération.

La flèche pointant de «In Get Object» vers **dsmEndGetData** indique que vous pouvez faire appel à la fonction **dsmEndGetData** suite à un appel à **dsmGetObj** ou **dsmGetData**. Cette opération peut s'avérer nécessaire si une condition d'erreur a été générée lors de l'obtention d'un objet de la mémoire IBM Spectrum Protect et que vous souhaitez mettre fin à l'opération. Toutefois, en toute autre circonstance, faites d'abord appel à la fonction **dsmEndGetObj** .

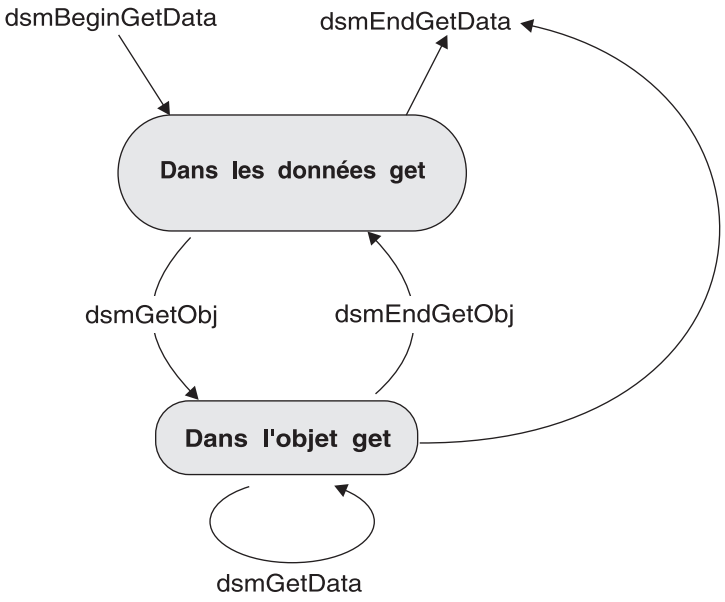


Figure 17. Diagramme d'état relatif aux opérations de restauration et de récupération

La figure 18, à la page 75 affiche l'organigramme relatif aux opérations de restauration et de récupération.

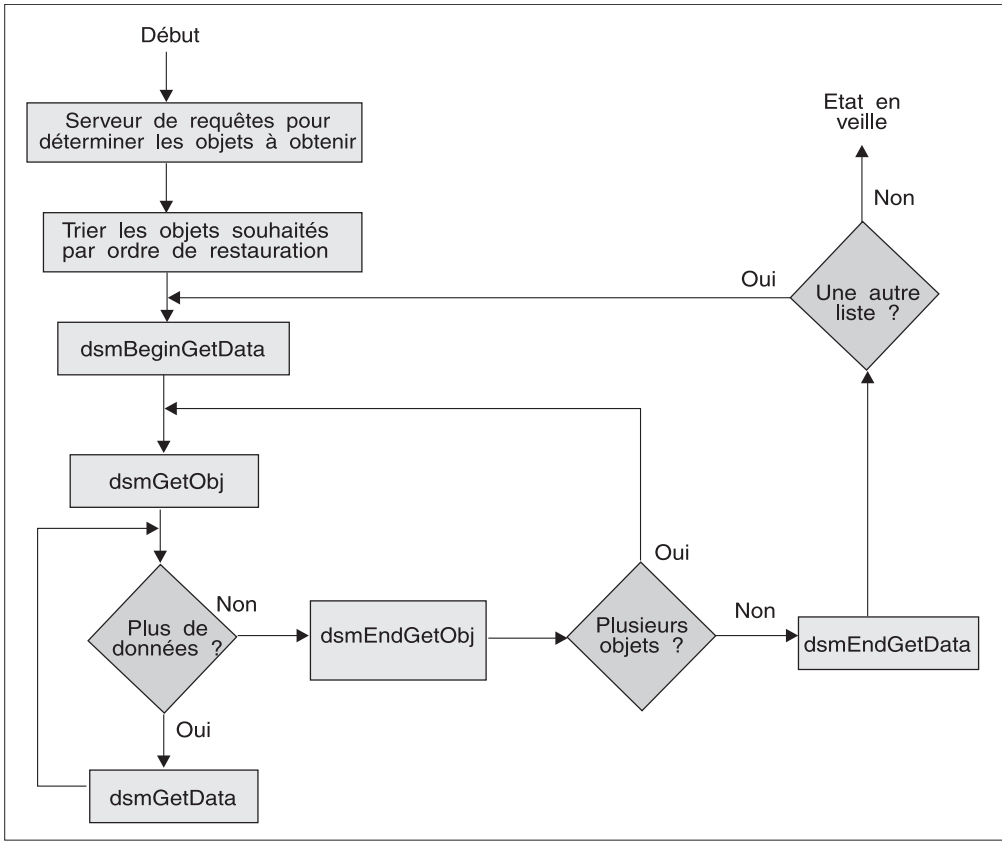


Figure 18. Organigramme relatif aux opérations de restauration et de récupération

Exemple de code de réception de données d'un serveur

Cet exemple décrit l'utilisation des fonction de l'API pour récupérer des données à partir de la mémoire IBM Spectrum Protect.

L'appel de fonction **dsmBeginGetData** apparaît dans une instruction de transfert afin de permettre l'appel de différents paramètres selon l'opération effectuée (opération de restauration ou opération de récupération). La fonction **dsmGetData** est appelée de l'intérieur d'une boucle qui obtient de manière répétitive des données du serveur jusqu'à ce qu'un indicateur soit défini pour permettre à l'exécution de programme de mettre fin à la boucle.

Figure 19. Exemple de récupération de données d'un serveur

```
/* 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("*** Echec dsmBeginGetData  : ");
   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;
```

Mise à jour et suppression des objets sur le serveur

L'appel de fonction **dsmUpdateObj** ou **dsmUpdateObjEx** permet à vos applications API de mettre à jour les objets archivés ou sauvegardés. Utilisez l'un ou l'autre uniquement à l'état de session, en mettant à jour un objet à la fois. Utilisez **dsmUpdateObjEx** pour mettre à jour un objet archivé parmi plusieurs objets archivés portant le même nom.

Pour sélectionner un objet archivé, définissez l'appel de fonction **dsmSendType** sur **stArchive**.

- Avec **dsmUpdateObj**, seul le dernier objet archivé auquel a été attribué le nom est mis à jour.
- Avec **dsmUpdateObjEx**, tout objet archivé peut être mis à jour en spécifiant l'ID objet approprié.

Pour un objet archivé, l'application permet de mettre à jour les zones suivantes :

- Description
- Object information
- Propriétaire

Pour sélectionner un objet sauvegardé, définissez l'appel de fonction **dsmSendType** sur **stBackup**. Seule la version active des objets sauvegardés est mise à jour.

Pour un objet sauvegardé, l'application permet de mettre à jour les zones suivantes :

- Classe de gestion
- Object information
- Propriétaire

Suppression d'objets du serveur

Les applications API peuvent utiliser des appels de fonction pour supprimer des objets archivés ou désactiver des objets sauvegardés. Les autorisations accordées au poste lors de son enregistrement par l'administrateur déterminent si le poste peut ou non supprimer des objets archivés. Les administrateurs peuvent autoriser des postes à effectuer cette opération.

Utilisez l'appel de fonction **dsmDeleteObj** pour supprimer des objets archivés et désactiver des objets sauvegardés. L'utilisation de la valeur **delType** entraîne la suppression de l'objet de sauvegarde du serveur. Cette suppression de l'objet de la base de données du serveur se fait en fonction de l'**objID**. Seul le propriétaire d'un objet a le droit de le supprimer. Toute version (active ou inactive) d'un objet peut être supprimée. Le serveur réconcilie les versions. Si vous supprimez une version active d'un objet, la première version inactive devient active. Si vous supprimez une version inactive d'un objet, toutes les anciennes versions avancent d'un niveau. Le poste doit être enregistré avec l'autorisation **backDel**.

Un objet archivé est marqué comme devant être supprimé de la mémoire lors du prochain cycle d'expiration d'objets effectué par le système. Une fois un objet archivé supprimé du serveur, il ne peut être récupéré.

Lorsque vous désactivez un objet sauvegardé au niveau du serveur, l'objet passe d'un état actif à un état inactif. Ces deux états sont associés à des règles de conservation différentes qui sont déterminées par la classe de gestion affectée.

Comme c'est le cas pour l'appel de fonction **dsmSendObj**, un appel à **dsmDeleteObj** est envoyé lors d'une transaction. Le diagramme d'état de la figure 12, à la page 62 montre comment un appel à la fonction **dsmDeleteObj** est précédé d'un appel à la fonction **dsmBeginTxn** et suivi par un appel à la fonction **dsmEndTxn**.

Consignation des événements

Une application API peut consigner des messages d'évènements dans des emplacements centraux. Elle peut consigner des données dans le serveur IBM Spectrum Protect, l'ordinateur local ou les deux. L'appel de fonction **dsmLogEventEx** est fait pendant une session. Pour afficher des messages consignés sur le serveur, utilisez la commande **query actlog** par l'intermédiaire du client d'administration.

Utilisez l'option client IBM Spectrum Protect, `errorlogretention`, pour supprimer le fichier historique des erreurs client si l'application génère de nombreux messages client dans le journal client `dsmLogType`, `logLocal` ou `logBoth`.

Pour plus d'informations sur les journaux IBM Spectrum Protect, voir la documentation sur le serveur IBM Spectrum Protect.

Récapitulatif du diagramme d'état pour l'API de IBM Spectrum Protect

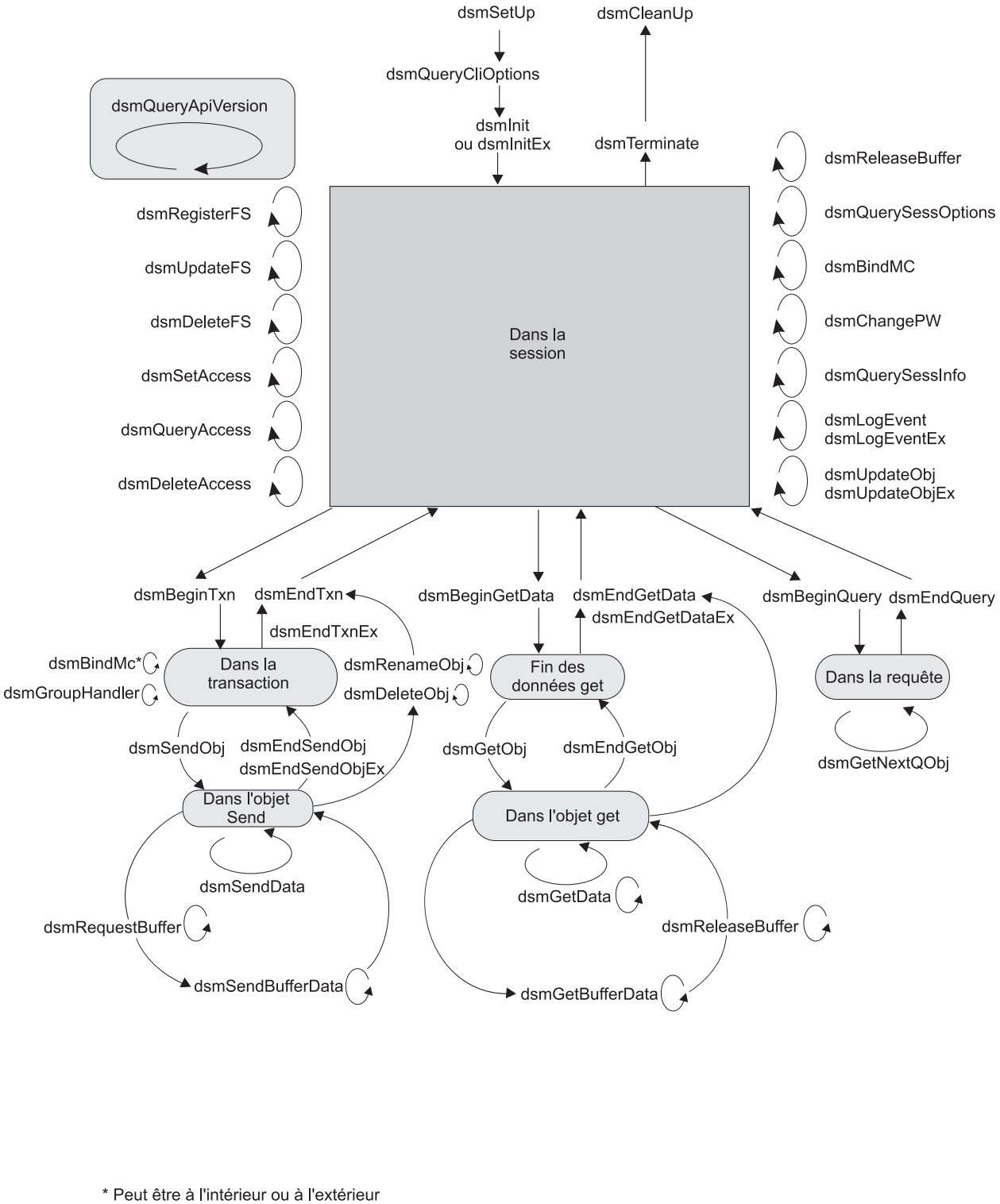
Après avoir consulté toutes les remarques relatives à la création de votre propre application avec l'API de IBM Spectrum Protect, consultez ce diagramme d'état récapitulant l'ensemble d'une application.

La figure 20, à la page 79 contient le diagramme d'état relatif à l'API. Elle contient également tous les diagrammes d'état affichés préalablement ainsi que quelques autres appels non affichés précédemment.

Ce diagramme contient une illustration des consignes suivantes :

- Faites appel à la fonction **dsmQueryApiVersionEx** à tout moment. Aucun état ne lui est associé. Pour un exemple, voir la figure 1, à la page 16.
- Appelez la fonction **dsmQueryCliOptions** avant d'appeler **dsmInitEx** uniquement.
- Utilisez **dsmRegisterFS**, **dsmUpdateFS** et **dsmDeleteFS** pour gérer les espaces fichier. Ces appels sont effectués lors d'une session à l'état inactif. Utilisez l'appel à **dsmBeginQuery** pour interroger les espaces fichier. Pour des informations supplémentaires concernant les appels relatifs aux espaces fichier, voir «Gestion des espaces fichier», à la page 27.
- Faites appel à la fonction **dsmBindMC** lors d'une session à l'état inactif ou à l'état de transaction relatif à un envoi d'objet. Voir l'exemple figurant dans la figure 8, à la page 32.
- Faites appel à la fonction **dsmChangePW** lors d'une session à l'état inactif.

  Remarque : Si l'appel de fonction **dsmInitEx** renvoie un code retour indiquant un mot de passe expiré, faites appel à la fonction **dsmChangePW** avant de démarrer une session valide. Voir la figure 4, à la page 22 pour un exemple utilisant l'appel de fonction **dsmChangePW**.
- Si un appel renvoie une erreur, l'état demeure inchangé. Par exemple, si **dsmGetObj** renvoie une erreur, l'état demeure In Get Data et tout appel de la fonction **dsmEndGetObj** constitue une erreur de séquence d'appel.



* Peut être à l'intérieur ou à l'extérieur

Figure 20. Diagramme d'état récapitulatif pour l'API

Chapitre 4. Comprendre l'interopérabilité

L'API dispose de deux type d'interopérabilité : entre le client de sauvegarde-archivage et les applications de l'API et entre les différents systèmes d'exploitation.

Interopérabilité du client de sauvegarde-archivage

La ligne de commande de sauvegarde-archivage peut accéder aux objets de l'API de façon à fournir une interopérabilité limitée. Les objets de l'API ne peuvent être affichés et utilisés qu'à partir de la ligne de commande de sauvegarde-archivage, mais ne le sont pas à partir de n'importe quelle interface graphique. Le client de ligne de commande de sauvegarde-archivage ne peut restaurer que le contenu du fichier, ainsi vous ne devez l'utiliser que pour un type d'opération de récupération.

Les actions de ligne de commande suivantes sont disponibles :
- Delete archive
- Delete filespace
- Query
- Restore
- Retrieve
- Set access

Les informations relatives au chemin d'accès renvoient aux répertoires actuels des objets du client de sauvegarde-archivage. Au contraire, les informations relatives au chemin d'accès à l'objet de l'API risquent de ne pas être liées aux répertoires existants : le chemin d'accès risque d'être complètement faux. L'interopérabilité ne modifie pas cet aspect de ces types d'objets. Pour utiliser cette fonction convenablement, respectez les restrictions et les conventions.

Remarques :

1. Il n'existe pas d'interopérabilité entre le client de sauvegarde-archivage et les objets de l'API stockés sur un serveur de protection de la conservation de données.
2. Vous ne pouvez pas utiliser les interfaces graphiques du client de sauvegarde-archivage pour accéder aux fichiers stockés à l'aide du client de l'API. Vous ne pouvez utiliser la ligne de commande que pour accéder à ces fichiers.

Dénomination des objets de l'API

Etablissez une convention d'attribution de nom cohérente pour les noms des objets de l'API. La convention d'attribution de nom doit traiter le nom d'espace fichier, le qualificatif de haut niveau et le qualificatif de bas niveau. Le nom de l'espace fichier et les qualificatifs de haut niveau peuvent renvoyer aux noms actuels des répertoires. Chaque nom d'objet peut se composer de plusieurs noms de répertoires applicables au qualificatif de bas niveau.

Il est recommandé d'utiliser le nom de l'objet qui n'est pas doté d'un préfixe se rapportant aux informations du répertoire comme qualificatif de bas niveau. Pour plus d'informations, voir «Noms et ID d'objet», à la page 24.

Les noms des espaces fichiers doivent être complets lorsqu'ils sont référencés par l'API ou par la ligne de commande de sauvegarde-archivage. Par exemple, dan un système d'exploitation UNIX ou Linux, enregistrez les espaces fichier suivants :

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

Lorsque vous faite référence à `/a`, seuls les objets associés à l'espace fichier `/a` sont affichés. Pour afficher les objets associés à `/a/b`, vous devez spécifier `/a/b` comme nom d'espace fichier.

Une fois que vous avez enregistré les deux espaces fichier, si vous sauvegardez l'objet `b` dans l'espace fichier `/a`, une requête relative à `/a/b` affiche toujours des objets associés à l'espace fichier `/a/b` uniquement.

L'exception à cette restriction se produit dans les références des espaces fichiers lorsque vous tentez d'interroger ou de supprimer des espaces fichiers en utilisant l'API. Dans les deux cas, il n'est pas nécessaire d'indiquer des noms complets pour les espaces fichier si vous utilisez un caractère générique. Par exemple, `/a*` se réfère à `/a` et `/a/b`.

Conseil : Si vous accordez de l'importance à l'interopérabilité, évitez d'utiliser des noms d'espaces fichiers qui se chevauchent.

Sur les systèmes Windowsplacez les noms des espaces fichier entre parenthèses { } pour les objets de l'API lorsque vous y accédez à partir de l'interface de ligne de commande de sauvegarde-archivage. Les systèmes d'exploitation Windows utilisent automatiquement des majuscules pour les noms des espaces fichier lorsque vous les enregistrez ou que vous les référencez. Toutefois, cette fonction automatique ne s'applique pas au reste de la spécification de noms d'objets. Si vous désirez une interopérabilité complète, utilisez des majuscules pour les qualificatifs de haut et de bas niveaux dans l'application lors de la sauvegarde des objets de l'API. Si votre application ne met pas en majuscules les qualificatifs de haut niveau (noms de répertoires) et les qualificatifs de bas niveau (noms de fichiers) avant d'envoyer les objets au serveur, vous ne pourrez pas accéder aux objets directement par leur nom via le client de sauvegarde-archivage.

Par exemple, si un objet est stocké sur le serveur en tant que `{"FileSpacename"}\TEST\MYDIRNAME\file.txt`, vous ne pouvez pas directement restaurer ou interroger l'objet `file.txt` car votre application n'a pas mis le nom du fichier en majuscules avant qu'il soit copié vers le serveur. Le seul moyen de manipuler ces objets est d'utiliser des caractères génériques. Par exemple, pour interroger `\TEST\MYDIRNAME\file.txt`, un utilisateur du client de sauvegarde-archivage doit utiliser des caractères génériques pour toutes les parties du nom de l'objet qui n'ont pas été mises en majuscules avant d'être envoyées au serveur. La commande suivante doit être utilisée pour interroger ce fichier `file.txt` :

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

Si un autre des autres qualificatifs a également été sauvegardé en minuscules, ces qualificatifs doivent également être interrogés en utilisant des caractères génériques. Par exemple, pour interroger un objet stocké en tant que `{"FileSpaceName"}\TEST\mydirname\file.txt`, utilisez la commande suivante :

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

Les exemples suivants illustrent ces concepts. Dans les environnements Windows et UNIX ou Linux, vous n'avez pas besoin de spécifier un qualificatif de bas ou de haut niveau complet. Cependant, si vous ne le faites pas, vous devez utiliser le caractère générique.

Plateforme	Exemple
Windows	Pour interroger tous les fichiers sauvegardés dans l'espace fichier `MYFS`, entrez la chaîne suivante : `dsmc q ba "{MYFS}\*\*"` Vous devez utiliser au moins un astérisque (*) pour chaque qualificatif de haut et de bas niveau.
UNIX ou Linux	Pour interroger tous les fichiers sauvegardés dans l'espace fichier `/A`, entrez la chaîne suivante : `dsmc q ba "/A/*/*"` Vous devez utiliser au moins un astérisque (*) pour chaque qualificatif de haut et de bas niveau.

Commandes du client de sauvegarde-archivage pouvant être utilisées avec l'API

Vous pouvez utiliser un sous-ensemble de commandes du client de sauvegarde-archivage dans une application. Par exemple, vous pouvez afficher et gérer des objets que d'autres utilisateurs possèdent sur le même poste ou sur un poste différent.

Pour afficher et gérer des objets que d'autres utilisateurs possèdent sur le même poste ou sur un poste différent, procédez comme suit :

1. Autorisez l'accès à laide de la commande **set access**.
2. Indiquez le propriétaire et le poste. Utilisez les options *fromowner* et *fromnode* de la ligne de commande de sauvegarde-archivage pour indiquer le propriétaire et le poste. Par exemple :

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

Le tableau 18 décrit les commandes que vous pouvez utiliser avec les objets de l'API.

Tableau 18. Commandes du client de sauvegarde-archivage pouvant être utilisées avec des objets de l'API

Commande	Description
Delete Archive	Les fichiers archivés de l'utilisateur actuel peuvent être supprimés. Les paramètres de la commande set access n'ont aucune incidence sur cette commande.
Delete Filespace	La commande **delete filespace** affecte les objets de l'API.

Tableau 18. Commandes du client de sauvegarde-archivage pouvant être utilisées avec des objets de l'API (suite)

Commande	Description
Query	A partir de la ligne de commande de sauvegarde-archivage, vous pouvez interroger les objets de l'API sauvegardés et archivés ainsi que les objets que d'autres utilisateurs possèdent ou qui existent sur d'autres postes. Pour plus d'informations sur les requêtes adressées aux objets de l'API, voir «Dénomination des objets de l'API», à la page 81. Utilisez l'option *–fromowner* existante pour interroger des objets appartenant à un utilisateur différent à qui les droit d'accès set access ont été accordés. Utilisez l'option *–fromnode* existante pour interroger des objets qui existent sur un autre poste pour lequel les droits d'accès set access ont été accordés. Pour plus d'informations, voir «**dsmInitEx**», à la page 124.
Restauration **Retrieve**	**Remarque :** Utilisez ces commandes uniquement pour des situations d'exception. Les objets de l'API chiffrés à l'aide de la clé gérée par l'application peuvent être restaurés ou récupérés si la clé de chiffrement est connue ou enregistrée dans le registre ou fichier de mots de passe. Les objets de l'API chiffrés à l'aide du chiffrement transparent ne peuvent être restaurés ou récupérés à l'aide du client de sauvegarde-archivage. Ces commandes renvoient des données en tant que fichiers bits créés à l'aide des attributs de fichiers par défaut. Vous pouvez restaurer ou extraire des objets de l'API appartenant à d'autres utilisateurs ou provenant d'un poste différent. La commande set access détermine les objets qualifiés.
Set Access	La commande **set access** permet aux utilisateurs de gérer les objets de l'API appartenant à un autre utilisateur ou provenant d'un poste différent.

L'interopérabilité du système d'exploitation

L'API IBM Spectrum Protect prend en charge l'interopérabilité multiplateforme. Les applications sous un système UNIX ou Linux peuvent s'exécuter sur les espaces de fichiers et les objets sauvegardés à partir d'un système Windows. De même, un système Windows peut s'exécuter sur les espaces fichier et les objets qui sont sauvegardés à partir d'un système UNIX ou Linux.

Par défaut, les noms d'objets d'un système UNIX sont compatibles avec les noms d'objets provenant d'autres systèmes UNIX. Par défaut, les noms d'objets des systèmes Windows ne sont pas compatibles avec les noms d'objets provenant des systèmes UNIX. Plusieurs paramètres contrôlent la désignation d'objets dans les espaces fichier IBM Spectrum Protect. Si vous configurez une application de manière appropriée, les noms des objets peuvent être utilisés par des applications qui s'exécutent à la fois sur des systèmes Windows et des systèmes UNIX. Utilisez les mêmes paramètres pour sauvegarder et restaurer des objets.

Restriction : Une application Windows qui utilise Unicode crée un espace fichier qui n'est pas compatible avec des applications qui s'exécutent sur des systèmes UNIX.

Pour obtenir l'interopérabilité, effectuez les tâches de configuration suivantes :

1. Etablissez une convention d'attribution de nom cohérente. Sélectionnez un caractère pour le délimiteur `dir`, tel qu'une barre oblique (/) ou une barre oblique inversée (\). Placez le caractère délimiteur de répertoire devant le nom d'espace fichier, le qualificatif de haut niveau et le qualificatif de bas niveau.

2. Lorsque vous appelez **dsmInitEx**, indiquez dans la zone **dirDelimiter** le caractère délimiteur de répertoire que vous avez sélectionné et affectez au paramètre **bCrossPlatform** la valeur **bTrue**.
3. Affectez à l'indicateur **useUnicode** la valeur **bFalse** lorsque vous utilisez l'interface IBM Spectrum Protect. Les noms de fichier Unicode ne sont pas compatibles avec les noms de fichier non Unicode.

Sauvegarde de postes multiples avec la prise en charge du proxy sur le poste client

Les sauvegardes de plusieurs postes partageant le stockage peuvent être regroupées en un nom de poste cible commun sur le serveur IBM Spectrum Protect. Cette méthode est utile lorsque ce n'est pas le même système qui effectue qui effectue la sauvegarde (en cas de changement de cluster). Vous pouvez également utiliser l'option `asnodename` pour restaurer les données à partir d'un autre système que celui qui a effectué la sauvegarde.

Utilisez l'option `asnodename` sur la chaîne d'options **dsmInitEx** pour sauvegarder, archiver, restaurer, extraire, interroger ou supprimer des données sous le nom de poste cible sur le serveur IBM Spectrum Protect. Vous pouvez également indiquer l'option `asnodename` dans le fichier `dsm.opt` ou `dsm.sys`.

Restriction : N'utilisez pas de postes cibles comme postes classiques, notamment lorsque vous chiffrez vos fichiers avant de les sauvegarder sur le serveur.

Pour activer cette option, procédez comme suit.

1. Installez le client de l'API sur tous les postes dans un environnement de données partagées.
2. Si ce n'est pas déjà fait, enregistrez chaque poste sur le serveur IBM Spectrum Protect. Dans votre environnement de données partagées, enregistrez le nom de poste "cible" commun que vous souhaitez partager entre chaque poste agent.
3. Enregistrez chaque poste agent de l'environnement de données partagées sur le serveur. Le nom de poste agent est utilisé à des fins d'authentification. Les données ne sont pas stockées avec le nom de poste agent lorsque l'option `asnodename` est utilisée.
4. Demandez à votre administrateur d'attribuer des droits de proxy à tous les postes de l'environnement partagé pour accéder au nom du poste cible sur le serveur IBM Spectrum Protect, en utilisant la commande **grant proxynode** .
5. Utilisez la commande du client d'administration **query proxynode** pour afficher les postes client qui se sont vus accorder le droit d'effectuer des opérations client à la place d'un autre poste. Ce droit est accordé par la commande **grant proxynode**. Vous pouvez également utiliser la commande **dsmQuery** avec une requête de type **qtProxyNodeAuth** pour afficher les postes sur lesquels ce poste peut agir en tant que proxy.
6. Si l'application utilise le chiffrement de données de l'utilisateur, et non `TSMENCRKEY`, vérifiez que tous les postes utilisent la même clé de chiffrement key. Vous devez utiliser la même clé de chiffrement pour tous les fichiers sauvegardés dans l'environnement partagé.

Tâches associées:

Sauvegarde de données avec la prise en charge du proxy sur le poste client (systèmes UNIX et Linux)

Sauvegarde de données avec la prise en charge du proxy sur le poste client (systèmes Windows)

Chapitre 5. Utilisation de l'API à l'aide de Unicode

L'API IBM Spectrum Protect prend en charge Unicode UCS2, une page de codes double octets de longueur fixe, dotée de points de code pour les pages de codes connues, telles le japonais, le chinois ou l'allemand. Elle prend en charge un maximum de 65,535 points de code uniques.

Restriction : Cette fonction n'est disponible que sous Windows.

Avec Unicode, votre application a la possibilité de sauvegarder et de restaurer les noms de fichier dans n'importe quel jeu de caractères contenu sur une même machine. Par exemple, un ordinateur configuré en anglais vous permet de sauvegarder et de restaurer les noms de fichier dans d'autres pages de codes de langues.

Quand utiliser Unicode

Vous pouvez simplifier une application multilingue en écrivant une application Unicode et en exploitant l'interface Unicode de IBM Spectrum Protect.

Utilisez l'interface Unicode de IBM Spectrum Protect si l'une des conditions suivantes est remplie :

- Si votre application est déjà compilée pour fonctionner avec le format Unicode et qu'elle effectue des conversions vers un jeu de caractères multi-octets (mbcs) avant d'appeler l'API IBM Spectrum Protect.
- Si vous effectuez une opération d'écriture dans une nouvelle application et souhaitez lui permettre de prendre en charge Unicode.
- Si votre application utilise une chaîne provenant d'un système d'exploitation ou de toute autre application utilisant Unicode.

Inutile de recompiler votre application si vous n'avez pas besoin d'utiliser Unicode.

L'interface dsm est toujours prise en charge par l'API. Les exemples de programmes callmtu1.c et callmtu2.c sont inclus dans le SDK de l'API, illustrant l'utilisation de l'API Unicode. Utilisez **makemtu** pour compiler ces programmes.

Configuration d'Unicode

Pour configurer et utiliser Unicode vous devez effectuer une procédure particulière afin que l'API enregistre un espace fichier Unicode sur le serveur et que tous les noms de fichier contenus dans cet espace fichier deviennent des chaînes Unicode.

Restriction : Les noms de fichiers Unicode et non Unicode ne peuvent pas être stockés dans le même espace fichier.

1. Compilez le code à l'aide de l'indicateur `-DUNICODE`.
2. Les chaînes contenues dans votre application doivent être de type wchar.
3. Pour appeler l'API, suivez les structures du fichier tsmapitd.h file et les définitions de fonction du fichier tsmapifp.h.
4. Définissez l'indicateur *useUnicode* sur *bTrue* pour l'appel de fonction **tsmInitEx**. Tout nouvel espace fichier est enregistré en tant qu'espace fichier Unicode.

Lorsque l'API envoie des données vers des espaces fichier non Unicode préalablement enregistrés, les noms de fichiers sont toujours envoyés en tant que non Unicode. Remplacez le nom des anciens espaces fichier présents sur le serveur par fsname_old et commencez un nouvel espace fichier Unicode pour les nouvelles données. L'API restaure les données non Unicode à partir des anciens espaces fichier. Utilisez les données de la zone **bIsUnicode** de la structure **tsmQryRespFSData** qui est renvoyée sur un espace fichier relatif aux requêtes pour déterminer si un espace fichier est au format Unicode.

Chaque appel de fonction **dsmXXX** a un appel de fonction **tsmXXX** correspondant. Les structures utilisées différencient les deux appels de fonction. Toutes les structures d'appel de fonction **tsmXXX** comportent des valeurs de chaîne de type dsChar_t quand elles sont compilées avec l'indicateur UNICODE. Le type dsChar_r est associé à wchar. Il n'y a aucune autre différence entre ces interfaces.

Restriction : Utilisez l'une ou l'autre des interfaces. Ne combinez pas les interfaces d'appel de fonction **dsmXXX** et **tsmXXX**. Prenez soin d'utiliser les structures IBM Spectrum Protect et les définitions de version IBM Spectrum Protect.

Certaines constantes étant toujours définies dans le fichier dsmapitd.h, les fichiers dsmapitd.h et tsmapitd.h vous sont nécessaires au cours de la compilation.

Vous pouvez utiliser l'interface IBM Spectrum Protect sur d'autres systèmes d'exploitation, tels qu'UNIX ou Linux mais, sur ces systèmes d'exploitation, le type dsChar_t est mappé à char, car Unicode n'est pris en charge que sur les systèmes d'exploitation Windows. Avec l'interface de IBM Spectrum Protect, vous pouvez écrire une seule variation de l'application et effectuer la compilation sur plusieurs systèmes d'exploitation. Lors d'une opération d'écriture dans une nouvelle application, utilisez l'interface de IBM Spectrum Protect.

Lors d'une mise à niveau d'une application existante :
1. Convertissez les structures d'appel de fonction **dsmXXX** ainsi que les appels vers l'interface IBM Spectrum Protect.
2. Migrez les espaces fichier existants.
3. Sauvegardez les nouveaux espaces fichier en définissant l'indicateur *useUnicode* sur *true*.

Remarque : Après avoir utilisé un client activé pour Unicode pour accéder à un poste, vous ne pouvez pas vous connecter à ce même nom de poste en utilisant une version antérieure de l'API ou une API issue d'un autre système d'exploitation. Si votre application fait usage de capacités multiplateformes, n'utilisez pas l'indicateur `Unicode`. Il n'y a pas de prise en charge multiplateforme entre les systèmes d'exploitations Unicode et non Unicode.

Une fois l'indicateur *useUnicode* activé, toutes les structures de chaîne sont considérées comme des chaînes Unicode. Sur le serveur, les zones répertoriées ci-dessous sont les seules à conserver le format Unicode :
- Nom d'espace fichier
- Niveau supérieur
- Niveau inférieur
- Description d'archive.

Les données des zones restantes sont converties en jeux de caractères multi-octets dans la page de codes locale avant d'être envoyées sur le serveur. Les données des zones relatives au nom de poste sont des chaînes wchar. Ces données doivent être valides dans l'environnement local en cours. Par exemple, avec un ordinateur

configuré en japonais, vous avez la possibilité d'effectuer des sauvegardes de fichiers en utilisant des noms chinois, mais le nom de poste doit être une chaîne valide en japonais. Le fichier d'options est conservé dans la page de codes en cours. Si vous devez créer une liste inclusive-exclusive Unicode, utilisez l'option *inclexcl* doté du nom de fichier et créez des fichiers avec des modèles Unicode inclus.

Référence associée:

Option inclexcl

Chapitre 6. Appels de fonction API

Le tableau 19 contient la liste des appels de fonction de l'API classés par ordre alphabétique, accompagnés d'une brève description et de l'emplacement à consulter pour plus d'informations sur les appels de fonction : .

Elément	Description
Objectif	Décrit l'appel de fonction.
Syntaxe	Contient le code C réel de l'appel de fonction. Ce code est copié à partir de la version UNIX ou Linux du fichier d'en-tête dsmapifp.h. Voir Annexe C, «Fichier source des définitions de type d'API», à la page 207. Ce fichier se présente légèrement différemment sous d'autres systèmes d'exploitation. Les programmeurs d'application pour les autres systèmes d'application doivent vérifier leur version du fichier d'en-tête, dsmapifp.h, pour connaître la syntaxe exacte des définitions d'API.
Paramètres	Décrit chacun des paramètres de l'appel de fonction et l'identifie comme paramètre d'entrée (I) ou de sortie (O), selon la façon dont il est utilisé. Certains paramètres sont désignés à la fois comme des paramètres d'entrée et de sortie (I/O). Les types de données référencés dans cette section sont définis dans le fichier d'en-tête dsmapitd.h. Voir Annexe B, «Fichiers source des définitions de type d'API», à la page 165.
Codes retour	Contient la liste des codes retour spécifiques à l'appel de fonction. Les erreurs système générales, telles que les erreurs de communication, les problèmes de serveur ou les erreurs utilisateur susceptibles d'apparaître dans un appel ne sont pas répertoriés. Les codes retour sont définis dans le fichier d'en-tête dsmrc.h. Voir Annexe A, «Fichier source des codes retour de l'API : `dsmrc.h`», à la page 155.

Tableau 19. Appels de fonction API

Appel de fonction et emplacement	Description
«dsmBeginGetData », à la page 94	Démarre une opération de restauration ou d'extraction sur une liste d'objets en mémoire.
«dsmBeginQuery», à la page 95	Lance une demande de requête d'informations vers IBM Spectrum Protect for information.
«dsmBeginTxn», à la page 100	Démarre une ou plusieurs transactions qui commencent une action complète. Soit toutes les actions aboutissent, soit aucune.
«dsmBindMC», à la page 101	Associe ou lie une classe de gestion à l'objet qui est transmis.
«dsmChangePW», à la page 102	Modifie un mot de passe IBM Spectrum Protect.
«dsmCleanUp», à la page 103	Cet appel est utilisé si **`dsmSetUp`** a été appelé.
«dsmDeleteAccess», à la page 104	Supprime les règles d'autorisation en cours pour les versions de sauvegarde et les copies d'archivage de vos objets.
«dsmDeleteFS», à la page 104	Supprime un espace fichier de la mémoire.
«dsmDeleteObj», à la page 105	Désactive les objets de sauvegarde et supprime les objets archivés de la mémoire.

Tableau 19. Appels de fonction API (suite)

Appel de fonction et emplacement	Description
«dsmEndGetData », à la page 107	Termine une session **dsmBeginGetData** qui utilise des objets de la mémoire.
«dsmEndGetDataEx», à la page 107	Fournit le total des octets hors réseau local qui ont été envoyés.
«dsmEndGetObj », à la page 108	Termine une session **dsmGetObj** qui obtient des données pour un objet spécifié.
«dsmEndQuery», à la page 108	Signifie la fin d'une action **dsmBeginQuery**.
«dsmEndSendObj», à la page 109	Indique la fin des données envoyées dans la mémoire.
«dsmEndSendObjEx», à la page 109	Fournit des informations sur la compression et le nombre d'octets envoyés.
«dsmEndTxn», à la page 110	Arrête une transaction IBM Spectrum Protect.
«dsmEndTxnEx», à la page 111	Fournit les informations d'ID objet du chef de groupe à utiliser avec l'appel **dsmGroupHandlerfunction**.
«dsmGetData», à la page 113	Obtient un flot d'octets de données à partir de IBM Spectrum Protect et le place dans la mémoire tampon de l'appelant.
«dsmGetBufferData», à la page 114	Obtient une mémoire tampon de données attribuée par IBM Spectrum Protect depuis le serveur IBM Spectrum Protect
«dsmGetNextQObj», à la page 115	Obtient la réponse à la requête suivante à partir d'un appel **dsmBeginQuery** précédent et la place dans la mémoire tampon de l'appelant.
«dsmGetObj», à la page 118	Obtient la donnée objet demandée à partir du flux de données et la place dans la mémoire tampon de l'appelant.
«dsmGroupHandler», à la page 119	Exécute une action sur un groupe de fichiers logiques en fonction de l'entrée fournie.
«dsmInit», à la page 120	Démarre une session API et connecte le client à la mémoire.
«**dsmInitEx**», à la page 124	Démarre une session API avec les paramètres supplémentaires qui permettent une vérification étendue.
«dsmLogEvent», à la page 128	Consigne un message utilisateur dans le fichier journal du serveur, dans le journal d'erreurs local ou dans les deux.
«dsmLogEventEx», à la page 129	Consigne un message utilisateur dans le fichier journal du serveur, dans le journal d'erreurs local ou dans les deux.
«dsmQueryAccess», à la page 130	Interroge le serveur pour connaître toutes les règles d'autorisation d'accès pour les versions de sauvegarde ou pour les copies archivées de vos objets.
«dsmQueryApiVersion», à la page 131	Exécute une requête concernant la version de bibliothèque d'API à laquelle accède le client d'application.
«dsmQueryApiVersionEx», à la page 131	Exécute une requête concernant la version de bibliothèque d'API à laquelle accède le client d'application.
«dsmQueryCliOptions», à la page 132	Interroge les valeurs des options importantes dans les fichiers d'options de l'utilisateur.
«dsmQuerySessInfo», à la page 133	Lance une demande de requête à IBM Spectrum Protect pour obtenir des informations relatives au fonctionnement de la session indiquée dans **dsmHandle**.

Tableau 19. Appels de fonction API (suite)

Appel de fonction et emplacement	Description
«dsmQuerySessOptions», à la page 134	Interroge les valeurs des options importantes qui sont valides dans la session indiquée dans **dsmHandle**.
«dsmRCMsg», à la page 135	Obtient le texte du message associé à un code retour d'API.
«dsmRegisterFS», à la page 136	Enregistre un nouvel espace fichier auprès du serveur.
«dsmReleaseBuffer», à la page 137	Renvoie une mémoire tampon attribuée par IBM Spectrum Protect.
«dsmRenameObj», à la page 137	Renomme le nom d'objet de haut niveau ou de bas niveau.
«dsmRequestBuffer», à la page 139	Obtient une mémoire tampon attribuée par IBM Spectrum Protect pour l'élimination de la copie de la mémoire tampon.
«dsmRetentionEvent», à la page 140	Envoie une liste d'ID objet au serveur ainsi qu'une opération relative à un événement de conservation à effectuer sur ces objets.
«dsmSendBufferData», à la page 141	Envoie des données à partir d'une mémoire tampon attribuée par IBM Spectrum Protect.
«dsmSendData», à la page 142	Envoie un flot d'octets de données à IBM Spectrum Protect par l'intermédiaire d'une mémoire tampon.
«dsmSendObj», à la page 143	Démarre une requête pour envoyer un objet unique dans la mémoire.
«dsmSetAccess», à la page 147	Accorde à d'autres utilisateurs, ou postes, l'accès aux versions de sauvegarde ou aux copies archivées de vos objets, l'accès à tous vos objets ou à un ensemble sélectif.
«dsmSetUp», à la page 148	Ecrase les valeurs des variables d'environnement.
«dsmTerminate», à la page 149	Termine une session avec le serveur et nettoie l'environnement IBM Spectrum Protect.
«dsmUpdateFS», à la page 150	Met à jour un espace fichier en mémoire.
«dsmUpdateObj», à la page 151	Met à jour l'information objInfo qui est associée à un objet de sauvegarde actif déjà présent sur le serveur ou met à jour des objets archivés.
«dsmUpdateObjEx», à la page 152	Met à jour l'information objInfo associée à un objet d'archivage spécifique même lorsqu'il y existe plusieurs objets de même nom ou met à jour les objets de sauvegarde actifs.

Référence associée:

Codes retour d'API

dsmBeginGetData

L'appel de fonction **dsmBeginGetData** démarre une opération de restauration ou d'extraction sur une liste d'objets dans la mémoire. Cette liste d'objets est contenue dans la structure **dsmGetList**. L'application crée cette liste avec les valeurs provenant de la requête qui a précédé un appel à **dsmBeginGetData** .

L'appelant doit d'abord utiliser les zones d'ordre de restauration obtenues à partir de la requête d'objet pour trier la liste contenue dans cet appel. Ceci garantit que les objets sont restaurés de la mémoire de la manière la plus efficace possible sans rembobiner ou remonter les bandes de données.

Lors de l'obtention d'objets entiers, la valeur *dsmGetList.numObjID* maximale est DSM_MAX_GET_OBJ. Lors de l'obtention d'objets partiels, la valeur maximale est DSM_MAX_PARTIAL_GET_OBJ.

Faites suivre l'appel à **dsmBeginGetData** d'un ou de plusieurs appels à **dsmGetObj** pour obtenir chacun des objets de la liste. Une fois que chaque objet a été obtenu, ou lorsqu'aucune donnée supplémentaire pour l'objet n'est requise, l'appel **dsmEndGetObj** est envoyé.

Lorsque tous les objets sont obtenus ou lorsque **dsmEndGetObj** est annulée, l'appel **dsmEndGetData** est envoyé. Vous pouvez alors relancer le cycle.

Syntaxe

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

Paramètres

dsUint32_t dsmHandle (I)
Descripteur qui associe cet appel à un appel **dsmInitEx** précédent.

dsBool_t mountWait (I)
Valeur booléenne true ou false qui indique si le client d'application attend ou non que les supports déconnectés soient montés si les données requises sont actuellement hors ligne. Si mountWait est défini sur true, l'application attend que le serveur monte le support requis. L'application attend que le support soit monté ou la demande est annulée.

dsmGetType getType (I)
Type énuméré composé de **gtBackup** et de **gtArchive** qui indique le type d'objet à obtenir.

dsmGetList *dsmGetObjListP (I)
Structure contenant des informations sur les objets entiers ou partiels à restaurer ou à récupérer. Cette structure pointe vers une liste d'ID objet et, dans le cas d'une restauration ou d'une récupération d'objet partiel, vers une liste de décalages et de longueurs associés. Si votre application utilise la fonction de restauration et de récupération d'objet partiel, définissez la zone **dsmGetList.stVersion** sur **dsmGetListPORVersion**. Dans une restauration ou une récupération d'objet partiel, vous ne pouvez pas compresser les données lors de leur envoi. Pour appliquer cette règle, définissez **ObjAttr.objCompressed** sur *bTrue*.

Pour plus d'informations sur cette structure, voir figure 19, à la page 75 et Annexe B, «Fichiers source des définitions de type d'API», à la page 165.

Pour plus d'informations sur la restauration ou la récupération d'objets partiels, voir «Restauration ou récupération partielle de l'objet», à la page 69.

Codes retour

Les numéros de code retour sont indiqués entre parenthèses ( ).

Tableau 20. Codes retour pour dsmBeginGetData

Code retour	Explications
DSM_RC_ABORT_INVALID_OFFSET (33)	Le décalage indiqué lors d'une extraction d'objet partiel est supérieur à la longueur de l'objet.
DSM_RC_ABORT_INVALID_LENGTH (34)	La longueur spécifiée lors d'une récupération d'objet partiel est supérieure à la longueur de l'objet, ou le décalage ajouté à la longueur dépasse l'extrémité de l'objet.
DSM_RC_NO_MEMORY (102)	Il ne reste plus de mémoire RAM pour terminer la requête.
DSM_RC_NUMOBJ_EXCEED (2029)	La valeur **dsmGetList.numObjId** est supérieure à celle de DSM_MAX_GET_OBJ.
DSM_RC_OBJID_NOTFOUND (2063)	ID objet introuvable. L'objet n'a pas été restauré.
DSM_RC_WRONG_VERSION_PARM (2065)	La version de l'API du client d'application est différente de la version de la bibliothèque de IBM Spectrum Protect.

dsmBeginQuery

L'appel de fonction **dsmBeginQuery** lance une demande de requête au serveur pour obtenir des informations sur les données, les espaces fichier et les classes de gestion.

La commande **dsmBeginQuery** peut interroger les éléments suivants :
- Données archivées
- Données sauvegardées
- Données sauvegardées actives
- Espaces fichier
- Classes de gestion

Les données de requête renvoyées à partir de l'appel sont obtenues par un ou plusieurs appels à **dsmGetNextQObj**. Lorsque la requête est terminée, l'appel **dsmEndQuery** est envoyé.

Syntaxe

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

Paramètres

dsUint32_t dsmHandle (I)

Descripteur qui associe cet appel à un appel **dsmInitEx** précédent.

dsmQueryType queryType (I)

Indique le type de requête à exécuter. Affectez l'une des options suivantes :

qtArchive

Lance la requête sur les objets archivés.

qtBackup

Lance la requête sur les objets sauvegardés.

qtBackupActive

Lance la requête sur les objets sauvegardés actifs uniquement pour le nom d'espace fichier entier que vous transmettez. Cette requête, appelée «raccourci», représente un moyen efficace de lancer une requête sur des objets actifs à partir de la mémoire.

Prérequis : Vous devez être connecté en tant que superutilisateur sous un système UNIX ou Linux.

qtFilespace

Lance une requête sur les espaces fichier enregistrés.

qtMC

Lance une requête sur les classes de gestion définies.

qtBackupGroups

Lance une requête sur les groupes fermés.

qtOpenGroups

Lance une requête sur les groupes ouverts.

qtProxyNodeAuth

Lance une requête sur les postes sur lesquels ce poste peut agir en tant que proxy.

qtProxyNodePeer

Lance une requête sur les poste de même niveau et dotés de la même cible.

dsmQueryBuff *queryBuffer (I)

Identifie un pointeur vers une mémoire tampon mappée sur une structure de données particulière. Cette structure est associée au type de requête que vous transmettez. Ces structures contiennent les critères de sélection pour chaque type de requête. Renseignez les zones de chaque structure pour indiquer la portée de la requête que vous souhaitez effectuer. La zone `stVersion` de chaque structure contient le numéro de version de la structure.

Les structures de données et leurs zones correspondantes incluent les éléments suivants :

qryArchiveData

objName

Nom d'objet complet. Vous pouvez utiliser un caractère générique, tel qu'un astérisque (*) ou un point d'interrogation (?), dans la partie de haut niveau ou bas niveau du nom. Un astérisque remplace zéro, un ou plusieurs caractères, et un point d'interrogation remplace exactement un caractère. La zone `objType` de `objName` peut contenir l'une des valeurs suivantes :

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

Pour plus d'informations sur les noms de haut et de bas niveau, consultez la rubrique suivante : «Noms de haut et de bas niveau», à la page 25.

owner

Nom du propriétaire de l'objet.

insDateLowerBound

Limite inférieure pour la date d'insertion à laquelle l'objet a été archivé. Pour obtenir la limite inférieure par défaut, définissez le composant year sur `DATE_MINUS_INFINITE`.

insDateUpperBound

Limite supérieure pour la date d'insertion à laquelle l'objet a été archivé. Pour obtenir la limite supérieure par défaut, définissez le composant year sur `DATE_PLUS_INFINITE`.

expDateLowerBound

Limite inférieure de la date d'expiration. Les valeurs par défaut des deux zones de date d'expiration sont identiques à celles des zones de date d'insertion.

expDateUpperBound

Limite supérieure de la date d'expiration.

descr

Description de l'archive. Entrez un astérisque (*) pour rechercher toutes les descriptions.

qryBackupData

objName

Nom d'objet complet. Vous pouvez utiliser un caractère générique, tel qu'un astérisque (*) ou un point d'interrogation (?), dans la partie de haut niveau ou bas niveau du nom. Un astérisque remplace zéro, un ou plusieurs caractères, et un point d'interrogation remplace exactement un caractère. La zone `objType` de `objName` peut contenir l'une des valeurs suivantes :

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

Pour plus d'informations sur les noms de haut et de bas niveau, consultez la rubrique suivante : «Noms de haut et de bas niveau», à la page 25.

owner

Nom du propriétaire de l'objet.

objState

Vous pouvez effectuer une requête relative à l'un des états d'objet suivants :

- `DSM_ACTIVE`
- `DSM_INACTIVE`

- DSM_ANY_MATCH

pitDate

Valeur de date de référence. Une requête utilisant cette zone renvoie le dernier objet qui a été sauvegardé avant cette date et cette heure. L'état objState peut actif ou inactif. Les objets supprimés avant pitDate ne sont pas renvoyés. Par exemple :

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

Vendredi, appelez la requête avec une valeur de date de référence définie à Mercredi à 12:00:00 du matin. L'appel renvoie les informations suivantes :

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

L'appel ne renvoie pas DEF car cet objet a été supprimé avant la valeur de date de référence.

qryABackupData

objName

Nom d'objet complet. Vous pouvez utiliser un caractère générique, tel qu'un astérisque (*) ou un point d'interrogation (?), dans la partie de haut niveau ou bas niveau du nom. Un astérisque remplace zéro, un ou plusieurs caractères, et un point d'interrogation remplace exactement un caractère. La zone objType de objName peut contenir l'une des valeurs suivantes :

- DSM_OBJ_FILE
- DSM_OBJ_DIRECTORY
- DSM_OBJ_ANY_TYPE

Pour plus d'informations sur les noms de haut et de bas niveau, consultez la rubrique suivante : «Noms de haut et de bas niveau», à la page 25.

qryFSData

fsName

Entrez le nom d'un espace fichier spécifique dans cette zone ou entrez un astérisque (*) pour extraire des informations sur tous les espaces fichier enregistrés.

qryMCData

mcName

Entrez le nom d'une classe de gestion spécifique ou entrez une chaîne vide (« ») pour extraire des informations sur toutes les classes de gestion.

Remarque : Vous ne pouvez pas utiliser l'astérisque (*).

mcDetail

Détermine si les informations sur les groupes de copie d'archivage et de sauvegarde de la classe de gestion sont renvoyées. Les valeurs suivantes sont valides :

- bTrue
- bFalse

qryBackupGroup :

groupType

Le type de groupe est DSM_GROUPTYPE_PEER.

fsName

Nom d'espace fichier.

owner

ID du propriétaire.

groupLeaderObjId

ID objet du chef de groupe.

objType

Type d'objet.

qryProxyNodeAuth :

targetNodeName

Nom de poste cible.

peerNodeName

Nom de poste homologue.

hlAddress

Adresse homologue du nom de haut niveau.

llAddress

Adresse homologue du nom de bas niveau.

qryProxyNodePeer :

targetNodeName

Nom de poste cible.

peerNodeName

Nom de poste homologue.

hlAddress

Adresse homologue du nom de haut niveau.

llAddress

Adresse homologue du nom de bas niveau.

Codes retour

Le tableau suivant décrit les codes retour pour l'appel de fonction **dsmBeginQuery**.

Tableau 21. Codes retour pour dsmBeginQuery

Code retour	Numéro de code retour	Explications
DSM_RC_NO_MEMORY	102	Il ne reste plus de mémoire pour terminer la requête.
DSM_RC_FILE_SPACE_NOT_FOUND	124	L'espace fichier indiqué est introuvable.

Tableau 21. Codes retour pour dsmBeginQuery (suite)

Code retour	Numéro de code retour	Explications
DSM_RC_NO_POLICY_BLK	2007	Informations sur les règles d'administration du serveur non disponibles.
DSM_RC_INVALID_OBJTYPE	2010	Type d'objet non valide.
DSM_RC_INVALID_OBJOWNER	2019	Nom de propriétaire objet non valide.
DSM_RC_INVALID_OBJSTATE	2024	Condition d'objet non valide.
DSM_RC_WRONG_VERSION_PARM	2065	La version de l'API du client d'application est différente de la version de la bibliothèque de IBM Spectrum Protect.

dsmBeginTxn

L'appel de fonction **dsmBeginTxn** démarre une ou plusieurs transactions IBM Spectrum Protect qui lancent une action complète ; soit toutes les actions aboutissent, soit aucune. Une action peut être un appel unique ou une série d'appels. Par exemple, un appel **dsmSendObj** suivi de plusieurs appels **dsmSendData** peut être considéré comme une action unique. De la même manière, un appel **dsmSendObj** avec un paramètre **dataBlkPtr** qui indique une zone de données contenant l'objet à sauvegarder est également considéré comme une action unique.

Essayez de regrouper plusieurs objets en une seule transaction pour des opérations de transfert de données. Le regroupement d'objets permet d'obtenir des améliorations de performance significatives dans le système IBM Spectrum Protect. Pour le cient comme le serveur, le démarrage et la fin de chaque transaction consomment une quantité non négligeable de ressources système.

Les opérations que vous pouvez effectuer dans le cadre d'une transaction unique sont limitées. Ces restrictions sont notamment les suivantes :

- Le nombre d'objets que vous pouvez envoyer ou supprimer dans une seule transaction est limité. Cette limite est située dans les données que **dsmQuerySessInfo** renvoie dans la zone *ApiSessInfo.maxObjPerTxn*. Ceci correspond à l'option de serveur *TxnGroupMax*.
- Tous les objets envoyés au serveur (de sauvegarde ou d'archivage) en une seule transaction doivent avoir la même destination de copie que celle définie dans la liaison de classe de gestion pour l'objet. Cette valeur est située dans les données que **dsmBindMC** renvoie dans les zones **mcBindKey.backup_copy_dest** ou **mcBindKey.archive_copy_dest**.

Avec l'API, soit le client d'application peut surveiller et contrôler ces restrictions, soit l'API elle-même peut les contrôler. Si l'API contrôle les restrictions, les codes retour appropriés à partir des appels API informent le client d'application lorsqu'une ou plusieurs restrictions sont atteintes.

Associez toujours un appel **dsmBeginTxn** à un appel **dsmEndTxn** pour optimiser l'ensemble d'actions dans une paire d'appels **dsmBeginTxn** et **dsmEndTxn**.

Syntaxe

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

Paramètres

dsUint32_t dsmHandle (I)
Descripteur qui associe cet appel à un appel **dsmInitEx** précédent.

Codes retour

Les numéros de code retour sont indiqués entre parenthèses ( ).

Tableau 22. Codes retour pour dsmBeginTxn

Code retour	Explications
DSM_RC_ABORT_NODE_NOT_AUTHORIZED (36)	FROMNODE ou FROMOWNER n'est pas autorisé pour les opérations TXN.

dsmBindMC

L'appel de fonction **dsmBindMC** associe, ou lie, une classe de gestion à l'objet transmis. L'objet est transmis par l'intermédiaire de la liste inclusive-exclusive qui est pointée vers le fichier d'options. Si aucune correspondance n'est trouvée dans la liste inclusive pour une classe de gestion spécifique, la classe de gestion par défaut est affectée. La liste exclusive peut exclure les objets d'une sauvegarde, mais non d'une archive.

Le client d'application peut utiliser les paramètres qui sont renvoyés dans la structure `mcBindKey` pour déterminer si cet objet doit être sauvegardé ou archivé, ou si une nouvelle transaction doit être démarrée en raison de différentes destinations de copie. Pour plus d'informations, voir **dsmBeginTxn**.

Appelez **dsmBindMC** avant d'appeler **dsmSendObj** car à chaque objet doit être associée une classe de gestion. Cet appel peut être exécuté à l'intérieur ou à l'extérieur d'une transaction. Par exemple, à l'intérieur d'un transaction à plusieurs objets, si **dsmBindMC** indique que l'objet a une copie de destination différente de celle de l'objet précédent, la transaction doit être terminée et une nouvelle transaction démarrée. Dans ce cas, un autre **dsmBindMC** n'est pas nécessaire car il n'en y a déjà eu un d'exécuté pour cet objet.

Syntaxe

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

Paramètres

dsUint32_t dsmHandle (I)
Descripteur qui associe cet appel à un appel **dsmInitEx** précédent.

dsmObjName *objNameP (I)
Pointeur vers la structure qui contient le nom d'espace fichier, le nom d'objet de haut niveau, le nom d'objet de bas niveau et le type d'objet.

dsmSendType sendType (I)
Identifie si cette liaison de classe de gestion est exécutée pour les envois d'archivage ou de sauvegarde. Les valeurs possibles pour cet appel comprennent notamment les valeurs suivantes :

Nom	Description
stBackup	Objet de sauvegarde
stArchive	Objet archivé
stBackupMountWait	Objet de sauvegarde
stArchiveMountWait	Objet archivé

Pour l'appel **dsmBindMC**, stBackup et stBackupMountWait sont équivalents, et stArchive et stArchiveMountWait sont équivalents.

mcBindKey *mcBindKeyP (0)
Ceci est l'adresse d'une structure mcBindKey à laquelle est renvoyée l'information de classe de gestion. Le client d'application peut utiliser l'informations qui est renvoyée ici pour déterminer si cet objet est adapté à une transaction à objets multiples, ou pour effectuer une requête de classe de gestion sur la classe de gestion liée à cet objet.

Codes retour

Les numéros de code retour sont indiqués entre parenthèses ( ).

Tableau 23. Codes retour pour dsmBindMC

Code retour	Explications
DSM_RC_NO_MEMORY (102)	Il ne reste plus de mémoire RAM pour terminer la requête.
DSM_RC_INVALID_PARM (109)	L'un des paramètres transmis a une valeur non valide.
DSM_RC_TL_EXCLUDED (185)	L'objet de sauvegarde est exclu et ne peut pas être envoyé.
DSM_RC_INVALID_OBJTYPE (2010)	Type d'objet non valide.
DSM_RC_INVALID_SENDTYPE (2022)	Type d'envoi non valide.
DSM_RC_WRONG_VERSION_PARM (2065)	La version de l'API du client d'application est différente de la version de la bibliothèque de IBM Spectrum Protect.

dsmChangePW

L'appel de fonction **dsmChangePW** change un mot de passe IBM Spectrum Protect. Sous un système d'exploitation multi-utilisateur tel qu'UNIX ou Linux, seul le superutilisateur ou l'utilisateur autorisé peut utiliser cet appel.

Sur les systèmes d'exploitation Windows, vous pouvez indiquer le mot de passe dans le fichier dsm.opt. Dans ce cas, **dsmChangePW** ne met pas à jour le fichier dsm.opt. Une fois que l'appel à **dsmChangePW** a été effectué, vous pouvez mettre à jour séparément le fichier dsm.opt.

Cet appel doit être correctement exécuté si **dsmInitEx** renvoie DSM_RC_VERIFIER_EXPIRED. La session se termine si l'appel **dsmChangePW** échoue dans ce cas.

Si **dsmChangePW** est appelé pour une autre raison, la session reste ouverte quel que soit le code retour.

Syntaxe

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

Paramètres

dsUint32_t dsmHandle (I)
Descripteur qui associe cet appel à un appel **dsmInitEx** précédent.

char *oldPW (I)
Ancien mot de passe de l'appelant. La longueur maximale est DSM_MAX_VERIFIER_LENGTH.

char *newPW (I)
Nouveau mot de passe de l'appelant. La longueur maximale est DSM_MAX_VERIFIER_LENGTH.

Codes retour

Les numéros de code retour sont indiqués entre parenthèses ( ).

Tableau 24. Codes retour pour dsmChangePW

Code retour	Explications
DSM_RC_ABORT_BAD_VERIFIER (6)	Un mot de passe incorrect a été entré.
DSM_RC_AUTH_FAILURE (137)	Echec de l'authentification. L'ancien mot de passe est incorrect.
DSM_RC_NEWPW_REQD (2030)	Une valeur doit être entrée pour le nouveau mot de passe.
DSM_RC_OLDPW_REQD (2031)	Une valeur doit être entrée pour l'ancien mot de passe.
DSM_RC_PASSWD_TOOLONG (2103)	Le mot de passe indiqué est trop long.
DSM_RC_NEED_ROOT (2300)	L'appelant de l'API doit être un superutilisateur ou un utilisateur autorisé.

dsmCleanUp

l'appel de fonction **dsmCleanUp** est utilisé si **dsmSetUp** a été appelé. L'appel de fonction **dsmCleanUp** doit être appelé après **dsmTerminate**. Vous ne pouvez plus effectuer d'autres appels après avoir appelé **dsmCleanUp**.

Il n'existe aucun code retour spécifique à cet appel.

Syntaxe

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

Paramètres

dsBool_t mtFlag (I)
Ce paramètre indique que l'API a été utilisée en mode monotâche ou multitâche. Les valeurs possibles incluent :

- DSM_SINGLETHREAD
- DSM_MULTITHREAD

dsmDeleteAccess

L'appel de fonction **dsmDeleteAccess** supprime les règles d'autorisation en cours pour les versions de sauvegarde ou les copies archivées de vos objets. Lorsque vous supprimez une règle d'autorisation, vous pouvez révoquer l'accès d'un utilisateur à tout fichier spécifié par cette règle.

Lorsque vous utilisez **dsmDeleteAccess**, vous pouvez supprimer uniquement une règle à la fois. Obtenez l'ID règle à l'aide de la commande **dsmQueryAccess**.

Il n'existe aucun code retour spécifique à cet appel.

Syntaxe

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

Paramètres

dsUint32_t dsmHandle (I)
Descripteur qui associe cet appel à un appel **dsmInitEx** précédent.

dsUint32_t ruleNum (I)
ID règle de la règle d'accès supprimée. Cette valeur est obtenue à partir d'un appel de fonction **dsmQueryAccess**.

dsmDeleteFS

L'appel de fonction **dsmDeleteFS** supprime un espace fichier de la mémoire. Pour supprimer un espace fichier, vous devez avoir reçu les autorisations appropriées de votre administrateur IBM Spectrum Protect. Pour vérifier si vous disposez des autorisations requises, appelez **dsmQuerySessInfo**. Cet appel de fonction renvoie une structure de données de type *ApiSessInfo*, incluant deux zones, *archDel* et *backDel*.

Remarque :

- Sous un système d'exploitation UNIX ou Linux, seul un superutilisateur ou un utilisateur autorisé peut supprimer un espace fichier.
- Si l'espace fichier à supprimer contient des versions de sauvegarde, vous devez disposer du droit de suppression de sauvegarde (***backDel*** = BACKDEL_YES). Si l'espace fichier contient des copies d'archivage, vous devez disposer du droit de suppression d'archive (*archDel* = ARCHDEL_YES). Si l'espace fichier contient à la fois des versions de sauvegarde et des copies d'archivage, vous devez disposer des deux types de droit de suppression.
- Lorsqu'un serveur Archive Manager est utilisé, il est impossible de supprimer réellement un espace fichier. Cet appel de fonction renverra *rc=0* même si l'espace fichier n'a pas été réellement supprimé. La seule manière de vérifier que l'espace fichier a été supprimé est de lancer une requête sur l'espace fichier au serveur.
- La fonction de suppression d'espace fichier du serveur IBM Spectrum Protect est un processus en arrière-plan. Si des erreurs autres que celles détectées avant de transmettre un code retour se produisent, elles sont enregistrées dans le journal serveur de IBM Spectrum Protect.

Syntaxe

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

Paramètres

dsUint32_t dsmHandle (I)
Descripteur qui associe cet appel à un appel **dsmInitEx** précédent.

char *fsName (I)
Pointeur vers le nom d'espace fichier à supprimer. Le caractère générique n'est pas autorisé.

unsigned char repository (I)
Indique si l'espace fichier à supprimer est un référentiel de sauvegarde, un référentiel d'archivage, ou les deux. Les valeurs possibles pour cette zone incluent les valeurs suivantes :

```
DSM_ARCHIVE_REP     /* référentiel d'archivage  */
DSM_BACKUP_REP      /* référentiel de sauvegarde    */
DSM_REPOS_ALL       /* tous les types de référentiel */
```

Codes retour

Les numéros de code retour sont indiqués entre parenthèses ( ).

Tableau 25. Codes retour pour dsmDeleteFS

Code retour	Explications
DSM_RC_ABORT_NOT_AUTHORIZED (27)	Vous ne possédez pas les droits d'accès nécessaires pour supprimer l'espace fichier.
DSM_RC_INVALID_REPOS (2015)	Valeur non valide pour le référentiel.
DSM_RC_FSNAME_NOTFOUND (2060)	Nom d'espace fichier introuvable.
DSM_RC_NEED_ROOT (2300)	L'appelant de l'API doit être un superutilisateur.

dsmDeleteObj

L'appel de fonction **dsmDeleteObj** désactive ou supprime les objets de sauvegarde ou supprime les objets archivés dans la mémoire. Le type **dtBackup** désactive uniquement la copie de sauvegarde actuellement active. Le type **dtBackupID** supprime du serveur l'ID objet spécifié. Appelez cette fonction à partir d'une transaction.

Pour plus d'informations, voir **dsmBeginTxn**.

Avant d'envoyer **dsmDeleteObj**, envoyez la séquence de requête décrite dans «Interrogation du système IBM Spectrum Protect», à la page 35 pour obtenir les informations pour **delInfo**. L'appel à **dsmGetNextQObj** renvoie une structure de données nommée **qryRespBackupData** pour les requêtes de sauvegarde ou **qryRespArchiveData** pour les requêtes d'archivage. Ces structures de données contiennent les informations dont vous avez besoin pour **delInfo**.

La valeur de **maxObjPerTxn** détermine le nombre maximal d'objets que vous pouvez supprimer dans le cadre d'une transaction unique. Pour obtenir cette valeur, appelez **dsmQuerySessInfo**.

Conseil : Votre poste doit disposer de l'autorisation appropriée définie par votre administrateur. Pour supprimer des objets archivés, vous devez disposer du droit de suppression d'archive. Vous ne devez pas nécessairement disposer du droit de suppression de sauvegarde pour désactiver un objet sauvegardé.

Syntaxe

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

Paramètres

dsUint32_t dsmHandle (I)
Descripteur qui associe cet appel à un appel **dsmInitEx** précédent.

dsmDelType delType (I)
Indique le type d'objet (de sauvegarde ou archivé) à supprimer. Les valeurs possibles incluent :

Nom	Description
dtArchive	L'objet à supprimer a été précédemment archivé.
dtBackup	L'objet à désactiver a été sauvegardé précédemment.
dtBackupID	L'objet à supprimer a été sauvegardé précédemment. **Restriction :** L'utilisation de **delType** avec *objID* entraîne la suppression de l'objet de sauvegarde à partir du serveur. Un objet peut être supprimé uniquement par son propriétaire. Vous pouvez supprimer n'importe quelle version (active ou inactive) d'un objet. Le serveur réconcilie les versions. Si vous supprimez une version active d'un objet, la première version inactive devient active. Si vous supprimez une version inactive d'un objet, toutes les anciennes versions avancent. Le poste doit être enregistré avec l'autorisation **backDel**.

dsmDelInfo delInfo (I)
Structure dont les zones identifient l'objet. Les zones sont différentes, selon que l'objet est un objet de sauvegarde ou un objet archivé. La structure pour désactiver un objet de sauvegarde, `delBack`, contient le nom d'objet et le groupe de copie d'objet. La structure pour un objet archivé, `delArch`, contient l'ID objet.

La structure pour supprimer un objet de sauvegarde, `delBackID`, contient l'ID objet.

Codes retour

Les numéros de code retour sont indiqués entre parenthèses ( ).

Tableau 26. Codes retour pour dsmDeleteObj

Code retour	Explications
DSM_RC_FS_NOT_REGISTERED (2061)	Le nom d'espace fichier n'est pas enregistré.
DSM_RC_WRONG_VERSION_PARM (2065)	La version de l'API du client d'application est différente de la version de la bibliothèque de IBM Spectrum Protect.

dsmEndGetData

L'appel de fonction **dsmEndGetData** termine une session **dsmBeginGetData** qui obtient des objets à partir de la mémoire.

L'appel de fonction **dsmEndGetData** démarre une fois que tous les objets que vous souhaitez restaurer ont été traités ou termine le processus d'obtention de manière prématurée. Appelez **dsmEndGetData** pour terminer une session **dsmBeginGetData** afin de pouvoir poursuivre un autre traitement.

Selon le moment auquel **dsmEndGetData** est appelé, l'API peut avoir besoin de terminer le traitement d'un flux de données partiel avant d'arrêter le processus. L'appelant, par conséquent, ne doit pas attendre un retour immédiat de cet appel. Utilisez **dsmTerminate** si l'application a besoin de fermer la session et de mettre fin à la restauration immédiatement.

Il n'existe aucun code retour spécifique à cet appel.

Syntaxe

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

Paramètres

dsUint32_t dsmHandle (I)
Descripteur qui associe cet appel à un appel **dsmInitEx** précédent.

dsmEndGetDataEx

L'appel de fonction **dsmEndGetDataEx** fournit le total d'octets hors réseau local qui ont été envoyés. C'est une extension de l'appel de fonction **dsmEndGetData**.

Syntaxe

Il n'existe aucun code retour spécifique à cet appel.

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

Paramètres

dsmEndGetDataExIn_t *dsmEndGetDataExInP (I)
Transmet la fonction dsmHandle d'objet get final qui identifie la session et l'associe aux appels suivants.

dsmEndGetDataExOut_t * dsmEndGetDataExOutP (O)
Cette structure contient le paramètre d'entrée :

totalLFBytesRecv
Nombre total d'octets hors réseau local reçus.

dsmEndGetObj

L'appel de fonction **dsmEndGetObj** termine une session **dsmGetObj** qui obtient des données pour un objet spécifié.

Démarrez l'appel **dsmEndGetObj** après qu'une fin de données a été reçue pour l'objet. Ceci indique que toutes les données ont été reçues ou qu'aucune donnée supplémentaire ne sera reçue pour cet objet. Pour pouvoir démarrer un autre appel **dsmGetObj**, vous devez précédemment appeler **dsmEndGetObj** .

Selon le moment auquel **dsmEndGetObj** est appelé, l'API peut avoir besoin de terminer le traitement d'un flux de données partiel avant de pouvoir arrêter le processus. N'attendez pas un retour immédiat de cet appel.

Syntaxe

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

Paramètres

dsUint32_t dsmHandle (I)
Descripteur qui associe cet appel à un appel **dsmInitEx** précédent.

Codes retour

Les numéros de code retour sont indiqués entre parenthèses ( ).

Tableau 27. Codes retour pour dsmEndGetObj

Code retour	Explications
DSM_RC_NO_MEMORY (102)	Il ne reste plus de mémoire RAM pour terminer la requête.

dsmEndQuery

L'appel de fonction **dsmEndQuery** signifie la fin d'une action **dsmBeginQuery**. Le client d'application envoie **dsmEndQuery** pour exécuter une requête. Cet appel est envoyé après l'obtention de toutes les réponses à la requête par l'intermédiaire de **dsmGetNextQObj** ou il est envoyé pour terminer une requête avant l'envoi de toutes les données.

Conseil : IBM Spectrum Protect continue à envoyer les données de requête à partir du serveur vers le client, mais l'API élimine les données restantes.

Une fois qu'une requête **dsmBeginQuery** est envoyée, une requête **dsmEndQuery** doit être préalablement envoyée pour permettre le démarrage d'une autre activité.

Il n'existe aucun code retour spécifique à cet appel.

Syntaxe

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

Paramètres

dsUint32_t dsmHandle (I)
Descripteur qui associe cet appel à un appel **dsmInitEx** précédent.

dsmEndSendObj

L'appel de fonction **dsmEndSendObj** indique la fin des données qui est envoyée vers la mémoire.

Entrez l'appel de fonction **dsmEndSendObj** pour indiquer la fin des données à partir des appels **dsmSendObj** et **dsmSendData**. Une violation de protocole se produit si cette opération n'est pas exécutée. Il y a une exception à cette règle lorsque vous appelez **dsmEndTxn** pour terminer la transaction. Dans ce cas, toutes les données envoyées pour cette transaction sont éliminées.

Syntaxe

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

Paramètres

dsUint32_t dsmHandle (I)
Descripteur qui associe cet appel à un appel **dsmInitEx** précédent.

Codes retour

Les numéros de code retour sont indiqués entre parenthèses ( ).

Tableau 28. Codes retour pour dsmEndSendObj

Code retour	Explications
DSM_RC_NO_MEMORY (102)	Il ne reste plus de mémoire RAM pour terminer cette requête.

dsmEndSendObjEx

L'appel de fonction **dsmEndSendObjEx** fournit des informations supplémentaires à propos du nombre d'octets traités. Ces informations comprennent : le nombre total d'octets envoyés, les informations de compression, les octets hors LAN et les informations de dédoublonnage.

L'appel de fonction **dsmEndSendObjEx** est une extension de l'appel de fonction **dsmEndSendObj**.

Syntaxe

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

Paramètres

dsmEndSendObjExIn_t *dsmEndSendObjExInP (I)
Ce paramètre transmet la fonction dsmHandle d'objet send final qui identifie la session et l'associe aux appels suivants.

dsmEndSendObjExOut_t *dsmEndSendObjExOutP (O)
Ce paramètre transmet les informations sur la fin de l'envoi d'objet :

Nom	Description
totalBytesSent	Nombre total d'octets lus à partir de l'application.
objCompressed	Indicateur qui affiche si l'objet a été compressé.
totalCompressedSize	Taille totale en octets après la compression.
totalLFBytesSent	Nombre total d'octets hors réseau local qui ont été envoyés.
objDeduplicated	Un indicateur s'affiche si l'objet a été dédoublonné par l'API.

Nom	Description
totalDedupSize	Nombre total d'octets envoyés après dédoublonnage.

Codes retour

Les numéros de code retour sont indiqués entre parenthèses ( ).

Tableau 29. Codes retour pour dsmEndSendObjEx

Code retour	Explications
DSM_RC_NO_MEMORY (102)	Il ne reste plus de mémoire RAM pour terminer cette requête.

dsmEndTxn

L'appel de fonction **dsmEndTxn** met fin à une transaction IBM Spectrum Protect. Associez l'appel de fonction **dsmEndTxn** à **dsmBeginTxn** pour identifier l'appel ou l'ensemble d'appels considérés comme une transaction. Le client d'application peut indiquer sur l'appel **dsmEndTxn** si la transaction doit être validée ou terminée.

Effectuez tous les appels suivants dans les limites d'une transaction :

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

Syntaxe

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

Paramètres

dsUint32_t dsmHandle (I)
Descripteur qui associe cet appel à un appel **dsmInitEx** précédent.

dsUint8_t vote (I)
Indique si le client d'application valide toutes les actions exécutées entre l'appel **dsmBeginTxn** précédent et cet appel. Les valeurs suivantes sont possibles :

```
DSM_VOTE_COMMIT     /* valider la transaction en cours    */
DSM_VOTE_ABORT      /* annuler la transaction en cours */
```

Utilisez DSM_VOTE_ABORT uniquement si votre application a trouvé une raison d'arrêter la transaction.

dsUint16_t *reason (O)
Si l'appel à **dsmEndTxn** se termine avec une erreur ou si la valeur de vote n'est pas acceptée, ce paramètre comporte un code raison indiquant la raison de l'échec du vote. Le code retour de cet appel peut être zéro et le code raison peut être différent de zéro. Par conséquent, le client d'application doit toujours vérifier la présence d'erreurs éventuelles à la fois au niveau du code retour et du code raison (if (rc || reason)) pour que vous puissiez considérer que l'opération s'est terminée correctement.

Si l'application indique un vote DSM_VOTE_ABORT, le code raison est DSM_RS_ABORT_BY_CLIENT (3). Voir Annexe A, «Fichier source des codes retour de l'API : dsmrc.h», à la page 155 pour consulter la liste des codes raison

possibles. Les nombres 1 à 50 dans la liste des codes retour sont réservés pour les codes raison. Si le serveur termine la transaction, le code retour est DSM_RC_CHECK_REASON_CODE. Dans ce cas, la valeur de la raison contient des informations supplémentaires sur la cause de l'abandon.

Codes retour

Les numéros de code retour sont indiqués entre parenthèses ( ).

Tableau 30. Codes retour pour dsmEndTxn

Code retour	Explications
DSM_RC_ABORT_CRC_FAILED (236)	Le contrôle de redondance cyclique qui a été reçu du serveur ne correspond pas au contrôle de redondance cyclique qui a été calculé par le client.
DSM_RC_INVALID_VOTE (2011)	La valeur qui a été spécifiée pour vote n'est pas valide.
DSM_RC_CHECK_REASON_CODE (2302)	La transaction a été abandonnée. Vérifiez la zone cause.
DSM_RC_ABORT_STGPOOL_COPY_CONT_NO (241)	L'écriture sur l'un des pools de stockage de copie a échoué et l'option de pool de stockage IBM Spectrum Protect COPYCONTINUE est définie sur NO. La transaction se termine.
DSM_RC_ABORT_RETRY_SINGLE_TXN (242)	Ce code d'abandon indique que la transaction en cours a été abandonnée en raison d'un incident lors d'une opération de stockage. Le problème peut être résolu par l'envoi de chaque fichier dans une transaction particulière. Cette erreur est typique dans les cas suivants : • Le pool de stockage suivant comporte une liste de pool de stockage de copie différente. • L'opération est basculée vers ce pool au cours d'une transaction. .

dsmEndTxnEx

L'appel de fonction **dsmEndTxnEx** fournit des informations sur l'ID objet du chef de groupe que vous pouvez utiliser avec l'appel de fonction **dsmGroupHandler**. C'est une extension de l'appel de fonction **dsmEndTxn**.

Syntaxe

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

Paramètres

dsmEndTxnExIn_t *dsmEndTxnExInP (I)
Cette structure contient les paramètres suivants :

dsmHandle
Descripteur qui identifie la session et l'associe aux appels IBM Spectrum Protect suivants.

dsUint8_t vote (I)
Indique si le client d'application valide toutes les actions exécutées entre l'appel **dsmBeginTxn** précédent et cet appel. Les valeurs possibles sont :

```
DSM_VOTE_COMMIT     /* valider la transaction en cours    */
DSM_VOTE_ABORT      /* annuler la transaction en cours */
```

Utilisez DSM_VOTE_ABORT uniquement si votre application a trouvé une raison d'arrêter la transaction.

dsmEndTxnExOut_t *dsmEndTxnExOutP (0)
Cette structure contient les paramètres suivants :

dsUint16_t *reason (0)
Si l'appel à **dsmEndTxnEx** se termine avec une erreur ou si la valeur de *vote* n'est pas acceptée, ce paramètre comporte un code raison indiquant la raison de l'échec du vote.

Conseil : Le code retour de cet appel peut être zéro et le code raison peut être différent de zéro. Par conséquent, le client d'application doit toujours vérifier la présence d'erreurs éventuelles à la fois au niveau du code retour et du code raison (`if (rc || reason)`) pour que vous puissiez considérer que l'opération s'est terminée correctement.

Si l'application indique un vote DSM_VOTE_ABORT, le code raison est DSM_RS_ABORT_BY_CLIENT (3). Voir Annexe A, «Fichier source des codes retour de l'API : `dsmrc.h`», à la page 155 pour consulter la liste des codes raison possibles. Les nombres 1 à 50 dans la liste des codes retour sont réservés pour les codes raison. Si le serveur termine la transaction, le code retour est DSM_RC_CHECK_REASON_CODE. Dans ce cas, la valeur de la raison contient des informations supplémentaires sur la cause de l'abandon.

groupLeaderObjId
L'ID objet du chef de groupe qui est renvoyé lorsque que l'indicateur DSM_ACTION_OPEN est utilisé avec l'appel **dsmGroupHandler**.

Codes retour

Les numéros de code retour sont indiqués entre parenthèses ( ).

Tableau 31. Codes retour pour dsmEndTxnEx

Code retour	Description
DSM_RC_INVALID_VOTE (2011)	La valeur qui a été indiquée pour vote n'est pas valide.
DSM_RC_CHECK_REASON_CODE (2302)	La transaction a été abandonnée. Vérifiez la zone cause.
DSM_RC_ABORT_STGPOOL_COPY_CONT_NO (241)	L'écriture sur l'un des pools de stockage de copie a échoué et l'option de pool de stockage IBM Spectrum Protect COPYCONTINUE était définie sur NO. La transaction se termine.
DSM_RC_ABORT_RETRY_SINGLE_TXN (242)	Au cours d'une opération d'écriture simultanée, un objet de la transaction est envoyé à une cible contenant des pools de stockage de copie différents. Terminez la transaction en cours et renvoyez chaque objet dans sa propre transaction.

dsmGetData

L'appel de fonction **dsmGetData** obtient un flot d'octets de données de IBM Spectrum Protect et le place dans la mémoire tampon de l'appelant. Le client d'application appelle **dsmGetData** lorsqu'il reste des données supplémentaires à recevoir d'un appel **dsmGetObj** ou **dsmGetData** précédent.

Syntaxe

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

Paramètres

dsUint32_t dsmHandle (I)
Descripteur qui associe cet appel à un appel **dsmInitEx** précédent.

DataBlk *dataBlkPtr (I/O)
Pointe vers une structure qui inclut à la fois un pointeur vers la mémoire tampon pour les données reçues et la taille de la mémoire tampon. En retour, cette structure contient le nombre d'octets réellement transférés. Pour la définition du type, voir Annexe B, «Fichiers source des définitions de type d'API», à la page 165.

Codes retour

Les numéros de code retour sont indiqués entre parenthèses ( ).

Tableau 32. Codes retour pour dsmGetData

Code retour	Description
DSM_RC_ABORT_INVALID_OFFSET (33)	Le décalage indiqué lors d'une extraction d'objet partiel est supérieur à la longueur de l'objet.
DSM_RC_ABORT_INVALID_LENGTH (34)	La longueur spécifiée lors d'une extraction d'objet partiel est supérieure à la longueur de l'objet, ou le décalage ajouté à la longueur dépasse l'extrémité de l'objet.
DSM_RC_FINISHED (121)	Traitement terminé. La dernière mémoire tampon a été reçue. Vérifiez la valeur de `numBytes` pour connaître la quantité de données, puis appelez IBM Spectrum ProtectdsmEndGetObj .
DSM_RC_NULL_DATABLKPTR (2001)	Le pointeur du bloc de données est Null.
DSM_RC_ZERO_BUFLEN (2008)	La capacité de la mémoire tampon est zéro pour le pointeur du bloc de données.
DSM_RC_NULL_BUFPTR (2009)	Le pointeur de la mémoire tampon est Null pour le pointeur du bloc de données.
DSM_RC_WRONG_VERSION_PARM (2065)	La version de l'API du client d'application est différente de la version de la bibliothèque de IBM Spectrum Protect.
DSM_RC_MORE_DATA (2200)	Il y a des données supplémentaires à obtenir.

dsmGetBufferData

L'appel de fonction **dsmGetBufferData** reçoit un flot d'octets de données à partir de IBM Spectrum Protect via une mémoire tampon. Après chaque appel, l'application doit copier les données et libérer la mémoire tampon par l'intermédiaire d'un appel à **dsmReleaseBuffer**. Si le nombre de mémoires tampons détenus par l'application est égal à la valeur numTsmBuffers indiquée dans l'appel **dsmInitEx**, la fonction **dsmGetBufferData** est bloquée jusqu'à l'appel d'un **dsmReleaseBuffer**.

Syntaxe

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

Paramètres

getDataExIn_t * dsmGetBufferDataExInP (I)
Cette structure contient le paramètre d'entrée suivant.

dsUint32_t dsmHandle
Descripteur qui identifie la session et l'associe à un appel **dsmInitEx** précédent.

getDataExOut_t * dsmGetBufferDataExOutP (0)
Cette structure contient les paramètres de sortie suivants.

dsUint8_t tsmBufferHandle(0)
Descripteur qui identifie la mémoire tampon reçue.

char *dataPtr(0)
Adresse à laquelle les données ont été écrites.

dsUint32_t numBytes(0)
Nombre réel d'octets écrits par IBM Spectrum Protect.

Codes retour

Les numéros de code retour sont indiqués entre parenthèses ( ).

Tableau 33. Codes retour pour dsmGetBufferData

Code retour	Description
DSM_RC_BAD_CALL_SEQUENCE (2041)	L'appel n'a pas été émis dans l'état approprié.
DSM_RC_OBJ_ENCRYPTED (2049)	Cette fonction ne peut pas être utilisée pour les objets chiffrés.
DSM_RC_OBJ_COMPRESSED (2048)	Cette fonction ne peut pas être utilisée pour les objets compressés.
DSM_RC_BUFF_ARRAY_ERROR (2045)	Une erreur de tableau de mémoires tampons s'est produite.

dsmGetNextQObj

L'appel de la fonction **dsmGetNextQObj** obtient la réponse à la requête suivante à partir d'un appel **dsmBeginQuery** précédent et la place dans la mémoire tampon de l'appelant.

L'appel **dsmGetNextQObj** est effectué une ou plusieurs fois. Chaque fois que la fonction est appelée, un enregistrement de requête est extrait ou une erreur ou un code raison DSM_RC_FINISHED est renvoyé. Si DSM_RC_FINISHED est renvoyé, il n'y a plus aucune donnée à traiter. Lorsque toutes les données de requête ont été extraites, ou si aucune donnée de requête supplémentaire n'est nécessaire, émettez l'appel **dsmEndQuery** afin de mettre fin au traitement de requête.

Le paramètre **dataBlkPtr** doit pointer vers une mémoire tampon qui est définie avec le type de structure **qryResp*Data**. Le contexte dans lequel **dsmGetNextQObj** est appelé détermine le type de structure qui est entré dans la réponse à la requête.

Syntaxe

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

Paramètres

dsUint32_t dsmHandle (I)
Descripteur qui associe cet appel à un appel **dsmInitEx** précédent.

DataBlk *dataBlkPtr (I/O)
Pointe vers une structure qui inclut à la fois un pointeur vers la mémoire tampon pour les données reçues et la taille de la mémoire tampon. Cette mémoire tampon correspond à la structure de la réponse **qryResp*Data**. En retour, cette structure contient le nombre d'octets transférés. La structure associée à chaque type de requête est décrite dans le tableau suivant. Pour plus d'informations sur la définition de type de **DataBlk**, consultez la rubrique suivante : Annexe B, «Fichiers source des définitions de type d'API», à la page 165.

Tableau 34. Structure du pointeur DataBlk

Query	Structure de la réponse	Zones spéciales
qtArchive	**qryRespArchiveData**	**sizeEstimate** Contient la valeur transmise à un appel **dsmSendObj** précédent. **mediaClass** Peut avoir une valeur MEDIA_FIXED si l'objet est sur disque ou MEDIA_LIBRARY si l'objet est sur bande. **clientDeduplicated** Indique si l'objet a été dédoublonné par le client.

Tableau 34. Structure du pointeur DataBlk (suite)

Query	Structure de la réponse	Zones spéciales
qtBackup	**qryRespBackupData**	**restoreOrderExt** Est de type dsUint16_t. Exécutez un tri sur ce champ lors de la restauration de plusieurs objets sur un appel **dsmBeginGetData**. Un exemple de code de tri pour cet appel est présenté dans le modèle d'API, dapiqry.c. Pour obtenir un exemple de tri, consultez la rubrique suivante : figure 16, à la page 71. **sizeEstimate** Contient la valeur transmise à un appel **dsmSendObj** précédent. **mediaClass** Peut avoir une valeur MEDIA_FIXED si l'objet est sur disque ou MEDIA_LIBRARY si l'objet est sur bande. **clientDeduplicated** Indique si l'objet a été dédoublonné par le client.
qtBackupActive	**qryARespBackupData**	
qtBackupGroups	**qryRespBackupData**	**dsBool_t isGroupLeader** Si la valeur est true, cet objet est un chef de groupe.
qtOpenGroups	**qryRespBackupData**	**dsBool_t isOpenGroup;** Si la valeur est true, ce groupe est ouvert et incomplet.

Tableau 34. Structure du pointeur DataBlk (suite)

Query	Structure de la réponse	Zones spéciales
qtFilespace	**qryRespFSData**	**backStartDate** Contient l'horodatage du serveur lorsque l'espace fichier est mis à jour avec l'action **backStartDate**. **backCompleteDate** Contient l'horodatage du serveur lorsque l'espace fichier est mis à jour avec l'action **backCompleteDate**. **lastReplStartDate** Contient l'horodatage de la dernière fois où cette réplication a été lancée sur le serveur. **lastReplCmpltDate** Contient l'horodatage de la dernière fois où cette réplication a été effectuée, même si un incident s'est produit. **lastBackOpDateFromServer** Contient l'horodatage du dernier stockage sauvegardé sur le serveur. **lastBackOpDateFromLocal** Contient l'horodatage du dernier stockage sauvegardé sur le client.
qtMC	**qryRespMCData** **qryRespMCDetailData**	
qtProxyNodeAuth	**qryRespProxyNodeData** **targetNodeName** **peerNodeName** **hlAddress** **llAddress**	
qtProxyNodePeer	**qryRespProaxyNodeData** **targetNodeName** **peerNodeName** **hlAddress** **llAddress**	

Codes retour

Le tableau suivant décrit les codes retour pour l'appel de fonction **dsmGetNextQObj**.

Tableau 35. Codes retour pour l'appel de fonction **dsmGetNextQObj**

Code retour	Numéro de code retour	Description
DSM_RC_ABORT_NO_MATCH	2	Aucune correspondance pour la requête n'a été demandée.
DSM_RC_FINISHED	121	Traitement terminé (start **dsmEndQuery**). Il n'y a plus de données à traiter.

Tableau 35. Codes retour pour l'appel de fonction **dsmGetNextQObj** *(suite)*

Code retour	Numéro de code retour	Description
DSM_RC_UNKNOWN_FORMAT	122	Le fichier que IBM Spectrum Protect a tenté de restaurer ou de récupérer a un format inconnu.
DSM_RC_COMM_PROTOCOL_ERROR	136	Erreur de protocole de communication.
DSM_RC_NULL_DATABLKPTR	2001	Le pointeur ne pointe pas vers un bloc de données.
DSM_RC_INVALID_MCNAME	2025	Nom de classe de gestion non valide.
DSM_RC_BAD_CALL_SEQUENCE	2041	La séquence d'appels n'est pas valide.
DSM_RC_WRONG_VERSION_PARM	2065	La version de l'API du client d'application est différente de la version de la bibliothèque de IBM Spectrum Protect.
DSM_RC_MORE_DATA	2200	Il y a des données supplémentaires à obtenir.
DSM_RC_BUFF_TOO_SMALL	2210	La mémoire tampon est trop petite.

dsmGetObj

L'appel de fonction **dsmGetObj** obtient les données objet demandées à partir du flux de données IBM Spectrum Protect et les place dans la mémoire tampon de l'appelant. l'appel **dsmGetObj** utilise l'ID objet pour obtenir l'objet ou l'objet partiel suivant à partir du flux de données.

Les données pour l'objet indiqué sont placées dans la mémoire tampon vers laquelle pointe **DataBlk**. Si des données supplémentaires sont disponibles, vous devez effectuer un ou plusieurs appels à **dsmGetData** pour recevoir les données objets restantes jusqu'à ce qu'un code retour DSM_RC_FINISHED soit retourné. Consultez la zone numBytes dans **DataBlk** pour vérifier s'il reste des données dans la mémoire tampon.

Les objets doivent être demandés dans l'ordre dans lequel ils sont répertoriés sur l'appel **dsmBeginGetData** dans le paramètre **dsmGetList**. Il y a exception lorsque le client d'application doit passer un objet dans le flux de données pour accéder à un objet plus loin dans la liste. Si l'objet qui est indiqué par l'ID objet n'est pas l'objet suivant dans le flux de données, le flux de données est traité jusqu'à ce que l'objet soit localisé, ou le flux est terminé. Utilisez cette fonction avec précaution, car il peut être nécessaire de traiter et d'éliminer de grandes quantités de données pour localiser l'objet demandé.

Exigence : Si **dsmGetObj** renvoie un code d'incident (NOT FINISHED ou MORE_DATA), la session doit être terminée pour permettre l'abandon de l'opération de restauration. Ceci est particulièrement important quand vous utilisez un chiffrement et recevez une clé RC_ENC_WRONG_KEY. Vous devez démarrer une nouvelle session avec la clé appropriée.

Syntaxe

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

Paramètres

dsUint32_t dsmHandle (I)
Descripteur qui associe cet appel à un appel **dsmInitEx** précédent.

ObjID *objIdP (I)
Pointeur vers l'ID de l'objet à restaurer.

DataBlk *dataBlkPtr (I/O)
Pointeur vers la mémoire tampon dans laquelle sont placées les données restaurées.

Codes retour

Les numéros de code retour sont indiqués entre parenthèses ( ).

Tableau 36. Codes retour pour dsmGetObj

Code retour	Description
DSM_RC_ABORT_INVALID_OFFSET (33)	Le décalage qui est indiqué durant une récupération d'objet partiel est supérieur à la longueur de l'objet.
DSM_RC_ABORT_INVALID_LENGTH (34)	La longueur spécifiée lors d'une récupération d'objet partiel est supérieure à la longueur de l'objet, ou le décalage ajouté à la longueur dépasse l'extrémité de l'objet.
DSM_RC_FINISHED (121)	Traitement terminé (démarrer **dsmEndGetObj**).
DSM_RC_WRONG_VERSION_PARM (2065)	La version de l'API du client d'application est différente de la version de la bibliothèque de IBM Spectrum Protect.
DSM_RC_MORE_DATA (2200)	Il y a des données supplémentaires à obtenir.
RC_ENC_WRONG_KEY (4580)	La clé fournie dans l'appel **dsmInitEx**, ou la clé sauvegardée, ne correspond pas à la clé qui a été utilisée pour chiffer cet objet. Terminez la session et fournissez la clé appropriée.

dsmGroupHandler

L'appel de fonction **dsmGroupHandler** exécute une action sur le groupe de fichiers logiques en fonction de l'entrée fournie. Le client met en relation ensemble plusieurs objets pour les référencer et les gérer sur le serveur IBM Spectrum Protect en tant que groupe logique.

Pour plus d'informations, voir «Regroupement de fichiers», à la page 66.

Syntaxe

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

Paramètres

dsmGroupHandlerIn_t *dsmGroupHandlerInP (I)
Transmet les attributs de groupe à l'API.

groupType
Type du groupe. Les valeurs possibles incluent :
- DSM_GROUPTYPE_PEER - groupe homologue

actionType
Action à exécuter. Les valeurs possibles incluent :
- DSM_GROUP_ACTION_OPEN - crée un groupe

- DSM_GROUP_ACTION_CLOSE - valide et sauvegarde un groupe ouvert
- DSM_GROUP_ACTION_ADD - ajoute à un nouveau groupe
- DSM_GROUP_ACTION_ASSIGNTO - affecte à un autre groupe
- DSM_GROUP_ACTION_REMOVE- supprime un membre d'un groupe

memberType.
Type de groupe de l'objet. Les valeurs possibles incluent :
- DSM_MEMBERTYPE_LEADER - chef de groupe
- DSM_MEMBERTYPE_MEMBER - membre de groupe

***uniqueGroupTagP**
ID chaîne unique qui est associé à un groupe.

leaderObjId
ID objet du chef de groupe.

***objNameP**
Pointeur vers le nom d'objet du chef de groupe.

memberObjList
Liste des objets à supprimer ou à affecter.

dsmGroupHandlerOut_t *dsmGroupHandlerOutP (0)
Transmet l'adresse de la structure que l'API complète. Le numéro de version de la structure est renvoyé.

Codes retour

Les numéros de code retour sont indiqués entre parenthèses ( ).

Tableau 37. Codes retour pour dsmGroupHandler

Code retour	Description
DSM_RC_ABORT_INVALID_GROUP_ACTION (237)	Une opération non valide a été tentée sur un chef de groupe ou un membre de groupe.

dsmInit

L'appel de fonction **dsmInit** démarre une session API et connecte le client à la mémoire IBM Spectrum Protect. Le client d'application peut avoir seulement une session active ouverte à la fois. Pour ouvrir une autre session avec des paramètres différents, utilisez d'abord l'appel **dsmTerminate** pour terminer la session en cours.

Pour permettre une requête et une restauration ou une récupération entre postes, utilisez les options de chaîne *-fromnode* et *-fromowner* . Pour plus d'informations, voir «Accès aux objets sur plusieurs postes et propriétaires», à la page 26.

Syntaxe

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

Paramètres

dsUint32_t *dsmHandle (O)
Descripteur qui identifie cette session d'initialisation et l'associe aux appels IBM Spectrum Protect suivants.

dsmApiVersion *dsmApiVersionP (I)
Pointeur vers la structure de données identifiant la version de l'API qui est utilisée par le client d'application pour cette session. Cette structure contient les valeurs des trois constantes, DSM_API_VERSION, DSM_API_RELEASE et DSM_API_LEVEL, définies dans le fichier dsmapitd.h. Un appel précédent à **dsmQueryApiVersion** doit être exécuté pour s'assurer qu'il y a compatibilité entre la version de l'API du client d'application et la version de la bibliothèque API qui est installée sur le poste de travail de l'utilisateur.

char *clientNodeNameP (I)
Ce paramètre est un pointeur vers le poste pour la session IBM Spectrum Protect. A chaque session doit être associée un nom de poste. La constante, DSM_MAX_NODE_LENGTH, dans le fichier dsmapitd.h définit la taille maximale autorisée pour un nom de poste.

Vous pouvez utiliser indifféremment les majuscules et les minuscules dans le nom de poste.

Si ce paramètre est défini sur NULL, et que *passwordaccess* est défini sur *prompt*, l'API tente d'obtenir d'abord le nom de poste à partir de la chaîne d'option qui a été transmise. Si l'API ne le trouve pas à cet emplacement, l'API tente de l'obtenir le nom de poste à partir du fichier de configuration ou des fichiers d'options. Si ces tentatives pour trouver le nom de poste échouent, l'API UNIX ou Linux utilise le nom d'hôte du système, tandis que les API des autres systèmes d'exploitation renvoient le code DSM_RC_REJECT_ID_UNKNOWN.

Ce paramètre doit être NULL si l'option *passwordaccess* du fichier dsm.sys est défini sur *generate*. L'API utilise le nom d'hôte du système.

char *clientOwnerNameP (I)
Ce paramètre est un pointeur vers le propriétaire de la session IBM Spectrum Protect. Si le système d'exploitation sous lequel démarre la session dans un système d'exploitation multiutilisateur, un nom de propriétaire NULL (superutilisateur) a l'autorisation de sauvegarder, archiver, restaurer ou récupérer tout objet appartenant à l'application, quel que soit le propriétaire de l'objet.

La différenciation majuscules/minuscules s'applique au nom de propriétaire.

Ce paramètre doit être NULL si l'option *passwordaccess* du fichier dsm.sys est défini sur *generate*. L'API utilise alors l'ID utilisateur de connexion.

Remarque : Sous un système d'exploitation multiutilisateur, si *passwordaccess* est définie sur *prompt*, le nom de propriétaire ne doit pas nécessairement correspondre à l'ID utilisateur actif de la session qui exécute l'application.

char *clientPasswordP (I)
Ce paramètre est un pointeur vers le mot de passe du poste sur lequel s'exécute la session IBM Spectrum Protect. La constanteDSM_MAX_VERIFIER_LENGTH présente dans le fichier dsmapitd.h définit la taille maximale autorisée pour un mot de passe.

Vous pouvez utiliser indifféremment les majuscules et les minuscules dans le mot de passe.

Excepté lors du premier démarrage du fichier de mot de passe, la valeur de ce paramètre est ignorée si *passwordaccess* est défini sur *generate*.

char *applicationType (I)
Ce paramètre identifie l'application qui exécute la session. Le client d'application définit la valeur.

Chaque fois qu'un client d'application API démarre une session avec le serveur, le type d'application (ou plateforme) du client est mis à jour sur le serveur. Il est recommandé que la valeur du type d'application contienne une abréviation du système d'exploitation, car cette valeur est entrée dans la zone **plateforme** sur le serveur. La longueur maximale de la chaîne est DSM_MAX_PLATFORM_LENGTH.

Pour afficher la valeur en cours du type d'application, appelez **dsmQuerySessInfo**.

char *configfile (I)
Ce paramètre pointe vers une chaîne de caractères qui contient le nom complet d'un fichier de configuration API. Les options indiquées dans ce fichier remplace leur spécification dans le fichier des options client. Les fichiers d'options sont définis lorsque IBM Spectrum Protect (client ou API) est installé.

char *options (I)
Pointe vers une chaîne de caractères qui peut contenir des options utilisateur telles que :

- *Compressalways*
- *Servername* (UNIX ou Linux uniquement)
- *TCPServeraddr*
- *Fromnode*
- *Fromowner*
- *EnableClientEncryptKey*

Le client d'application peut utiliser cette liste d'options pour remplacer les valeurs de ces options définies par le fichier de configuration.

Le format de ces options est :
1. Chacune des options indiquées dans cette liste d'options commence par un tiret (-) et est suivie par le mot clé de l'option.
2. Le mot clé, à son tour, est suivi d'un signe égal (=) puis du paramètre de l'option.
3. Si le paramètre de l'option contient un espace, encadrez le paramètre par des apostrophes ou des guillemets.
4. Si plusieurs options sont indiquées, séparez-les par des espaces.

Si les options sont NULL, les valeurs pour toutes les options sont extraites du fichier des options utilisateur ou du fichier de configuration API.

Codes retour

Les numéros de code retour sont indiqués entre parenthèses ( ).

Tableau 38. Codes retour pour dsmInit

Code retour	Description
DSM_RC_ABORT_SYSTEM_ERROR (1)	Le serveur a détecté une erreur système et a averti les clients.
DSM_RC_REJECT_VERIFIER_EXPIRED (52)	Le mot de passe a expiré et doit être modifié.

Tableau 38. Codes retour pour dsmInit (suite)

Code retour	Description
DSM_RC_REJECT_ID_UNKNOWN (53)	Le nom de poste est introuvable.
DSM_RC_AUTH_FAILURE (137)	Echec de l'authentification.
DSM_RC_NO_STARTING_DELIMITER (148)	Aucun délimiteur de démarrage n'a été trouvé dans le modèle.
DSM_RC_NEEDED_DIR_DELIMITER (149)	Un délimiteur de répertoire doit être présent immédiatement avant et après la métachaîne «match directories» («...») mais n'a pas été localisé.
DSM_RC_NO_PASS_FILE (168)	Le fichier des mots de passe n'est pas disponible.
DSM_RC_UNMATCHED_QUOTE (177)	Apostrophe non appariée dans la chaîne d'options.
DSM_RC_NLS_CANT_OPEN_TXT (0610)	Impossible d'ouvrir le fichier texte du message.
DSM_RC_INVALID_OPT (400)	Une entrée de la chaîne d'options est non valide.
DSM_RC_INVALID_DS_HANDLE (2014)	Descripteur DSM non valide.
DSM_RC_NO_OWNER_REQD (2032)	Le paramètre owner doit être NULL lorsque *passwordaccess* est défini sur *generate*.
DSM_RC_NO_NODE_REQD (2033)	Le paramètre node doit être NULL lorsque *passwordaccess* est défini sur *generate*.
DSM_RC_WRONG_VERSION (2064)	La version de l'API du client d'application a une valeur plus élevée que la version de IBM Spectrum Protect.
DSM_RC_PASSWD_TOOLONG (2103)	Le mot de passe indiqué est trop long.
DSM_RC_NO_OPT_FILE (2220)	Un fichier de configuration n'a pu être localisé.
DSM_RC_INVALID_KEYWORD (2221)	Un mot clé indiqué dans une chaîne d'options est non valide.
DSM_RC_PATTERN_TOO_COMPLEX (2222)	Le modèle d'inclusion-exclusion est trop complexe pour permettre l'interprétation par IBM Spectrum Protect.
DSM_RC_NO_CLOSING_BRACKET (2223)	Un crochet fermant est manquant dans le modèle.
DSM_RC_INVALID_SERVER (2225)	Pour un environnement multiutilisateur, le serveur n'a pas été trouvé dans le fichier de configuration du système.
DSM_RC_NO_HOST_ADDR (2226)	Informations insuffisantes pour effectuer la connexion à l'hôte.
DSM_RC_MACHINE_SAME (2227)	Le nom de poste défini dans le fichier des options ne peut pas être identique au nom d'hôte du système.
DSM_RC_NO_API_CONFIGFILE (2228)	Impossible d'ouvrir le fichier de configuration.
DSM_RC_NO_INCLEXCL_FILE (2229)	Fichier d'inclusion-exclusion introuvable.
DSM_RC_NO_SYS_OR_INCLEXCL (2230)	Le fichier dsm.sys ou le fichier d'inclusion-exclusion est introuvable.

Concepts associés:

- Présentation du fichier d'options client
- Options de traitement

dsmInitEx

L'appel de fonction **dsmInitEx** démarre une session API en utilisant les paramètres supplémentaires qui permettent une vérification étendue.

Syntaxe

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

Paramètres

dsUint32_t *dsmHandleP (O)
Descripteur qui identifie cette session d'initialisation et l'associe aux appels IBM Spectrum Protect suivants.

dsmInitExIn_t *dsmInitExInP
Cette structure contient les paramètres d'entrée suivants :

dsmApiVersion *dsmApiVersionP (I)
Ce paramètre est un pointeur vers la structure de données qui identifie la version de l'API que le client d'application utilise pour cette session. Cette structure contient les valeurs des quatre constantes, DSM_API_VERSION, DSM_API_RELEASE, DSM_API_LEVEL et DSM_API_SUBLEVEL définies dans le fichier dsmapitd.h. Appelez **dsmQueryApiVersionEx** et vérifiez que la version d'API du client d'application et la version de la bibliothèque d'API installée sur le poste de travail de l'utilisateur sont compatibles.

char *clientNodeNameP (I)
Ce paramètre est un pointeur vers le poste pour la session IBM Spectrum Protect. Toutes les sessions doivent être associées à un nom de poste. La constante DSM_MAX_NODE_LENGTH dans le fichier dsmapitd.h définit la taille maximale d'un nom de poste.

Vous pouvez utiliser indifféremment les majuscules et les minuscules dans le nom de poste.

Si ce paramètre est défini sur NULL, et que **passwordaccess** soit défini sur prompt, l'API tente d'obtenir d'abord le nom de poste à partir de la chaîne d'options qui a été transmise. Si l'API ne le trouve pas à cet emplacement, l'API tente de l'obtenir le nom de poste à partir du fichier de configuration ou des fichiers d'options. Si ces tentatives pour trouver le nom de poste échouent, l'API UNIX ou Linux utilise le nom d'hôte du système, tandis que les API d'autres systèmes d'exploitation renvoient le code DSM_RC_REJECT_ID_UNKNOWN.

Ce paramètre doit être NULL si l'option **passwordaccess** dans le fichier dsm.sys a pour valeur generate. L'API utilise alors le nom d'hôte du système.

char *clientOwnerNameP (I)
Ce paramètre est un pointeur vers le propriétaire de la session IBM Spectrum Protect. Si le système d'exploitation est une plateforme multiutilisateur, un nom de propriétaire NULL (le superutilisateur) a l'autorisation de sauvegarder, d'archiver, de restaurer ou de récupérer tout objet appartenant à l'application, indépendamment de l'identité du propriétaire de l'objet.

La différenciation majuscules/minuscules s'applique au nom de propriétaire.

Ce paramètre doit être NULL si l'option **passwordaccess** dans le fichier dsm.sys a pour valeur generate. L'API utilise alors l'ID utilisateur de connexion.

Conseil : Sur une plateforme multiutilisateur, si **passwordaccess** a pour valeur prompt, le nom de propriétaire ne doit pas nécessairement correspondre à l'ID utilisateur actif de la session qui exécute l'application.

char *clientPasswordP (I)
Pointeur vers le mot de passe du poste sur lequel s'exécute la session IBM Spectrum Protect. La constante DSM_MAX_VERIFIER_LENGTH dans le fichier dsmapitd.h définit la taille maximale autorisée pour un mot de passe.

Vous pouvez utiliser indifféremment les majuscules et les minuscules dans le mot de passe.

Excepté lors du premier démarrage du fichier de mot de passe, la valeur de ce paramètre est ignorée si **passwordaccess** est défini sur generate.

char *userNameP;
Pointeur vers le nom d'utilisateur d'administration qui possède les droits d'accès client pour ce poste.

char *userPasswordP;
Pointeur vers le mot de passe pour le paramètre **userName**, si une valeur est fournie.

char *applicationType (I)
Identifie l'application qui exécute la session IBM Spectrum Protect. Le client d'application identifie la valeur.

Chaque fois qu'un client d'application démarre une session avec le serveur, le type d'application (ou le système d'exploitation) du client est mis à jour sur le serveur. La valeur est entrée dans la zone **platform** sur le serveur. Vous devez envisager d'utiliser un ID système d'exploitation dans la valeur. La longueur maximale de la chaîne est définie dans la constante DSM_MAX_PLATFORM_LENGTH.

Pour afficher la valeur en cours du type d'application, appelez **dsmQuerySessInfo**.

char *configfile (I)
Pointe vers une chaîne de caractères qui contient le nom complet d'un fichier de configuration API. Les options qui sont indiquées dans ce fichier remplacent les spécifications correspondantes dans le fichier d'options client. Les fichiers d'options sont définis lorsque IBM Spectrum Protect (client ou API) est installé.

char *options (I)
Pointe vers une chaîne de caractères qui peut contenir des options utilisateur telles que :
- Compressalways
- Servername (systèmes UNIX et Linux uniquement)
- TCPServeraddr (pas pour les systèmes UNIX)
- Fromnode
- Fromowner

Le client d'application peut utiliser cette liste d'options pour écraser les valeurs de ces options définies par le fichier de configuration.

Les options sont écrites au format suivant :

1. Chacune des options indiquées dans cette liste d'options commence par un tiret (-) et est suivie par le mot clé de l'option.
2. Le mot clé est suivi d'un signe égal (=), puis du paramètre de l'option.
3. Si le paramètre d'option contient un espace, placez-le entre apostrophes ou entre guillemets.
4. Si plusieurs options sont indiquées, séparez-les par des espaces.

Si les options sont NULL, les valeurs pour toutes les options sont extraites du fichier d'options utilisateur ou du fichier de configuration API.

dirDelimiter
Délimiteur de répertoire qui est préfixé sur l'espace de fichier, les noms de haut niveau et de bas niveau. Vous devez spécifier le paramètre **dirDelimiter** uniquement si l'application remplace les valeurs par défaut du système. Dans un environnement UNIX ou Linux, la valeur par défaut est une barre oblique (/). Dans un environnement Windows, la valeur par défaut est une barre oblique inversée (\).

useUnicode
Indicateur booléen qui indique si Unicode est activé. L'indicateur **useUnicode** doit avoir pour valeur `false` pour qu'une interopérabilité multiplateforme entre des systèmes UNIX et Windows puisse être obtenue.

bCrossPlatform
Un indicateur booléen doit être défini (`bTrue`) pour qu'une interopérabilité multiplateforme entre des systèmes UNIX et Windows puisse être obtenue. Lorsque l'indicateur `bCrossPlatform` est défini, l'API s'assure que les espaces de fichier ne sont pas en Unicode et que l'application n'utilise pas Unicode. Une application Windows qui utilise Unicode n'est pas compatible avec des applications qui utilisent des encodages non Unicode. L'indicateur `bCrossPlatform` ne doit pas être défini pour une application Windows qui utilise Unicode.

UseTsmBuffers
Indique si l'élimination de la copie de la mémoire tampon est utilisée.

numTsmBuffers
Nombre de mémoires tampon lorsque `useTsmBuffers=bTrue`.

bEncryptKeyEnabled
Indique si le chiffrement avec une clé gérée par l'application est utilisé.

encryptionPasswordP
Mot de passe de chiffrement.

Restriction : Lorsque `encryptkey=save`, si une clé de chiffrement existe, la valeur qui est spécifiée dans **encryptionPasswordP** est ignorée.

dsmAppVersion *appVersionP (I)
Ce paramètre est un pointeur vers la structure de données qui identifie les informations sur la version de l'application qui démarre une session API. La structure contient les valeurs des quatre constantes, applicationVersion, applicationRelease, applicationLevel et applicationSubLevel, définies dans le fichier `tsmapitd.h`.

dsmInitExOut_t *dsmInitExOut P
Cette structure contient les paramètres de sortie.

dsUint32_t *dsmHandle (0)
Descripteur qui identifie cette session d'initialisation et l'associe aux appels d'API suivants.

infoRC
Informations supplémentaires sur le code retour. Vérifiez à la fois le code retour de la fonction et la valeur de **infoRC**. Si la valeur de **infoRC** est DSM_RC_REJECT_LASTSESS_CANCELED (69), IBM Spectrum Protect indique que l'administrateur a annulé la dernière session.

Codes retour

Les numéros de code retour sont indiqués entre parenthèses ( ).

Tableau 39. Codes retour pour **dsmInitEx**

Code retour	Description
DSM_RC_ABORT_SYSTEM_ERROR (1)	Le serveur IBM Spectrum Protect a détecté une erreur système et en a averti les clients.
DSM_RC_REJECT_VERIFIER_EXPIRED (52)	Le mot de passe a expiré et doit être mis à jour. L'appel suivant doit être **dsmChangePW** avec le descripteur renvoyé pour cet appel.
DSM_RC_REJECT_ID_UNKNOWN (53)	Le nom de poste est introuvable.
DSM_RC_TA_COMM_DOWN (103)	La liaison de communication est hors service.
DSM_RC_AUTH_FAILURE (137)	Echec de l'authentification.
DSM_RC_NO_STARTING_DELIMITER (148)	Aucun délimiteur de démarrage n'a été trouvé dans le modèle.
DSM_RC_NEEDED_DIR_DELIMITER (149)	Un délimiteur de répertoire doit être présent immédiatement avant et après la métachaîne «match directories» («...»), or, il est introuvable.
DSM_RC_NO_PASS_FILE (168)	Le fichier des mots de passe n'est pas disponible.
DSM_RC_UNMATCHED_QUOTE (177)	Apostrophe non appariée dans la chaîne d'options.
DSM_RC_NLS_CANT_OPEN_TXT (0610)	Impossible d'ouvrir le fichier texte du message.
DSM_RC_INVALID_OPT (2013)	Une entrée de la chaîne d'options est non valide.
DSM_RC_INVALID_DS_HANDLE (2014)	Descripteur DSM non valide.
DSM_RC_NO_OWNER_REQD (2032)	Le paramètre owner doit être NULL lorsque **passwordaccess** est défini sur `generate`.
DSM_RC_NO_NODE_REQD (2033)	Le paramètre Node doit être NULL lorsque le paramètre **passwordaccess** a pour valeur generate.
DSM_RC_WRONG_VERSION (2064)	La version de l'API du client d'application a une valeur plus élevée que la version de IBM Spectrum Protect.
DSM_RC_PASSWD_TOOLONG (2103)	Le mot de passe indiqué est trop long.
DSM_RC_NO_OPT_FILE (2220)	Fichier de configuration introuvable.
DSM_RC_INVALID_KEYWORD (2221)	Un mot clé indiqué dans une chaîne d'options n'est pas valide.
DSM_RC_PATTERN_TOO_COMPLEX (2222)	Le modèle d'inclusion-exclusion est trop complexe pour pouvoir être interprété par IBM Spectrum Protect.
DSM_RC_NO_CLOSING_BRACKET (2223)	Un crochet fermant est manquant dans le modèle.
DSM_RC_INVALID_SERVER (2225)	Pour un environnement multiutilisateur, le serveur n'a pas été trouvé dans le fichier de configuration du système.
DSM_RC_NO_HOST_ADDR (2226)	Informations insuffisantes pour effectuer la connexion à l'hôte.
DSM_RC_MACHINE_SAME (2227)	Le nom de poste défini dans le fichier d'options ne peut pas être identique au nom d'hôte du système.
DSM_RC_NO_API_CONFIGFILE (2228)	Impossible d'ouvrir le fichier de configuration.
DSM_RC_NO_INCLEXCL_FILE (2229)	Fichier d'inclusion-exclusion introuvable.

Tableau 39. Codes retour pour **dsmInitEx** *(suite)*

Code retour	Description
DSM_RC_NO_SYS_OR_INCLEXCL (2230)	Le fichier dsm.sys ou le fichier inclusif-exclusif est introuvable.

Concepts associés:

- Présentation du fichier d'options client
- Options de traitement

dsmLogEvent

L'appel de fonction **dsmLogEvent** consigne un message utilisateur (ANE4991 I) dans le fichier journal du serveur, dans le journal d'erreurs local ou dans les deux. Une structure de type **logInfo** est transmise dans l'appel. Cet appel doit être effectué à l'intérieur d'une session lorsque l'état est **InSession**. Vous ne devez pas l'effectuer dans le cadre d'une commande send, get ou query. Pour extraire les messages consignés sur le serveur, utilisez la commande **query actlog** par l'intermédiaire du client d'administration.

Voir le diagramme d'état récapitulatif, figure 20, à la page 79.

Syntaxe

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

Paramètres

dsUint32_t dsmHandle(I)
Descripteur qui associe cet appel à un appel **dsmInitEx** précédent.

logInfo *logInfoP (I)
Transmet le message et la destination. Le client d'application est responsable de l'allocation de mémoire à la structure.

Les zones de la structure **logInfo** sont les suivantes :

message
Texte du message à consigner. Il doit s'agir d'une chaîne terminée par un Null. La longueur maximale est DSM_MAX_RC_MSG_LENGTH.

dsmLogtype
Indique l'emplacement où doit être consigné le message. Les valeurs admises sont les suivantes : **logServer**, **logLocal**, **logBoth**.

Codes retour

Les numéros de code retour sont indiqués entre parenthèses ( ).

Tableau 40. Codes retour pour dsmLogEvent

Code retour	Description
DSM_RC_STRING_TOO_LONG (2120)	La chaîne du message est trop longue.

dsmLogEventEx

L'appel de fonction **dsmLogEventEx** consigne un message utilisateur dans le fichier journal du serveur, dans le journal d'erreurs local ou dans les deux. Cet appel doit être effectué à l'intérieur d'une session lorsque l'état est **InSession**. L'appel ne doit pas être effectué dans le cadre d'un appel send, get, ou query.

Diagramme d'état récapitulatif : Pour obtenir un aperçu des interactions de la session, reportez-vous au diagramme d'état récapitulatif de la rubrique suivante :

figure 20, à la page 79

La gravité détermine le numéro du message IBM Spectrum Protect. Pour afficher les messages consignés sur le serveur, utilisez la commande **query actlog** par l'intermédiaire du client d'administration. Utilisez l'option client IBM Spectrum Protect, `errorlogretention`, pour supprimer le fichier journal d'erreurs client si l'application génère de nombreux messages client écrits dans le journal client, `dsmLogType` doit avoir la valeur `logLocal` ou `logBoth`. Pour plus d'informations, consultez la documentation du serveur IBM Spectrum Protect.

Syntaxe

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

Paramètres

dsUint32_t dsmHandle(I)
Descripteur qui associe cet appel à un appel **dsmInitEx** précédent.

dsmLogExIn_t *dsmLogExInP
Cette structure contient les paramètres d'entrée.

dsmLogSeverity severity;
Ce paramètre correspond à la gravité de l'événement. Les valeurs possibles sont :

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

char appMsgID[8];
Ce paramètre est une chaîne qui identifie le message d'application spécifique. Le format recommandé est trois caractères suivis de quatre nombres, par exemple : `DSM0250`.

dsmLogType logType;
Ce paramètre spécifie l'emplacement vers lequel doit être dirigé l'événement. Les valeurs admises sont les suivantes :

- `logServer`
- `logLocal`
- `logBoth`

char *message;
Ce paramètre correspond au texte du message d'événement à consigner. Le texte terminée par un Null. La longueur maximale est `DSM_MAX_RC_MSG_LENGTH`.

Restriction : Les messages transmis au serveur doivent être en anglais. Les messages dans une autre langue que l'anglais ne s'affichent pas correctement.

dsmLogExOut_t *dsmLogExOutP
Cette structure contient les paramètres de sortie. Actuellement, il n'y a pas de paramètres de sortie.

Codes retour

Les numéros de code retour sont indiqués entre parenthèses ( ).

Tableau 41. Codes retour pour dsmLogEventEx

Code retour	Description
DSM_RC_STRING_TOO_LONG (2120)	La chaîne du message est trop longue.

dsmQueryAccess

L'appel de fonction **dsmQueryAccess** interroge le serveur pour connaître toutes les règles d'autorisation d'accès pour les versions de sauvegarde ou les copies archivées de vos objets. Un pointeur vers un tableau de règles d'accès est transmis vers l'appel et le tableau complet est renvoyé. Un pointeur vers le nombre de règles est transmis pour indiquer le nombre de règles contenues dans le tableau.

Il n'existe pas de codes retour spécifiques à cet appel.

Syntaxe

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

Paramètres

dsUint32_t dsmHandle (I)
Descripteur qui associe cet appel à un appel **dsmInitEx** précédent.

qryRespAccessData **accessListP (O)
Pointeur vers un tableau d'éléments qryRespAccessData attribués par la bibliothèque d'API. Chaque élément correspond à une règle d'accès. Le nombre d'éléments du tableau est renvoyé dans le paramètre **numberOfRules**. Les informations renvoyées dans chaque élément qryRespAccessData incluent les informations suivantes :

Nom	Description
ruleNumber	ID de la règle d'accès identifiant la règle à supprimer.
AccessType	Type de sauvegarde ou d'archivage.
Node	Poste sur lequel vous avez autorisé l'accès.
Owner	Utilisateur auquel vous avez accordé l'accès.
objName	Descripteurs d'espace fichier de haut niveau ou de bas niveau.

dsUint32_t *numberOfRules (O)
Renvoie le nombre de règles contenues dans le tableau accessList.

dsmQueryApiVersion

L'appel de fonction **dsmQueryApiVersion** exécute une requête concernant la version de la bibliothèque d'API auquel accède le client d'application.

Toutes les mises à jour de l'API sont effectuées dans un format garantissant la compatibilité ascendante. Les clients d'application possédant une version d'API inférieure ou égale à celle de la bibliothèque d'API sur le poste de travail de l'utilisateur final peuvent s'exécuter sans modification. Avant de continuer, notez que si l'appel **dsmQueryApiVersion** renvoie une version ou une édition de version plus ancienne que celle des clients d'application, certains appels d'API risquent d'être modifiés d'une manière non prise en charge par la version plus ancienne de l'API de l'utilisateur final.

Le numéro de version de l'API de l'application est enregistré dans le fichier d'en-tête dsmapitd.h sous forme de constantes DSM_API_VERSION, DSM_API_RELEASE et DSM_API_LEVEL.

Il n'existe pas de codes retour spécifiques à cet appel.

Syntaxe

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

Paramètres

dsmApiVersion *apiVersionP (0)
Ce paramètre est un pointeur vers la structure qui contient les composants de version, d'édition et de niveau de la bibliothèque d'API. Par exemple, si la bibliothèque est de la version 1.1.0, après renvoi à partir de l'appel, les zones de la structure contiennent les valeurs suivantes :

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

dsmQueryApiVersionEx

L'appel de fonction **dsmQueryApiVersionEx** exécute une requête portant sur la version de la bibliothèque d'API à laquelle accède le client d'application.

Toutes les mises à jour de l'API sont effectuées dans un format garantissant la compatibilité ascendante. Les clients d'application possédant une version ou une édition d'API inférieure ou égale à celle de la bibliothèque d'API sur le poste de travail de l'utilisateur final s'exécutent sans modification. Pour plus d'informations sur les exceptions relatives à la compatibilité ascendante, voir la section Récapitulatif des modifications apportées au code du fichier README_api_enu. Si l'appel **dsmQueryApiVersionEx** renvoie une version ou une édition différente de celle du client d'application, notez, avant de poursuivre, que certains appels d'API sont susceptibles d'être modifiés d'une manière non prise en charge par la version antérieure de l'API de l'utilisateur final.

Le numéro de version de l'API d'application est enregistré dans le fichier d'en-tête dsmapitd.h en tant que constantes DSM_API_VERSION, DSM_API_RELEASE, DSM_API_LEVEL et DSM_API_SUBLEVEL.

Il n'existe pas de codes retour spécifiques à cet appel.

Syntaxe

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

Paramètres

dsmApiVersionEx *apiVersionP (O)
Ce paramètre est un pointeur vers la structure qui contient les composants de version, d'édition, de niveau et de sous-niveau de la bibliothèque d'API. Par exemple, si la bibliothèque est de la version 5.5.0.0, après renvoi à partir de l'appel, les zones de la structure contiennent les valeurs suivantes :

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

dsmQueryCliOptions

L'appel de fonction **dsmQueryCliOptions** interroge les valeurs des options importantes dans les fichiers des options de l'utilisateur. Une structure de type **optStruct** est transmise dans l'appel et contient ces informations. Cet appel est effectué avant l'appel de **dsmInitEx** et il détermine la configuration avant la session.

Il n'existe aucun code retour spécifique à cet appel.

Syntaxe

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

Paramètres

optStruct *optstructP (I/O)
Ce paramètre transmet l'adresse de la structure complétée par l'API. Le client d'application est responsable de l'allocation de mémoire à la structure. En cas de renvoi correct, les informations appropriées sont entrées dans les zones de la structure.

Les informations suivantes sont renvoyées dans la structure **optStruct** :

Nom	Description
dsmiDir	Valeur de la variable DSMI_DIR d'environnement.
dsmiConfig	Fichier d'options client indiqué par la variable d'environnement DSMI_CONFIG.
serverName	Nom du serveur IBM Spectrum Protect.
commMethod	Méthode de communication sélectionnée. Voir la valeur #defines pour DSM_COMM_* dans le fichier dsmapitd.h.
serverAddress	Adresse du serveur basée sur la méthode de communication.
nodeName	Nom de poste client (machine).
compression	Cette zone fournit des informations sur l'option compression.
passwordAccess	Les valeurs sont les suivantes : *bTrue* pour generate et *bFalse* pour prompt.

Concepts associés:

Options de traitement

dsmQuerySessInfo

L'appel de fonction **dsmQuerySessInfo** lance une requête à IBM Spectrum Protect pour obtenir des informations relatives à la session indiquée dans **dsmHandle**. Une structure de type **ApiSessInfo** est transmise dans l'appel, avec toutes les informations disponibles liées à la session entrées. Cet appel est démarré après un appel **dsmInitEx** réussi.

Les informations renvoyées dans la structure **ApiSessInfo** incluent les informations suivantes :

- Informations sur le serveur : numéro de port, date et heure et type
- Valeurs par défaut client : type d'application, permissions de suppression, délimiteurs et limites de transaction
- Informations sur la session : ID de connexion et propriétaire
- Données de règles : domaine, ensemble de règles actif et délai de grâce de conservation

Pour plus d'informations sur le contenu de la structure qui est transmise et chacune de ses zones, voir Annexe B, «Fichiers source des définitions de type d'API», à la page 165.

Syntaxe

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

Paramètres

dsUint32_t dsmHandle (I)
Descripteur qui associe cet appel à un appel **dsmInitEx** précédent.

ApiSessInfo *SessInfoP (I/O)
Ce paramètre transmet l'adresse de la structure complétée par l'API. Le client d'application est responsable de l'allocation de la mémoire pour la structure et du renseignement des zones qui indiquent la version de la structure utilisée. En cas de renvoi correct, les zones de la structure sont complétées par les informations appropriées. Le paramètre **adsmServerName** est le nom qui est fourni dans la commande **define server** sur le serveur IBM Spectrum Protect. Si la valeur de la zone archiveRetentionProtection est true, le serveur est activé pour la protection de la conservation.

Codes retour

Les numéros de code retour sont indiqués entre parenthèses ( ).

Tableau 42. Codes retour pour dsmQuerySessInfo

Code retour	Description
DSM_RC_NO_SESS_BLK (2006)	Absence d'informations sur le bloc de session serveur.
DSM_RC_NO_POLICY_BLK (2007)	Aucune information disponible sur les règles du serveur.
DSM_RC_WRONG_VERSION_PARM (2065)	La version de l'API du client d'application est différente de la version de la bibliothèque de IBM Spectrum Protect.

dsmQuerySessOptions

L'appel de fonction **dsmQuerySessOptions** interroge les valeurs des options importantes qui sont valides dans la session indiquée dans **dsmHandle**. Une structure de type optStruct est transmise dans l'appel et contient ces informations.

Cet appel est démarré après un appel **dsmInitEx** réussi. Les valeurs renvoyées peuvent être différentes des valeurs renvoyées sur un appel **dsmQueryCliOptions**, en fonction des valeurs transmises à l'appel **dsmInitEx**, principalement optString et optFile. Pour toute information sur la priorité des options, voir «Analyse des fichiers de configuration et d'options», à la page 1.

Il n'existe pas de codes retour spécifiques à cet appel.

Syntaxe

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

Paramètres

dsUint32_t dsmhandle(I)
Descripteur qui associe cet appel à un appel **dsmInitEx** précédent.

optStruct *optstructP (I/O)
Ce paramètre transmet l'adresse de la structure complétée par l'API. Le client d'application est responsable de l'allocation de mémoire à la structure. En cas de renvoi correct, les zones de la structure sont complétées par les informations appropriées.

Les informations renvoyées dans la structure optStruct sont les suivantes :

Nom	Description
dsmiDir	Valeur de la variable d'environnement DSMI_DIR.
dsmiConfig	Fichier dsm.opt spécifié par la variable d'environnement DSMI_CONFIG.
serverName	Nom de la section relative au serveur IBM Spectrum Protect dans le fichier des options.
commMethod	Méthode de communication sélectionnée. Voir la valeur #defines pour DSM_COMM_* dans le fichier dsmapitd.h.
serverAddress	Adresse du serveur basée sur la méthode de communication.
nodeName	Nom du poste (machine) du client.
compression	Valeur de l'option de compression (bTrue=on et bFalse=off).
compressAlways	Valeur de l'option compressalways (bTrue=on et bFalse=off).
passwordAccess	Valeur bTrue pour generate et bFalse pour prompt.

Concepts associés:

Options de traitement

dsmRCMsg

L'appel de fonction **dsmRCMsg** permet d'obtenir le texte du message associé à un code de retour d'API.

Le paramètre **msg** affiche le code retour de préfixe de message entre parenthèses ( ), suivi du texte du message. Par exemple, un appel à **dsmRCMsg** peut renvoyer les informations suivantes :

```
ANS0264E (RC2300) Seul un superutilisateur peut exécuter dsmChangePW ou dsmDeleteFS.
```

Pour certaines langues comportant des caractères différents des pages de codes ANSII et OEM, il peut être nécessaire de convertir des chaînes d'ANSII en OEM avant de les imprimer (par exemple, les jeux de caractères codés sur un seul octet d'Europe de l'Est). Un exemple est présenté ci-dessous :

```
dsmRCMsg(dsmHangle, rc, msgBuf);
#ifdef WIN32
#ifndef WIN64
CharToOemBuff(msgBuf, msgBuf, strlen(msgBuf));
#endif
#endif
printf("
```

Syntaxe

```
dsInt16_t dsmRCMsg  (dsUint32_t       dsmHandle,
   dsInt16_t             dsmRC,
   char        *msg);
```

Paramètres

dsUint32_t dsmHandle (I)
Descripteur qui associe cet appel à un appel **dsmInitEx** précédent.

dsInt16_t dsmRC (I)
Code retour d'API du texte de message associé. Les codes retour d'API sont répertoriés dans le fichier dsmrc.h. Pour plus d'informations, voir Annexe A, «Fichier source des codes retour de l'API : dsmrc.h», à la page 155.

char *msg (O)
Ce paramètre correspond au texte de message associé au code retour, **dsmRC**. L'appelant est responsable de l'allocation d'espace suffisant pour le texte du message.

La longueur maximale pour **msg** est définie comme DSM_MAX_RC_MSG_LENGTH.

Sur les plateformes possédant le support de langue nationale et un choix de fichiers de messages de langue, l'API renvoie une chaîne de message dans la langue nationale.

Codes retour

Les numéros de code retour sont indiqués entre parenthèses ( ).

Tableau 43. Codes retour pour dsmRCMsg

Code retour	Description
DSM_RC_NULL_MSG (2002)	Le paramètre **msg** pour l'appel dsmRCMsg est un pointeur NULL.
DSM_RC_INVALID_RETCODE (2021)	Le code retour qui a été transmis à l'appel **dsmRCMsg** est un code non valide.

Tableau 43. Codes retour pour dsmRCMsg (suite)

Code retour	Description
DSM_RC_NLS_CANT_OPEN_TXT (0610)	Impossible d'ouvrir le fichier texte du message.

dsmRegisterFS

L'appel de fonction **dsmRegisterFS** enregistre un nouvel espace fichier avec le serveur IBM Spectrum Protect. Vous devez préalablement enregistrer un espace fichier afin de pouvoir y sauvegarder des données.

Les clients d'application ne doivent pas utiliser les mêmes noms d'espace fichier qu'un client de sauvegarde-archivage.

- Sous UNIX ou Linux, exécutez la commande **df** pour ces noms.
- Sous Windows, ces noms sont généralement les noms de volume qui sont associés au différents lecteurs de votre système.

Syntaxe

```
dsInt16_t dsmRegisterFS  (dsUint32_t       dsmHandle,
   regFSData     *regFilespaceP);
```

Paramètres

dsUint32_t dsmHandle (I)
Descripteur qui associe cet appel à un appel **dsmInitEx** précédent.

regFSData *regFilespaceP (I)
Ce paramètre transmet le nom de l'espace fichier et les informations associées que vous devez enregistrer auprès du serveur IBM Spectrum Protect.

Conseil : La zone *fstype* inclut le préfixe, «**API:**». Toutes les requêtes d'espace fichier affichent cette chaîne. Par exemple, si l'utilisateur transmet *myfstype* pour *fstype* dans **dsmRegisterFS**, la chaîne de valeur réelle sur le serveur est renvoyée sous la forme `API:myfstype` en cas de requête. Ce préfixe distingue les objets d'API des objets de sauvegarde-archivage.

La zone utilisable pour **fsInfo** est maintenant DSM_MAX_USER_FSINFO_LENGTH.

Codes retour

Les numéros de code retour sont indiqués entre parenthèses ( ).

Tableau 44. Codes retour pour dsmRegisterFS

Code retour	Description
DSM_RC_INVALID_FSNAME (2016)	Nom d'espace fichier non valide.
DSM_RC_INVALID_DRIVE_CHAR (2026)	L'identificateur d'unité n'est pas un caractère alphabétique.
DSM_RC_NULL_FSNAME (2027)	Nom d'espace fichier Null.
DSM_RC_FS_ALREADY_REGED (2062)	L'espace fichier est déjà enregistré.
DSM_RC_WRONG_VERSION_PARM (2065)	La version de l'API du client d'application est différente de la version de la bibliothèque de IBM Spectrum Protect.
DSM_RC_FSINFO_TOOLONG (2106)	La valeur d'espace fichier est trop longue.

dsmReleaseBuffer

La fonction **dsmReleaseBuffer** renvoie une mémoire tampon vers IBM Spectrum Protect. L'application appelle **dsmReleaseBuffer** après que **dsmGetDataEx** a été appelé, et que l'application a transféré toutes les données à partir de la mémoire tampon et est prête à les libérer. **dsmReleaseBuffer** nécessite que **dsmInitEx** soit appelé avec l'option ***UseTsmBuffers*** définie sur *btrue* et qu'une valeur différente de zéro soit fournie pour *numTsmBuffers*. **dsmReleaseBuffer** doit également être appelée si l'application est sur le point d'appeler **dsmTerminate** alors qu'elle contient encore des mémoires tampons de données.

dsmReleaseBufferSyntax

```
dsInt16_t  dsmReleaseBuffer   (releaseBufferIn_t        *dsmReleaseBufferInP,
                               releaseBufferOut_t       *dsmReleaseBufferOutP) ;
```

Paramètres

releaseBufferIn_t * dsmReleaseBufferInP (I)
Cette structure contient les paramètres d'entrée suivants.

dsUint32_t dsmHandle (I)
Descripteur qui associe cet appel à un appel **dsmInitEx** précédent.

dsUint8_t tsmBufferHandle(I)
Descripteur qui identifie cette mémoire tampon.

char *dataPtr(I)
Adresse à laquelle l'application est écrite.

Codes retour

Les numéros de code retour sont indiqués entre parenthèses ( ).

Tableau 45. Codes retour pour dsmReleaseBuffer

Code retour	Description
DSM_RC_BAD_CALL_SEQUENCE	L'appel n'a pas été émis dans l'état approprié.
DSM_RC_INVALID_TSMBUFFER	Le descripteur ou la valeur de **dataPtr** ne sont pas valides.
DSM_RC_BUFF_ARRAY_ERROR	Une erreur de tableau de mémoires tampons s'est produite.

dsmRenameObj

L'appel de fonction **dsmRenameObj** modifie le nom d'objet de haut niveau ou de bas niveau. Pour les objets de sauvegarde, il transmet le nom d'objet en cours et le transforme en nom d'objet de haut niveau ou de bas niveau. Pour les objets archivés, il transmet le nom de l'espace fichier d'objet archivé et l'ID objet en cours et le transforme en nom d'objet de haut niveau ou de bas niveau. Utilisez cet appel de fonction dans le cadre d'appels **dsmBeginTxn** et **dsmEndTxn**.

L'indicateur de fusion détermine si un nom d'objet de sauvegarde en double est fusionné avec les sauvegardes existantes. Si le nouveau nom correspond à un objet existant et que la valeur de merge est true, l'objet en cours est affecté du nouveau nom tandis que l'objet actif existant qui possédait ce nom devient la première copie inactive de l'objet. Si le nouveau nom correspond à un objet existant et que la valeur de merge est false, cette fonction renvoie le code retour, DSM_RC_ABORT_DUPLICATE_OBJECT.

Restriction : Seul le propriétaire de l'objet est autorisé à le renommer.

L'appel de fonction **dsmRenameObj** teste les conditions de fusion suivantes :

- L'objet **dsmObjName** en cours et le nouvel objet de haut niveau ou de bas niveau doivent correspondre en ce qui concerne le propriétaire, le groupe de copie ou la classe de gestion.
- Le nom **dsmObjName** en cours doit avoir été sauvegardé plus récemment que l'objet actuellement actif possédant le nouveau nom.
- Il doit exister seulement une copie active du nom **dsmObjName** en cours sans copies inactives.

Syntaxe

```
dsInt16_t dsmRenameObj  (dsmRenameIn_t       *dsmRenameInP,
                         dsmRenameOut_t      *dsmRenameOutP);
```

Paramètres

dsUint32_t dsmHandle (I)
Descripteur qui associe cet appel à un appel **dsmInitEx** précédent.

dsmRenameIn_t *dsmRenameInP
Cette structure contient les paramètres d'entrée.

dsUint8_t repository (I);
Ce paramètre indique si l'espace fichier à supprimer est contenu dans le référentiel de sauvegarde ou dans le référentiel d'archivage.

dsmObjName *objNameP (I);
Ce paramètre est un pointeur vers la structure qui contient le nom d'espace fichier, le nom d'objet de haut niveau, le nom d'objet de bas niveau et le type d'objet en cours.

char newHl [DSM_MAX_HL_LENGTH + 1];
Ce paramètre spécifie le nouveau nom de haut niveau.

char newLl [DSM_MAX_LL_LENGTH + 1];
Ce paramètre spécifie le nouveau nom de bas niveau.

dsBool_t merge;
Ce paramètre détermine si un objet de sauvegarde est fusionné avec les objets nommés en double. Les valeurs sont soit true soit false.

ObjID;
ID objet des objets archivés.

dsmRenameOut_t *dsmRnameOutP
Cette structure contient les paramètres de sortie.

Remarque : Il n'y a pas de paramètres de sortie.

Codes retour

Les numéros de code retour sont indiqués entre parenthèses ( ).

Tableau 46. Codes retour pour dsmRenameObj

Code retour	Description
DSM_RC_ABORT_MERGE_ERROR (45)	Le serveur a détecté une erreur de fusion.
DSM_RC_ABORT_DUPLICATE_OBJECT (32)	L'objet existe déjà et la valeur de merge est false.

Tableau 46. Codes retour pour dsmRenameObj (suite)

Code retour	Description
DSM_RC_ABORT_NO_MATCH (2)	Objet introuvable.
DSM_RC_REJECT_SERVER_DOWNLEVEL (58)	La version du serveur IBM Spectrum Protect doit être V3.7.4.0 ou ultérieure pour que cette fonction opère.

dsmRequestBuffer

La fonction **dsmRequestBuffer** renvoie une mémoire tampon vers IBM Spectrum Protect. L'application appelle **dsmRequestBuffer** après que **dsmGetDataEx** a été appelé, et que l'application a transféré toutes les données à partir de la mémoire tampon et est prête à les libérer.

dsmReleaseBuffer nécessite que **dsmInitEx** soit appelé avec l'option *UseTsmBuffers* définie sur *btrue* et qu'une valeur différente de zéro soit fournie pour *numTsmBuffers*. **dsmReleaseBuffer** doit également être appelée si l'application est sur le point d'appeler **dsmTerminate** et qu'elle contienne encore des mémoires tampons de IBM Spectrum Protect.

Syntaxe

```
dsInt16_t  dsmRequestBuffer   (getBufferIn_t        *dsmRequestBufferInP,
                               getBufferOut_t       *dsmRequestBufferOutP) ;
```

Paramètres

getBufferIn_t * dsmRequestBufferInP (I)
Cette structure contient les paramètres d'entrée suivants :

dsUint32_t dsmHandle
Descripteur qui identifie la session et l'associe à un appel **dsmInitEx** précédent.

getBufferOut_t *dsmRequestBufferOut P (0)
Cette structure contient les paramètres de sortie.

dsUint8_t tsmBufferHandle(0)
Descripteur qui identifie cette mémoire tampon.

char *dataPtr(0)
Adresse à laquelle l'application est écrite.

dsUint32_t *bufferLen(0)
Nombre maximal d'octets pouvant être écrits dans cette mémoire tampon.

Codes retour

Les numéros de code retour sont indiqués entre parenthèses ( ).

Tableau 47. Codes retour pour dsmRequestBuffer

Code retour	Description
DSM_RC_BAD_CALL_SEQUENCE (33)	L'appel n'a pas été émis dans l'état approprié.
DSM_RC_SENDDATA_WITH_ZERO_SIZE (34)	Si l'objet envoyé a une longueur de 0, aucun appel à **dsmReleaseBuffer** n'est autorisé.
DSM_RC_BUFF_ARRAY_ERROR (121)	Impossible d'obtenir une mémoire tampon valide.

dsmRetentionEvent

L'appel de fonction **dsmRetentionEvent** envoie une liste d'ID objet au serveur IBM Spectrum Protect, avec une opération d'événement de conservation à effectuer sur ces objets. Utilisez cet appel de fonction dans le cadre d'appels **dsmBeginTxn** et **dsmEndTxn**.

Remarque : Le serveur doit être au niveau de la version 5.2.2.0 ou d'une version ultérieure pour que cette fonction soit opérationnelle.

Le nombre maximal d'objets dans un appel est limité à la valeur de *maxObjPerTxn* qui est renvoyée dans la structure *ApisessInfo* à partir d'un appel **dsmQuerySessInfo**.

Seul un propriétaire d'un objet peut envoyer un événement sur cet objet.

Les événements suivants sont possibles :

eventRetentionActivate
Peut être émis uniquement pour les objets qui sont liés à une classe de gestion basée sur l'événement. L'envoi de cet événement active l'événement pour cet objet et l'état de la conservation pour cet objet passe de DSM_ARCH_RETINIT_PENDING à DSM_ARCH_RETINIT_STARTED.

eventHoldObj
Cet événement émet un maintien de la conservation ou de la suppression sur l'objet de sorte que, jusqu'à l'émission d'une libération, l'objet n'est pas arrivé à expiration et ne peut pas être supprimé.

eventReleaseObj
Cet événement peut être émis uniquement pour un objet pour lequel la zone ***objectHeld*** a la valeur DSM_ARCH_HELD_TRUE et supprimera le maintien sur l'objet en reprenant les règles de conservation initiales.

Avant d'envoyer ***dsmRetentionEvent***, envoyez la séquence de requête décrite dans «Interrogation du système IBM Spectrum Protect», à la page 35 pour obtenir les informations pour l'objet. L'appel à **dsmGetNextQObj** renvoie une structure de données nommée **qryRespArchiveData** pour les requêtes d'archivage. Cette structure de données contient les informations requises pour **dsmRetentionEvent**.

Syntaxe

```
extern dsInt16_t DSMLINKAGE dsmRetentionEvent(
  dsmRetentionEventIn_t          *ddsmRetentionEventInP,
  dsmRetentionEventOut_t         *dsmRetentionEventOutP
   );
```

Paramètres

dsmRetentionEventIn_t *dsmRetentionEventP
Cette structure contient les paramètres d'entrée suivants :

dsUint16_t stVersion;
Ce paramètre indique la version de la structure.

dsUint32_t dsmHandle (I)
Descripteur qui associe cet appel à un appel **dsmInitEx** précédent.

dsmEventType_t evenType (I);
Ce paramètre indique le type d'événement. Voir le début de cette section

pour plus d'informations sur la signification de ces valeurs possibles : **eventRetentionActivate**, **eventHoldObj**, **eventReleaseObj**

dsmObjList_t objList;
Ce paramètre indique la liste des ID objet à signaler.

Codes retour

Les numéros de code retour sont indiqués entre parenthèses ( ).

Tableau 48. Codes retour pour dsmRetentionEvent

Code retour	Description
DSM_RC_ABORT_NODE_NOT_AUTHORIZED (36)	Le poste ou l'utilisateur ne possède pas les droits d'accès appropriés.
DSM_RC_ABORT_TXN_LIMIT_EXCEEDED (249)	Trop d'objets dans la transaction.
DSM_RC_ABORT_OBJECT_ALREADY_HELD (250)	L'objet est déjà suspendu, impossible d'émettre une autre suspension.
DSM_RC_REJECT_SERVER_DOWNLEVEL (58)	Le serveur doit être au niveau de la version 5.2.2.0 ou d'une version ultérieure pour que cette fonction soit opérationnelle.

dsmSendBufferData

L'appel de fonction **dsmSendBufferData** envoie un flot d'octets de données à IBM Spectrum Protect via une mémoire tampon fournie dans un appel **dsmReleaseBuffer** précédent. Le client d'application peut transmettre tout type de données à enregistrer sur le serveur. Il s'agit généralement, mais pas exclusivement, de données de fichier. Vous pouvez appeler plusieurs fois **dsmSendBufferData**, si le flot d'octets de données envoyé est volumineux. Indépendamment de la réussite ou de l'échec de l'appel, la mémoire tampon est libérée.

Restriction : Quand vous utilisez l'option *useTsmBuffers*, même si un objet est inclus pour compression, il n'est pas compressé.

Syntaxe

```
dsInt16_t  dsmSendBufferData   (sendBufferDataIn_t        *dsmSendBufferDataExInP,
                                sendBufferDataOut_t       *dsmSendBufferDataOutP) ;
```

Paramètres

sendBufferDataIn_t * dsmSendBufferDataInP (I)
Cette structure contient les paramètres d'entrée suivants.

dsUint32_t dsmHandle (I)
Descripteur qui associe cet appel à un appel **dsmInitEx** précédent.

dsUint8_t tsmBufferHandle(I)
Descripteur qui identifie la mémoire tampon à envoyer.

char *dataPtr(I)
Adresse à laquelle sont écrites les données de l'application.

dsUint32_t numBytes(I)
Nombre réel d'octets écrits par l'application (doit toujours être inférieur à la valeur fournie dans **dsmReleaseBuffer**).

Codes retour

Les numéros de code retour sont indiqués entre parenthèses ( ).

Tableau 49. Codes retour pour dsmSendBufferData

Code retour	Description
DSM_RC_BAD_CALL_SEQUENCE (2041)	L'appel n'a pas été émis dans l'état approprié.
DSM_RC_INVALID_TSMBUFFER (2042)	Le descripteur ou la valeur de **dataPtr** ne sont pas valides.
DSM_RC_BUFF_ARRAY_ERROR (2045)	Une erreur de tableau de mémoires tampons s'est produite.
DSM_RC_TOO_MANY_BYTES (2043)	La valeur de ***numBytes*** est supérieure à la taille de la mémoire tampon fournie dans l'appel **dsmReleaseBuffer**.

dsmSendData

L'appel de fonction **dsmSendData** envoie un flot d'octets de données à IBM Spectrum Protect par l'intermédiaire d'un tampon. Le client d'application peut transmettre tout type de données à enregistrer sur le serveur. Il s'agit généralement, mais pas exclusivement, de données de fichier. Vous pouvez appeler plusieurs fois **dsmSendData**, si le flot d'octets de données à envoyer est volumineux.

Restriction : Le client d'application ne peut pas réutiliser la mémoire tampon qui est spécifiée dans **dsmSendData** jusqu'au retour de l'appel **dsmSendData**.

Conseil : Si IBM Spectrum Protect renvoie le code 157 (DSM_RC_WILL_ABORT), démarre un appel à **dsmEndSendObj**, puis à **dsmEndTxn** avec un vote de DSM_VOTE_COMMIT. L'application reçoit alors le code retour 2302 (DSM_RC_CHECK_REASON_CODE) et retransmet le code raison à l'utilisateur de l'application. L'utilisateur est ainsi informé des raisons pour lesquelles le serveur termine la transaction.

Syntaxe

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

Paramètres

dsUint32_t dsmHandle (I)
Descripteur qui associe cet appel à un appel **dsmInitEx** précédent.

DataBlk *dataBlkPtr (I/O)
Ce paramètre pointe vers une structure qui inclut à la fois un pointeur vers la mémoire tampon à partir de laquelle les données doivent être envoyées, et la taille de la mémoire tampon. Après renvoi, cette structure contient le nombre d'octets réellement transférés. Pour la définition du type, voir Annexe B, «Fichiers source des définitions de type d'API», à la page 165.

Codes retour

Les numéros de code retour sont indiqués entre parenthèses ( ).

Tableau 50. Codes retour pour dsmSendData

Code retour	Description
DSM_RC_NO_COMPRESS_MEMORY (154)	Mémoire disponible insuffisante pour effectuer la compression ou l'expansion des données.

Tableau 50. Codes retour pour dsmSendData (suite)

Code retour	Description
DSM_RC_COMPRESS_GREW (155)	Durant la compression, la taille des données compressées augmente par rapport à celle des données initiales.
DSM_RC_WILL_ABORT (157)	Une erreur inconnue et inattendue s'est produite, causant une interruption de la transaction.
DSM_RC_WRONG_VERSION_PARM (2065)	La version de l'API du client d'application est différente de la version de la bibliothèque de IBM Spectrum Protect.
DSM_RC_NEEDTO_ENDTXN (2070)	Il est nécessaire de terminer la transaction.
DSM_RC_OBJ_EXCLUDED (2080)	La liste inclusive-exclusive exclut cet objet.
DSM_RC_OBJ_NOBCG (2081)	L'objet ne comporte pas de groupe de paramètres de sauvegarde et ne sera pas envoyé au serveur.
DSM_RC_OBJ_NOACG (2082)	L'objet ne comporte pas de groupe de paramètres d'archivage et n'est pas envoyé au serveur.
DSM_RC_SENDDATA_WITH_ZERO_SIZE (2107)	L'objet ne peut pas envoyer les données si la valeur *sizeEstimate* est de zéro octet.

dsmSendObj

L'appel de fonction **dsmSendObj** démarre une requête pour envoyer un objet unique dans la mémoire. Plusieurs appels **dsmSendObj** et appels **dsmSendData** associés peuvent être effectués dans les limites d'une transaction pour des raisons de performance.

L'appel **dsmSendObj** traite les données relatives à l'objet comme un flot de données transmis aux mémoires tampons. Le paramètre **dataBlkPtr** de l'appel **dsmSendObj** permet au client d'application d'effectuer l'une ou l'autre des opérations suivantes :

- transmettre les données et les attributs (les attributs sont transmis par l'intermédiaire du paramètre **objAttrPtr**) de l'objet dans un seul appel ;
- indiquer une partie des données de l'objet par l'intermédiaire de l'appel **dsmSendObj** et le reste des données par le biais d'un ou de plusieurs appels **dsmSendData**.

Le client d'application a également la possibilité de spécifier uniquement les attributs dans l'appel **dsmSendObj** et les données de l'objet par le biais d'un ou de plusieurs appels à **dsmSendData**. Pour utiliser cette méthode, définissez le paramètre **dataBlkPtr** à NULL sur l'appel **dsmSendObj**.

Conseil : Pour certains types d'objet, les données du flot d'octets ne sont pas nécessairement associés aux données ; par exemple, une entrée de répertoire sans attributs étendus.

Pour que **dsmSendObj** puisse être appelé, il est nécessaire d'effectuer préalablement un appel **dsmBindMC** pour lier correctement une classe de gestion à l'objet que vous souhaitez sauvegarder ou archiver. L'API conserve cette liaison de manière à pouvoir associer la classe de gestion correcte à l'objet lorsque celui-ci est envoyé au serveur. Si vous autorisez la classe de gestion liée à un appel **dsmSendObj** à prendre sa valeur par défaut pour un type d'objet de répertoire (DSM_OBJ_DIRECTORY), cette valeur par défaut ne sera pas nécessairement la classe de gestion par défaut. C'est la classe de gestion présentant le temps de conservation le plus élevé qui est utilisée à la place. S'il existe plusieurs classes de gestion présentant ce temps de conservation, c'est la première rencontrée qui est utilisée.

Suivez toutes les données objet qui sont envoyées dans la mémoire par un appel **dsmEndSendObj**. Si vous n'avez pas de données objet à envoyer au serveur, ou si toutes les données étaient contenues dans l'appel **dsmSendObj**, vous devez démarrer un appel **dsmEndSendObj** pour pouvoir démarrer un autre appel **dsmSendObj**. Si plusieurs envois de données étaient nécessaires par l'intermédiaire de l'appel **dsmSendData**, l'appel **dsmEndSendObj** suit le dernier envoi pour indiquer la modification d'état.

Conseil : Si IBM Spectrum Protect renvoie le code 157 (DSM_RC_WILL_ABORT), démarrez un appel à **dsmEndTxn** avec un vote de DSM_VOTE_COMMIT. L'application reçoit le code retour 2302 (DSM_RC_CHECK_REASON_CODE) et retransmet le code raison à l'utilisateur de l'application. L'utilisateur est ainsi informé des raisons pour lesquelles le serveur termine la transaction.

Si le code raison est 11 (DSM_RS_ABORT_NO_REPOSIT_SPACE), il est possible que la valeur *sizeEstimate* soit trop basse pour la quantité réelle de données. L'application doit déterminer une valeur *sizeEstimate* plus exacte et renvoyer les données.

Syntaxe

```
dsInt16_t dsmSendObj  (dsUint32_t       dsmHandle,
   dsmSendType            sendType,
   void                  *sendBuff,
   dsmObjName             *objNameP,
   ObjAttr                *objAttrPtr,
   DataBlk     *dataBlkPtr);
```

Paramètres

dsUint32_t dsmHandle (I)
Descripteur qui associe cet appel à un appel **dsmInitEx** précédent.

dsmSendType sendType (I)
Ce paramètre spécifie le type d'envoi effectué. Les valeurs possibles incluent :

Nom	Description
stBackup	Objet de sauvegarde envoyé au serveur.
stArchive	Objet archivé envoyé au serveur.
stBackupMountWait	Objet de sauvegarde pour lequel vous souhaitez que le serveur attende que le périphérique approprié, par exemple une bande, soit monté.
stArchiveMountWait	Objet archivé pour lequel vous souhaitez que le serveur attende que le périphérique approprié, par exemple une bande, soit monté.

Remarque : Utilisez les types **MountWait** si l'utilisateur de votre application est susceptible d'envoyer les données sur une bande.

void *sendBuff (I)
Ce paramètre est un pointeur vers une structure qui contient d'autres informations spécifiques de la valeur **sendType** définie sur l'appel. Actuellement, seul le type **sendType** de **stArchive** possède une structure associée. Cette structure, nommée **sndArchiveData**, contient la description de l'archive.

dsmObjName *objNameP (I)
Ce paramètre est un pointeur vers la structure qui contient le nom d'espace

fichier, le nom d'objet de haut niveau, le nom d'objet de bas niveau et le type d'objet. Pour plus d'informations, voir «Noms et ID d'objet», à la page 24.

ObjAttr *objAttrPtr (I)
Ce paramètre transmet les attributs d'objet importants à l'application. Pour la définition du type, voir Annexe B, «Fichiers source des définitions de type d'API», à la page 165.

Les attributs sont :

- **owner**, qui désigne le propriétaire de l'objet. Il est important de déterminer si le propriétaire est déclaré comme un nom spécifique ou comme une chaîne vide lors de la récupération de l'objet à partir de la mémoire IBM Spectrum Protect. Pour plus d'informations, voir «Accès aux objets en tant que propriétaire de la session», à la page 26.
- **sizeEstimate**, correspond à la meilleure estimation de la taille totale de l'objet données à envoyer au serveur. Soyez aussi précis que possible pour définir cette taille, car le serveur utilise cet attribut pour gérer efficacement l'allocation d'espace et le placement de l'objet dans ses ressources de mémoire.

  Si la valeur que vous définissez est nettement inférieure au nombre réel d'octets envoyés, le serveur peut avoir des difficulté pour allouer suffisamment d'espace et terminer la transaction avec un code raison de 11 (DSM_RS_ABORT_NO_REPOSIT_SPACE).

  Remarque : La taille estimée correspond à la taille totale de l'objet données en octets.

  Les objets d'une taille inférieure à la valeur DSM_MIN_COMPRESS_SIZE ne sont pas compressés.

  Si votre objet ne comporte pas de données binaires (seulement les informations d'attribut à partir de cet appel), la valeur **sizeEstimate** doit être zéro.

  Remarque : A partir de la Version 5.1.0, la destination de copie dans une transaction n'est pas soumise à un contrôle de cohérence sur les objets d'une longueur de zéro.
- **objCompressed**, qui est une valeur booléenne qui indique si les données objet ont déjà été compressées.

  Si l'objet est compressé (object *compressed=bTrue*), IBM Spectrum Protect n'essaie pas de le recompresser. S'il n'est pas compressé, IBM Spectrum Protect décide si l'objet doit être compressé, en fonction des valeurs de l'option `compression` définie par l'administrateur et dans les sources de configuration de l'API.

  Si votre application prévoie d'utiliser une restauration ou une récupération d'objet partiel, vous ne pouvez pas compresser les données lors de leur envoi. Pour appliquer cette règle, définissez *ObjAttr.objCompressed* sur *bTrue*.
- **objInfo**, qui sauvegarde les informations relatives à un objet particulier.

  Restriction : les informations ne sont pas enregistrées ici automatiquement. Quand cet attribut est utilisé, vous devez définir l'attribut *objInfoLength* pour afficher la longueur de *objInfo*.
- **mcNameP**, qui contient le nom d'une classe de gestion qui remplace la classe de gestion obtenue à partir de **dsmBindMC**.

- **disableDeduplication** est une valeur booléenne. Lorsqu'elle est définie sur vrai (true), cet objet n'est pas dédoublonné par le client.

DataBlk *dataBlkPtr (I/O)
Ce paramètre pointe vers une structure qui inclut à la fois un pointeur vers la mémoire tampon des données à sauvegarder ou à archiver, et la taille de cette mémoire tampon. Ce paramètre s'applique uniquement à **dsmSendObj**. Si vous souhaitez commencer à envoyer des données sur un appel **dsmSendData** suivant, et non sur l'appel **dsmSendObj**, définissez le pointeur de mémoire tampon dans la structure DataBlk sur NULL. Après renvoi, cette structure contient le nombre d'octets réellement transférés. Pour la définition du type, voir Annexe B, «Fichiers source des définitions de type d'API», à la page 165.

Codes retour

Les numéros de code retour sont indiqués entre parenthèses ( ).

Tableau 51. Codes retour pour dsmSendObj

Code retour	Description
DSM_RC_NO_COMPRESS_MEMORY (154)	Mémoire disponible insuffisante pour effectuer la compression ou l'expansion des données.
DSM_RC_COMPRESS_GREW (155)	Durant la compression, la taille des données compressées augmente par rapport à celle des données initiales.
DSM_RC_WILL_ABORT (157)	Une erreur inconnue et inattendue s'est produite, causant une interruption de la transaction.
DSM_RC_TL_NOACG (186)	La classe de gestion pour ce fichier ne comporte pas de groupe de paramètres valide pour le type d'envoi.
DSM_RC_NULL_OBJNAME (2000)	Nom d'objet Null.
DSM_RC_NULL_OBJATTRPTR (2004)	Pointeur d'attribut d'objet Null.
DSM_RC_INVALID_OBJTYPE (2010)	Type d'objet non valide.
DSM_RC_INVALID_OBJOWNER (2019)	Propriétaire objet non valide.
DSM_RC_INVALID_SENDTYPE (2022)	Type d'envoi non valide.
DSM_RC_WILDCHAR_NOTALLOWED (2050)	Les caractères génériques ne sont pas autorisés.
DSM_RC_FS_NOT_REGISTERED (2061)	Espace fichier non enregistré.
DSM_RC_WRONG_VERSION_PARM (2065)	La version de l'API du client d'application est différente de la version de la bibliothèque de IBM Spectrum Protect.
DSM_RC_NEEDTO_ENDTXN (2070)	La transaction doit être terminée.
DSM_RC_OBJ_EXCLUDED (2080)	La liste inclusive-exclusive a exclu l'objet.
DSM_RC_OBJ_NOBCG (2081)	L'objet ne comporte pas de groupe de paramètres de sauvegarde et n'est pas envoyé au serveur.
DSM_RC_OBJ_NOACG (2082)	L'objet ne comporte pas de groupe de paramètres d'archivage et n'est pas envoyé au serveur.
DSM_RC_DESC_TOOLONG (2100)	Description trop longue.
DSM_RC_OBJINFO_TOOLONG (2101)	Informations sur l'objet trop longues.
DSM_RC_HL_TOOLONG (2102)	Qualificatif de haut niveau trop long.
DSM_RC_FILESPACE_TOOLONG (2104)	Nom d'espace fichier trop long.
DSM_RC_LL_TOOLONG (2105)	Qualificatif de bas niveau trop long.
DSM_RC_NEEDTO_CALL_BINDMC (2301)	**dsmBindMC** doit être appelé en premier.

dsmSetAccess

L'appel de fonction **dsmSetAccess** accorde aux autres utilisateurs et postes l'accès aux versions de sauvegarde ou aux copies d'archivage de vos objets, l'accès à tous vos objets et à un jeu sélectif. Lorsque vous accordez l'accès à un autre utilisateur, celui-ci peut interroger, restaurer ou récupérer vos fichiers. Cette commande accepte les caractères génériques pour les champs suivants : *fs*, *hl*, *ll*, *node*, *owner*.

Remarque : Vous ne pouvez pas accorder l'accès aux version de sauvegarde et aux copies d'archivage à l'aide d'une seule commande. Vous devez indiquer s'il s'agit de la sauvegarde ou de l'archivage.

Syntaxe

```
dsInt16_t DSMLINKAGE dsmSetAccess
        (dsUint32_t        dsmHandle,
         dsmSetAccessType    accessType,
         dsmObjName             *objNameP,
         char                   *node,
         char                *owner);
```

Paramètres

dsUint32_t dsmHandle (I)
Descripteur qui associe cet appel à un appel **dsmInitEx** précédent.

dsmAccessType accessType (I)
Ce paramètre spécifie le type d'objet pour lequel vous souhaitez accorder l'accès. Les valeurs possibles incluent :

Nom	Description
atBackup	Indique que l'accès est défini pour les objets de sauvegarde.
atArchive	Indique que l'accès est défini pour les objets archivés.

dsmObjName *objNameP (I)
Ce paramètre est un pointeur vers la structure qui contient le nom d'espace fichier, le nom d'objet de haut niveau et le nom d'objet de bas niveau.

Remarque : Pour spécifier tous les espaces fichier, utilisez un astérisque (*) pour le nom d'espace fichier.

char *node (I)
Ce paramètre est un pointeur vers le nom de poste pour lequel l'accès est accordé. Pour spécifier tous les postes, indiquez un astérisque (*).

char *owner (I)
Ce paramètre est un pointeur vers le nom d'utilisateur sur le poste auquel vous avez accordé l'accès. Pour spécifier tous les utilisateurs, indiquez un astérisque (*).

Codes retour

Les numéros de code retour sont indiqués entre parenthèses ( ).

Tableau 52. Code retour pour dsmSetAccess

Code retour	Description
DSM_RC_INVALID_ACCESS_TYPE (2110)	Type d'accès non valide spécifié.

Tableau 52. Code retour pour dsmSetAccess (suite)

Code retour	Description
DSM_RC_FILE_SPACE_NOT_FOUND (124)	L'espace fichier indiqué est introuvable sur le serveur.
DSM_RC_QUERY_COMM_FAILURE (2111)	Erreur de communication lors de la requête au serveur.
DSM_RC_NO_FILES_BACKUP (2112)	Aucun fichier n'a été sauvegardé pour cet espace fichier.
DSM_RC_NO_FILES_ARCHIVE (2113)	Aucun fichier n'a été archivé pour cet espace fichier.
DSM_RC_INVALID_SETACCESS (2114)	Formulation de définition d'accès non valide.

dsmSetUp

L'appel de fonction **dsmSetUp** remplace les valeurs des variables d'environnement. Appelez **dsmSetUp**avant **dsmInitEx**. Les valeurs qui ont été transmises dans la structure **envSetUp** remplacent les variables d'environnement ou les valeurs par défaut existantes. Si vous spécifiez NULL pour une zone, les valeurs sont extraites de l'environnement. Si vous ne définissez pas de valeur, les valeurs sont extraites des valeurs par défaut.

Exigences :

1. Si vous utilisez **dsmSetUp**, appelez toujours **dsmTerminate**avant**dsmCleanUp**.
2. L'instrumentation de l'API ne peut être activée que si l'indicateur de test INSTRUMENT: API est défini dans le fichier de configuration et que les appels **dsmSetUp** ou **dsmCleanUp** sont utilisés dans l'application.

Syntaxe

```
dsInt16_t DSMLINKAGE dsmSetUp
        (dsBool_t    mtFlag,
         envSetUp    *envSetUpP);
```

Paramètres

dsBool_t mtFlag (I)
Ce paramètre indique si l'API sera utilisée en mode monotâche ou multitâche. Les valeurs possibles incluent :

```
DSM_SINGLETHREAD
DSM_MULTITHREAD
```

Exigence : L'indicateur de mode multitâche doit être activé pour permettre le transfert de données hors réseau.

envSetUp *envSetUpP(I)
Ce paramètre est un pointeur vers la structure qui contient les valeurs de remplacement. Indiquez NULL si vous ne voulez pas écraser les variables d'environnement. Les zones de la structure **envSetUp** sont les suivantes :

Nom	Description
dsmiDir	Chemin de répertoire complet qui contient un fichier de message sous UNIX ou Linux. Il indique également les répertoires dsmtca et dsm.sys.
dsmiConfig	Nom complet du fichier d'options client.
dsmiLog	Chemin complet du répertoire du journal des erreurs.
argv	Transmet le nom argv[0] du programme appelant si l'application doit s'exécuter en tant qu'utilisateur autorisé. Pour plus d'informations, voir «Définition de l'option passwordaccess sur generate sans TCA», à la page 22.

Nom	Description
logName	Nom de fichier d'un journal d'erreur si l'application n'utilise pas dsierror.log.
inclExclCaseSensitive	Indique si les règles d'inclusion/exclusion sont sensibles ou insensibles à la casse. Ce paramètre peut être utilisé uniquement sous Windows ; sous les autres systèmes d'exploitation, il est ignoré.

Codes retour

Les numéros de code retour sont indiqués entre parenthèses ( ).

Tableau 53. Code retour pour dsmSetUp

Code retour	Description
DSM_RC_ACCESS_DENIED (106)	L'accès au fichier ou au répertoire spécifié est refusé.
DSM_RC_INVALID_OPT (0400)	Une option non valide a été trouvée.
DSM_RC_NO_HOST_ADDR (0405)	Le paramètre TCPSERVERADDRESS pour ce serveur n'est pas défini dans la section du nom du serveur du fichier d'options système.
DSM_RC_NO_OPT_FILE (0406)	Le fichier d'options indiqué par filename est introuvable.
DSM_RC_MACHINE_SAME (0408)	La valeur NODENAME définie dans le fichier d'options ne peut pas être identique à la valeur *HostName* du système.
DSM_RC_INVALID_SERVER (0409)	Le fichier d'options système ne contient pas l'option SERVERNAME.
DSM_RC_INVALID_KEYWORD (0410)	Un mot clé d'option non valide a été trouvé dans le fichier de configuration **dsmInitEx**, la chaîne d'options, dsm.sys ou dsm.opt.
DSM_RC_PATTERN_TOO_COMPLEX (0411)	Le modèle d'inclusion ou d'exclusion émis est trop complexe pour être interprété de manière précise par IBM Spectrum Protect.
DSM_RC_NO_CLOSING_BRACKET (0412)	La construction du modèle d'inclusion ou d'exclusion est incorrecte. Le crochet fermant est absent.
DSM_RC_NLS_CANT_OPEN_TXT (0610)	Le système ne parvient pas à ouvrir le fichier texte du message.
DSM_RC_NLS_INVALID_CNTL_REC (0612)	Le système ne parvient pas à utiliser le fichier texte du message.
DSM_RC_NOT_ADSM_AUTHORIZED (0927)	Vous devez être l'utilisateur autorisé pour avoir accès au traitement multitâche et à la commande *passwordaccess*.
DSM_RC_NO_INCLEXCL_FILE (2229)	Fichier d'inclusion-exclusion introuvable.
DSM_RC_NO_SYS_OR_INCLEXCL (2230)	Le fichier dsm.sys ou le fichier d'inclusion-exclusion est introuvable.

dsmTerminate

L'appel de fonction **dsmTerminate** termine une session avec le serveur IBM Spectrum Protect et nettoie l'environnement IBM Spectrum Protect.

Syntaxe

Il n'existe pas de codes retour spécifiques à cet appel.

```
dsInt16_t dsmTerminate  (dsUint32_t dsmHandle);
```

Paramètres

dsUint32_t dsmHandle (I)
Descripteur qui associe cet appel à un appel **dsmInitEx** précédent.

dsmUpdateFS

L'appel de fonction **dsmUpdateFS** met à jour un espace fichier de la mémoire IBM Spectrum Protect. Cette mise à jour garantit que l'administrateur dispose d'un enregistrement actualisé de votre espace fichier.

Syntaxe

```
dsInt16_t dsmUpdateFS  (dsUint32_t       dsmHandle,
   char                   *fs,
   dsmFSUpd               *fsUpdP,
   dsUint32_t      fsUpdAct);
```

Paramètres

dsUint32_t dsmHandle (I)
Descripteur qui associe cet appel à un appel **dsmInitEx** précédent.

char *fs (I)
Ce paramètre est un pointeur vers le nom de l'espace fichier.

dsmFSUpd *fsUpdP (I)
Ce paramètre est un pointeur vers la structure qui contient les zones appropriées pour la mise à jour désirée. Renseignez uniquement les zones à modifier.

dsUint32_t fsUpdAct (I)
Mappe de bits sur deux octets qui indique la zone à mettre à jour. Les masques de contrôle des données possèdent les valeurs suivantes :

- DSM_FSUPD_FSTYPE
- DSM_FSUPD_FSINFO

  Conseil : Pour les systèmes d'exploitation Windows, l'identificateur d'unité à partir de **dsmDOSAttrib** est également mis à jour lorsque **FSINFO** est sélectionné.
- DSM_FSUPD_OCCUPANCY
- DSM_FSUPD_CAPACITY
- DSM_FSUPD_BACKSTARTDATE
- DSM_FSUPD_BACKCOMPLETEDATE

Pour obtenir une description de ces masques de contrôle des données, consultez les définitions DSM_FSUPD de la rubrique suivante : Annexe B, «Fichiers source des définitions de type d'API», à la page 165.

Codes retour

Le tableau suivant répertorie les codes retour pour l'appel de fonction **dsmUpdateFS**.

Tableau 54. Codes retour pour dsmUpdateFS

Code retour	Numéro de code retour	Description
DSM_RC_FS_NOT_REGISTERED	2061	Le nom d'espace fichier n'est pas enregistré.
DSM_RC_WRONG_VERSION_PARM	2065	La version de l'API du client d'application est différente de la version de la bibliothèque de IBM Spectrum Protect.
DSM_RC_FSINFO_TOOLONG	2106	La valeur d'espace fichier est trop longue.

dsmUpdateObj

L'appel de fonction **dsmUpdateObj** met à jour les informations meta associées à une sauvegarde active ou à un objet archivé déjà présents sur le serveur. Les données binaires de l'application ne sont pas affectées. Pour mettre à jour un objet, vous devez fournir un nom spécifique sans caractère générique. Pour mettre à jour un objet archivé, définissez **dsmSendType** sur **stArchive**. Seul le dernier objet archivé nommé est mis à jour.

Vous pouvez démarrer l'appel **dsmUpdateObj** uniquement dans l'état de session ; cet appel ne peut pas être appelé dans une transaction car il exécute sa propre transaction. Vous pouvez mettre à jour uniquement un objet à la fois.

Restriction : Sous un système d'exploitation UNIX ou Linux, si vous modifiez la zone du propriétaire, vous ne pouvez ni interroger ni restaurer l'objet à moins d'être superutilisateur.

Syntaxe

```
dsInt16_t  dsmUpdateObj
   (dsUint32_t        dsmHandle,
    dsmSendType            sendType,
    void                  *sendBuff,
    dsmObjName             *objNameP,
    ObjAttr        *objAttrPtr,    /* objInfo */
    dsUint16_t        objUpdAct);    /* vecteur de bit d'action */
```

Paramètres

Les descriptions de zone sont identiques à celles contenues dans **dsmSendObj**, avec les exceptions suivantes :

dsmObjName *objNameP (I)
Vous ne pouvez pas utiliser de caractère générique.

ObjAttr *objAttrPtr (I)
La zone **objCompressed** est ignorée pour cet appel.

Les autres différences sont les suivantes :

- **owner** : si vous indiquez une nouvelle zone **owner**, le propriétaire est modifié.
- **sizeEstimate** : si vous indiquez une valeur différente de zéro, cette valeur correspond à la valeur de la quantité réelle des données envoyées, en octets. Cette valeur est stockée dans les métadonnées IBM Spectrum Protect pour une utilisation ultérieure.
- **objInfo** : cet attribut contient la nouvelle information à entrer dans la zone **objInfo**. Entrez pour **objInfoLength** la longueur de la nouvelle valeur **objInfo**.

dsUint16_t objUpdAct
Les masques de contrôle et les actions possibles pour **objUpdAct** sont les suivants :

DSM_BACKUPD_MC
Met à jour la classe de gestion pour l'objet.

DSM_BACKUPD_OBJINFO
Met à jour **objInfo**, **objInfoLength**, et **sizeEstimate**.

DSM_BACKUPD_OWNER
Met à jour le propriétaire de l'objet.

DSM_ARCHUPD_DESCR
Met à jour la zone **Description**. Entrez la valeur de la nouvelle description par l'intermédiaire du paramètre **SendBuff**. Voir l'exemple de programme pour une utilisation correcte.

DSM_ARCHUPD_OBJINFO
Met à jour **objInfo**, **objInfoLength**, et **sizeEstimate**.

DSM_ARCHUPD_OWNER
Met à jour le propriétaire de l'objet.

Codes retour

Les numéros de code retour sont indiqués entre parenthèses ( ).

Tableau 55. Codes retour pour dsmUpdateObj

Code retour	Description
DSM_RC_INVALID_ ACTION (2232)	Action non valide.
DSM_RC_FS_NOT_REGISTERED (2061)	Espace fichier non enregistré.
DSM_RC_BAD_CALL_SEQUENCE (2041)	Séquence d'appels non valide.
DSM_RC_WILDCHAR_NOTALLOWED (2050)	Les caractères génériques ne sont pas autorisés.
DSM_RC_ABORT_NO_MATCH (2)	La requête précédente ne correspond pas.

dsmUpdateObjEx

L'appel de fonction **dsmUpdateObjEx** met à jour les méta-informations associées aux objets sauvegardés ou archivés sur le serveur. Les données binaires de l'application ne sont pas affectées. Pour mettre à jour un objet, vous devez fournir un nom sans caractère générique ou vous pouvez spécifier un ID objet pour mettre à jour un objet archivé spécifique. Vous ne pouvez pas utiliser de caractères génériques pour le nom. Pour mettre à jour un objet sauvegardé, définissez le paramètre **dsmSendType** sur **stBackup**. Pour mettre à jour un objet archivé, définissez le paramètre **dsmSendType** sur **stArchive**.

Vous pouvez démarrer l'appel **dsmUpdateObjEx** uniquement dans l'état de session ; cet appel ne peut pas être appelé dans une transaction car il exécute sa propre transaction. Vous ne pouvez mettre à jour qu'un objet à la fois.

Restriction : Sous un système d'exploitation UNIX ou Linux, si vous modifiez la zone du propriétaire, vous ne pouvez ni interroger ni restaurer l'objet à moins d'être superutilisateur. Seule la version active actuelle d'un objet sauvegardé peut être mise à jour.

Syntaxe

```
dsInt16_t  dsmUpdateObjEx
   (dsmUpdateObjExIn_t   *dsmUpdateObjExInP,
    dsmUpdateObjExOut_t  *dsmUpdateObjExOutP);
```

Paramètres

dsmUpdateObjExIn_t *dsmUpdateObjExInP
Cette structure contient les paramètres d'entrée suivants :

dsUint16_t stVersion (I)
Version de la structure actuellement utilisée.

dsUint32_t dsmHandle (I)
Descripteur qui associe cet appel à un appel **dsmInitEx** précédent.

dsmSendType sendType (I)
Type d'envoi effectué. La valeur peut être :

stBackup
Objet de sauvegarde envoyé au serveur.

stArchive
Objet archivé envoyé au serveur.

dsmObjName *objNameP (I)
Pointeur vers la structure qui contient le nom d'espace fichier, le nom d'objet de haut niveau, le nom d'objet de bas niveau et le type d'objet. Vous ne pouvez pas utiliser de caractère générique.

ObjAttr *objAttrPtr (I)
Transmet les attributs d'objet à l'application. Les valeurs mises à jour dépendent des options dans la zone **objUpdAct**. L'attribut **objCompressed** est ignoré pour cet appel.

Les attributs sont :

- **owner** change le propriétaire en cas de saisie d'un nouveau nom.
- **sizeEstimate** correspond au volume actuel de données envoyées en octets. Cette valeur est stockée dans les métadonnées IBM Spectrum Protect pour une utilisation ultérieure.
- **objCompressed**, qui est une valeur booléenne qui indique si les données objet ont déjà été compressées.
- **objInfo** est un attribut contenant la nouvelle information à entrer dans la zone **objInfo**. Indiquez la longueur de la nouvelle valeur **objInfo** dans **objInfoLength**.
- **mcNameP**, qui contient le nom d'une classe de gestion qui remplace la classe de gestion obtenue à partir de **dsmBindMC**.

dsUint32_t objUpdAct
Spécifie les masques de contrôle des données et les actions pour **objUpdAct**, soit :

DSM_BACKUPD_MC
Met à jour la classe de gestion pour l'objet.

DSM_BACKUPD_OBJINFO
Met à jour l'information (**objInfo**), la longueur de l'information (**objInfoLength**) et le volume de données envoyées (**sizeEstimate**) pour l'objet sauvegardé.

DSM_BACKUPD_OWNER
Met à jour le propriétaire de l'objet sauvegardé.

DSM_ARCHUPD_DESCR
Met à jour la zone **Description** pour l'objet archivé. Entrez la valeur de la nouvelle description via le paramètre **sendBuff**.

DSM_ARCHUPD_OBJINFO
Met à jour l'information (**objInfo**), la longueur de l'information (**objInfoLength**) et le volume de données envoyées (**sizeEstimate**) pour l'objet archivé.

DSM_ARCHUPD_OWNER
Met à jour le propriétaire de l'objet archivé.

ObjID archObjId
Spécifie l'ID objet unique d'un objet archivé spécifique. Etant donné que plusieurs objets archivés peuvent porter le même nom, ce paramètre identifie un objet spécifique. Vous pouvez obtenir l'ID objet à l'aide d'un appel d'interrogation d'archive.

dsmUpdateObjExOut_t *dsmUpdateObjExOutP
Cette structure contient le paramètre de sortie :

dsUint16_t stVersion (I)
Version de la structure actuellement utilisée.

Codes retour

Les numéros de code retour sont indiqués entre parenthèses ( ) dans le tableau suivant.

Tableau 56. Codes retour pour dsmUpdateObjEx

Code retour	Explication
DSM_RC_INVALID_ACTION (2012)	Action non valide.
DSM_RC_FS_NOT_REGISTERED (2061)	Espace fichier non enregistré.
DSM_RC_BAD_CALL_SEQUENCE (2041)	Séquence d'appels non valide.
DSM_RC_WILDCHAR_NOTALLOWED (2050)	Les caractères génériques ne sont pas autorisés.
DSM_RC_ABORT_NO_MATCH (2)	La requête précédente ne correspond pas.

Annexe A. Fichier source des codes retour de l'API : dsmrc.h

Le fichier d'en-tête dsmrc.h contient tous les codes retour que l'API peut renvoyer à une application.

Les informations fournies ici contiennent une copie ponctuelle du fichier dsmrc.h distribué avec l'API. Affichez le fichier dans l'API Distribution Package pour la version la plus récente.

```
/*****************************************************************
* Tivoli Storage Manager                                *
* API Client Component                                  *
*                                                                *
* (C) Copyright IBM Corporation 1993,2010                        *
*****************************************************************/

/*****************************************************************/
/* Header File Name:  dsmrc.h                                    */
/*                                                               */
/* Descriptive-name:  Return codes from Tivoli Storage Manager APIs   */
/*****************************************************************/
#ifndef _H_DSMRC
#define _H_DSMRC


#ifndef DSMAPILIB

#ifndef _H_ANSMACH
typedef int RetCode ;
#endif

#endif


#define DSM_RC_SUCCESSFUL               0 /* successful completion   */
#define DSM_RC_OK                       0 /* successful completion   */

#define DSM_RC_UNSUCCESSFUL            -1 /* unsuccessful completion */

/* dsmEndTxn reason code */
#define DSM_RS_ABORT_SYSTEM_ERROR           1
#define DSM_RS_ABORT_NO_MATCH               2
#define DSM_RS_ABORT_BY_CLIENT              3
#define DSM_RS_ABORT_ACTIVE_NOT_FOUND       4
#define DSM_RS_ABORT_NO_DATA                5
#define DSM_RS_ABORT_BAD_VERIFIER           6
#define DSM_RS_ABORT_NODE_IN_USE            7
#define DSM_RS_ABORT_EXPDATE_TOO_LOW        8
#define DSM_RS_ABORT_DATA_OFFLINE           9
#define DSM_RS_ABORT_EXCLUDED_BY_SIZE       10
#define DSM_RS_ABORT_NO_STO_SPACE_SKIP      11
#define DSM_RS_ABORT_NO_REPOSIT_SPACE       DSM_RS_ABORT_NO_STO_SPACE_SKIP
#define DSM_RS_ABORT_MOUNT_NOT_POSSIBLE     12
#define DSM_RS_ABORT_SIZESTIMATE_EXCEED     13
#define DSM_RS_ABORT_DATA_UNAVAILABLE       14
#define DSM_RS_ABORT_RETRY                  15
#define DSM_RS_ABORT_NO_LOG_SPACE           16
#define DSM_RS_ABORT_NO_DB_SPACE            17
#define DSM_RS_ABORT_NO_MEMORY              18

#define DSM_RS_ABORT_FS_NOT_DEFINED         20
#define DSM_RS_ABORT_NODE_ALREADY_DEFED     21
#define DSM_RS_ABORT_NO_DEFAULT_DOMAIN      22
#define DSM_RS_ABORT_INVALID_NODENAME       23
#define DSM_RS_ABORT_INVALID_POL_BIND       24
#define DSM_RS_ABORT_DEST_NOT_DEFINED       25
#define DSM_RS_ABORT_WAIT_FOR_SPACE         26
#define DSM_RS_ABORT_NOT_AUTHORIZED         27
#define DSM_RS_ABORT_RULE_ALREADY_DEFED     28
#define DSM_RS_ABORT_NO_STOR_SPACE_STOP     29

#define DSM_RS_ABORT_LICENSE_VIOLATION      30
```

```
#define DSM_RS_ABORT_EXTOBJID_ALREADY_EXISTS 31
#define DSM_RS_ABORT_DUPLICATE_OBJECT        32

#define DSM_RS_ABORT_INVALID_OFFSET          33    /* Partial Object Retrieve */
#define DSM_RS_ABORT_INVALID_LENGTH          34    /* Partial Object Retrieve */
#define DSM_RS_ABORT_STRING_ERROR            35
#define DSM_RS_ABORT_NODE_NOT_AUTHORIZED     36
#define DSM_RS_ABORT_RESTART_NOT_POSSIBLE    37
#define DSM_RS_ABORT_RESTORE_IN_PROGRESS     38
#define DSM_RS_ABORT_SYNTAX_ERROR            39

#define DSM_RS_ABORT_DATA_SKIPPED            40
#define DSM_RS_ABORT_EXCEED_MAX_MP           41
#define DSM_RS_ABORT_NO_OBJSET_MATCH         42
#define DSM_RS_ABORT_PVR_ERROR               43
#define DSM_RS_ABORT_BAD_RECOGTOKEN          44
#define DSM_RS_ABORT_MERGE_ERROR             45
#define DSM_RS_ABORT_FSRENAME_ERROR          46
#define DSM_RS_ABORT_INVALID_OPERATION       47
#define DSM_RS_ABORT_STGPOOL_UNDEFINED       48
#define DSM_RS_ABORT_INVALID_DATA_FORMAT     49
#define DSM_RS_ABORT_DATAMOVER_UNDEFINED     50

#define DSM_RS_ABORT_INVALID_MOVER_TYPE      231
#define DSM_RS_ABORT_ITEM_IN_USE             232
#define DSM_RS_ABORT_LOCK_CONFLICT           233
#define DSM_RS_ABORT_SRV_PLUGIN_COMM_ERROR   234
#define DSM_RS_ABORT_SRV_PLUGIN_OS_ERROR     235
#define DSM_RS_ABORT_CRC_FAILED              236
#define DSM_RS_ABORT_INVALID_GROUP_ACTION    237
#define DSM_RS_ABORT_DISK_UNDEFINED          238
#define DSM_RS_ABORT_BAD_DESTINATION         239
#define DSM_RS_ABORT_DATAMOVER_NOT_AVAILABLE 240
#define DSM_RS_ABORT_STGPOOL_COPY_CONT_NO    241
#define DSM_RS_ABORT_RETRY_SINGLE_TXN        242
#define DSM_RS_ABORT_TOC_CREATION_FAIL       243
#define DSM_RS_ABORT_TOC_LOAD_FAIL           244
#define DSM_RS_ABORT_PATH_RESTRICTED         245
#define DSM_RS_ABORT_NO_LANFREE_SCRATCH      246
#define DSM_RS_ABORT_INSERT_NOT_ALLOWED      247
#define DSM_RS_ABORT_DELETE_NOT_ALLOWED      248
#define DSM_RS_ABORT_TXN_LIMIT_EXCEEDED      249
#define DSM_RS_ABORT_OBJECT_ALREADY_HELD     250
#define DSM_RS_ABORT_INVALID_CHUNK_REFERENCE 254
#define DSM_RS_ABORT_DESTINATION_NOT_DEDUP   255
#define DSM_RS_ABORT_DESTINATION_POOL_CHANGED 257
#define DSM_RS_ABORT_NOT_ROOT                258


/* RETURN CODE */

#define DSM_RC_ABORT_SYSTEM_ERROR            DSM_RS_ABORT_SYSTEM_ERROR
#define DSM_RC_ABORT_NO_MATCH                DSM_RS_ABORT_NO_MATCH
#define DSM_RC_ABORT_BY_CLIENT               DSM_RS_ABORT_BY_CLIENT
#define DSM_RC_ABORT_ACTIVE_NOT_FOUND        DSM_RS_ABORT_ACTIVE_NOT_FOUND
#define DSM_RC_ABORT_NO_DATA                 DSM_RS_ABORT_NO_DATA
#define DSM_RC_ABORT_BAD_VERIFIER            DSM_RS_ABORT_BAD_VERIFIER
#define DSM_RC_ABORT_NODE_IN_USE             DSM_RS_ABORT_NODE_IN_USE
#define DSM_RC_ABORT_EXPDATE_TOO_LOW         DSM_RS_ABORT_EXPDATE_TOO_LOW
#define DSM_RC_ABORT_DATA_OFFLINE            DSM_RS_ABORT_DATA_OFFLINE
#define DSM_RC_ABORT_EXCLUDED_BY_SIZE        DSM_RS_ABORT_EXCLUDED_BY_SIZE

#define DSM_RC_ABORT_NO_REPOSIT_SPACE        DSM_RS_ABORT_NO_STO_SPACE_SKIP
#define DSM_RC_ABORT_NO_STO_SPACE_SKIP       DSM_RS_ABORT_NO_STO_SPACE_SKIP

#define DSM_RC_ABORT_MOUNT_NOT_POSSIBLE      DSM_RS_ABORT_MOUNT_NOT_POSSIBLE
#define DSM_RC_ABORT_SIZESTIMATE_EXCEED      DSM_RS_ABORT_SIZESTIMATE_EXCEED
#define DSM_RC_ABORT_DATA_UNAVAILABLE        DSM_RS_ABORT_DATA_UNAVAILABLE
#define DSM_RC_ABORT_RETRY                   DSM_RS_ABORT_RETRY
#define DSM_RC_ABORT_NO_LOG_SPACE            DSM_RS_ABORT_NO_LOG_SPACE
#define DSM_RC_ABORT_NO_DB_SPACE             DSM_RS_ABORT_NO_DB_SPACE
#define DSM_RC_ABORT_NO_MEMORY               DSM_RS_ABORT_NO_MEMORY

#define DSM_RC_ABORT_FS_NOT_DEFINED          DSM_RS_ABORT_FS_NOT_DEFINED
#define DSM_RC_ABORT_NODE_ALREADY_DEFED      DSM_RS_ABORT_NODE_ALREADY_DEFED
#define DSM_RC_ABORT_NO_DEFAULT_DOMAIN       DSM_RS_ABORT_NO_DEFAULT_DOMAIN
#define DSM_RC_ABORT_INVALID_NODENAME        DSM_RS_ABORT_INVALID_NODENAME
#define DSM_RC_ABORT_INVALID_POL_BIND        DSM_RS_ABORT_INVALID_POL_BIND
#define DSM_RC_ABORT_DEST_NOT_DEFINED        DSM_RS_ABORT_DEST_NOT_DEFINED
```

```
#define DSM_RC_ABORT_WAIT_FOR_SPACE          DSM_RS_ABORT_WAIT_FOR_SPACE
#define DSM_RC_ABORT_NOT_AUTHORIZED          DSM_RS_ABORT_NOT_AUTHORIZED
#define DSM_RC_ABORT_RULE_ALREADY_DEFED      DSM_RS_ABORT_RULE_ALREADY_DEFED
#define DSM_RC_ABORT_NO_STOR_SPACE_STOP      DSM_RS_ABORT_NO_STOR_SPACE_STOP

#define DSM_RC_ABORT_LICENSE_VIOLATION       DSM_RS_ABORT_LICENSE_VIOLATION
#define DSM_RC_ABORT_EXTOBJID_ALREADY_EXISTS DSM_RS_ABORT_EXTOBJID_ALREADY_EXISTS
#define DSM_RC_ABORT_DUPLICATE_OBJECT        DSM_RS_ABORT_DUPLICATE_OBJECT

#define DSM_RC_ABORT_INVALID_OFFSET          DSM_RS_ABORT_INVALID_OFFSET
#define DSM_RC_ABORT_INVALID_LENGTH          DSM_RS_ABORT_INVALID_LENGTH

#define DSM_RC_ABORT_STRING_ERROR            DSM_RS_ABORT_STRING_ERROR
#define DSM_RC_ABORT_NODE_NOT_AUTHORIZED     DSM_RS_ABORT_NODE_NOT_AUTHORIZED
#define DSM_RC_ABORT_RESTART_NOT_POSSIBLE    DSM_RS_ABORT_RESTART_NOT_POSSIBLE
#define DSM_RC_ABORT_RESTORE_IN_PROGRESS     DSM_RS_ABORT_RESTORE_IN_PROGRESS
#define DSM_RC_ABORT_SYNTAX_ERROR            DSM_RS_ABORT_SYNTAX_ERROR

#define DSM_RC_ABORT_DATA_SKIPPED            DSM_RS_ABORT_DATA_SKIPPED
#define DSM_RC_ABORT_EXCEED_MAX_MP           DSM_RS_ABORT_EXCEED_MAX_MP
#define DSM_RC_ABORT_NO_OBJSET_MATCH         DSM_RS_ABORT_NO_OBJSET_MATCH
#define DSM_RC_ABORT_PVR_ERROR               DSM_RS_ABORT_PVR_ERROR
#define DSM_RC_ABORT_BAD_RECOGTOKEN          DSM_RS_ABORT_BAD_RECOGTOKEN
#define DSM_RC_ABORT_MERGE_ERROR             DSM_RS_ABORT_MERGE_ERROR
#define DSM_RC_ABORT_FSRENAME_ERROR          DSM_RS_ABORT_FSRENAME_ERROR
#define DSM_RC_ABORT_INVALID_OPERATION       DSM_RS_ABORT_INVALID_OPERATION
#define DSM_RC_ABORT_STGPOOL_UNDEFINED       DSM_RS_ABORT_STGPOOL_UNDEFINED
#define DSM_RC_ABORT_INVALID_DATA_FORMAT     DSM_RS_ABORT_INVALID_DATA_FORMAT
#define DSM_RC_ABORT_DATAMOVER_UNDEFINED     DSM_RS_ABORT_DATAMOVER_UNDEFINED

#define DSM_RC_ABORT_INVALID_MOVER_TYPE      DSM_RS_ABORT_INVALID_MOVER_TYPE
#define DSM_RC_ABORT_ITEM_IN_USE             DSM_RS_ABORT_ITEM_IN_USE
#define DSM_RC_ABORT_LOCK_CONFLICT           DSM_RS_ABORT_LOCK_CONFLICT
#define DSM_RC_ABORT_SRV_PLUGIN_COMM_ERROR   DSM_RS_ABORT_SRV_PLUGIN_COMM_ERROR
#define DSM_RC_ABORT_SRV_PLUGIN_OS_ERROR     DSM_RS_ABORT_SRV_PLUGIN_OS_ERROR
#define DSM_RC_ABORT_CRC_FAILED              DSM_RS_ABORT_CRC_FAILED
#define DSM_RC_ABORT_INVALID_GROUP_ACTION    DSM_RS_ABORT_INVALID_GROUP_ACTION
#define DSM_RC_ABORT_DISK_UNDEFINED          DSM_RS_ABORT_DISK_UNDEFINED
#define DSM_RC_ABORT_BAD_DESTINATION         DSM_RS_ABORT_BAD_DESTINATION
#define DSM_RC_ABORT_DATAMOVER_NOT_AVAILABLE DSM_RS_ABORT_DATAMOVER_NOT_AVAILABLE
#define DSM_RC_ABORT_STGPOOL_COPY_CONT_NO    DSM_RS_ABORT_STGPOOL_COPY_CONT_NO
#define DSM_RC_ABORT_RETRY_SINGLE_TXN        DSM_RS_ABORT_RETRY_SINGLE_TXN
#define DSM_RC_ABORT_TOC_CREATION_FAIL       DSM_RS_ABORT_TOC_CREATION_FAIL
#define DSM_RC_ABORT_TOC_LOAD_FAIL           DSM_RS_ABORT_TOC_LOAD_FAIL
#define DSM_RC_ABORT_PATH_RESTRICTED         DSM_RS_ABORT_PATH_RESTRICTED
#define DSM_RC_ABORT_NO_LANFREE_SCRATCH      DSM_RS_ABORT_NO_LANFREE_SCRATCH
#define DSM_RC_ABORT_INSERT_NOT_ALLOWED      DSM_RS_ABORT_INSERT_NOT_ALLOWED
#define DSM_RC_ABORT_DELETE_NOT_ALLOWED      DSM_RS_ABORT_DELETE_NOT_ALLOWED
#define DSM_RC_ABORT_TXN_LIMIT_EXCEEDED      DSM_RS_ABORT_TXN_LIMIT_EXCEEDED
#define DSM_RC_ABORT_OBJECT_ALREADY_HELD     DSM_RS_ABORT_OBJECT_ALREADY_HELD
#define DSM_RC_ABORT_INVALID_CHUNK_REFERENCE DSM_RS_ABORT_INVALID_CHUNK_REFERENCE
#define DSM_RC_ABORT_DESTINATION_NOT_DEDUP   DSM_RS_ABORT_DESTINATION_NOT_DEDUP
#define DSM_RC_ABORT_DESTINATION_POOL_CHANGED DSM_RS_ABORT_DESTINATION_POOL_CHANGED
#define DSM_RC_ABORT_NOT_ROOT                DSM_RS_ABORT_NOT_ROOT


/* Definitions for server signon reject codes                          */
/* These error codes are in the range (51 to 99) inclusive.            */
#define DSM_RC_REJECT_NO_RESOURCES           51
#define DSM_RC_REJECT_VERIFIER_EXPIRED       52
#define DSM_RC_REJECT_ID_UNKNOWN             53
#define DSM_RC_REJECT_DUPLICATE_ID           54
#define DSM_RC_REJECT_SERVER_DISABLED        55
#define DSM_RC_REJECT_CLOSED_REGISTER        56
#define DSM_RC_REJECT_CLIENT_DOWNLEVEL       57
#define DSM_RC_REJECT_SERVER_DOWNLEVEL       58
#define DSM_RC_REJECT_ID_IN_USE              59
#define DSM_RC_REJECT_ID_LOCKED              61
#define DSM_RC_SIGNONREJECT_LICENSE_MAX      62
#define DSM_RC_REJECT_NO_MEMORY              63
#define DSM_RC_REJECT_NO_DB_SPACE            64
#define DSM_RC_REJECT_NO_LOG_SPACE           65
#define DSM_RC_REJECT_INTERNAL_ERROR         66
#define DSM_RC_SIGNONREJECT_INVALID_CLI      67 /* client type not licensed */
#define DSM_RC_CLIENT_NOT_ARCHRETPROT        68
#define DSM_RC_REJECT_LASTSESS_CANCELED      69
#define DSM_RC_REJECT_UNICODE_NOT_ALLOWED    70
#define DSM_RC_REJECT_NOT_AUTHORIZED         71
#define DSM_RC_REJECT_TOKEN_TIMEOUT          72
```

```
#define DSM_RC_REJECT_INVALID_NODE_TYPE      73
#define DSM_RC_REJECT_INVALID_SESSIONINIT    74
#define DSM_RC_REJECT_WRONG_PORT             75
#define DSM_RC_CLIENT_NOT_SPMRETPROT         79



#define DSM_RC_USER_ABORT          101 /* processing aborted by user        */
#define DSM_RC_NO_MEMORY           102 /* no RAM left to complete request   */
#define DSM_RC_TA_COMM_DOWN       2021 /* no longer used                    */
#define DSM_RC_FILE_NOT_FOUND      104 /* specified file not found          */
#define DSM_RC_PATH_NOT_FOUND      105 /* specified path doesn't exist      */
#define DSM_RC_ACCESS_DENIED       106 /* denied due to improper permission */
#define DSM_RC_NO_HANDLES          107 /* no more file handles available    */
#define DSM_RC_FILE_EXISTS         108 /* file already exists               */
#define DSM_RC_INVALID_PARM        109 /* invalid parameter passed. CRITICAL*/
#define DSM_RC_INVALID_HANDLE      110 /* invalid file handle passed        */
#define DSM_RC_DISK_FULL           111 /* out of disk space                 */
#define DSM_RC_PROTOCOL_VIOLATION  113 /* call protocol violation. CRITICAL */
#define DSM_RC_UNKNOWN_ERROR       114 /* unknown system error. CRITICAL */
#define DSM_RC_UNEXPECTED_ERROR    115 /* unexpected error. CRITICAL */
#define DSM_RC_FILE_BEING_EXECUTED 116 /* No write is allowed               */
#define DSM_RC_DIR_NO_SPACE        117 /* directory can't be expanded       */
#define DSM_RC_LOOPED_SYM_LINK     118 /* too many symbolic links were
                                          encountered in translating path.  */
#define DSM_RC_FILE_NAME_TOO_LONG  119 /* file name too long                */
#define DSM_RC_FILE_SPACE_LOCKED   120 /* filespace is locked by the system */
#define DSM_RC_FINISHED            121 /* finished processing               */
#define DSM_RC_UNKNOWN_FORMAT      122 /* unknown format                    */
#define DSM_RC_NO_AUTHORIZATION    123 /* server response when the client has
                                          no authorization to read another
                                          host's owner backup/archive data  */
#define DSM_RC_FILE_SPACE_NOT_FOUND 124/* specified file space not found    */
#define DSM_RC_TXN_ABORTED         125 /* transaction aborted               */
#define DSM_RC_SUBDIR_AS_FILE      126 /* Subdirectory name exists as file  */
#define DSM_RC_PROCESS_NO_SPACE    127 /* process has no more disk space.   */
#define DSM_RC_PATH_TOO_LONG       128 /* a directory path being build became
                                          too long                          */
#define DSM_RC_NOT_COMPRESSED      129 /* file thought to be compressed is
                                          actually not                      */
#define DSM_RC_TOO_MANY_BITS       130 /* file was compressed using more bits
                                          then the expander can handle      */
#define DSM_RC_SYSTEM_ERROR        131 /* internal system error             */
#define DSM_RC_NO_SERVER_RESOURCES 132 /* server out of resources.          */
#define DSM_RC_FS_NOT_KNOWN        133 /* the file space is not known by the
                                          server                            */
#define DSM_RC_NO_LEADING_DIRSEP   134 /* no leading directory separator    */
#define DSM_RC_WILDCARD_DIR        135 /* wildcard character in directory
                                          path when not allowed             */
#define DSM_RC_COMM_PROTOCOL_ERROR 136 /* communications protocol error     */
#define DSM_RC_AUTH_FAILURE        137 /* authentication failure            */
#define DSM_RC_TA_NOT_VALID        138 /* TA not a root and/or SUID program */
#define DSM_RC_KILLED              139 /* process killed.                   */

#define DSM_RC_RETRY               143 /* retry same operation again        */

#define DSM_RC_WOULD_BLOCK         145 /* operation would cause the system to
                                          block waiting for input.          */
#define DSM_RC_TOO_SMALL           146 /* area for compiled pattern small   */
#define DSM_RC_UNCLOSED            147 /* no closing bracket in pattern     */
#define DSM_RC_NO_STARTING_DELIMITER 148 /* pattern has to start with
                                            directory delimiter             */
#define DSM_RC_NEEDED_DIR_DELIMITER 149 /* a directory delimiter is needed
                                          immediately before and after the
                                          "match directories" metastring
                                          ("...") and one wasn't found      */
#define DSM_RC_UNKNOWN_FILE_DATA_TYPE 150 /* structured file data type is
                                             unknown                        */
#define DSM_RC_BUFFER_OVERFLOW     151 /* data buffer overflow              */


#define DSM_RC_NO_COMPRESS_MEMORY  154 /* Compress/Expand out of memory     */
#define DSM_RC_COMPRESS_GREW       155 /* Compression grew                  */
#define DSM_RC_INV_COMM_METHOD     156 /* Invalid comm method specified     */
#define DSM_RC_WILL_ABORT          157 /* Transaction will be aborted       */
#define DSM_RC_FS_WRITE_LOCKED     158 /* File space is write locked        */
#define DSM_RC_SKIPPED_BY_USER     159 /* User wanted file skipped in the
                                          case of ABORT_DATA_OFFLINE        */
#define DSM_RC_TA_NOT_FOUND        160 /* TA not found in it's directory    */
```

```
#define DSM_RC_TA_ACCESS_DENIED    161 /* Access to TA is denied           */
#define DSM_RC_FS_NOT_READY        162 /* File space not ready             */
#define DSM_RC_FS_IS_BAD           163 /* File space is bad                */
#define DSM_RC_FIO_ERROR           164 /* File input/output error          */
#define DSM_RC_WRITE_FAILURE       165 /* Error writing to file            */
#define DSM_RC_OVER_FILE_SIZE_LIMIT 166 /* File over system/user limit     */
#define DSM_RC_CANNOT_MAKE         167 /* Could not create file/directory,
                                          could be a bad name             */
#define DSM_RC_NO_PASS_FILE        168 /* password file needed and user is
                                          not root                        */
#define DSM_RC_VERFILE_OLD         169 /* password stored locally doesn't
                                          match the one at the host       */
#define DSM_RC_INPUT_ERROR         173 /* unable to read keyboard input    */
#define DSM_RC_REJECT_PLATFORM_MISMATCH 174 /* Platform name doesn't match
                                          up with what the server says
                                          is the platform for the client */
#define DSM_RC_TL_NOT_FILE_OWNER   175 /* User trying to backup a file is not
                                          the file's owner.               */
#define DSM_RC_COMPRESSED_DATA_CORRUPTED 176 /* Compressed data is corrupted*/
#define DSM_RC_UNMATCHED_QUOTE     177 /* missing starting or ending quote  */

#define DSM_RC_SIGNON_FAILOVER_MODE 178      /* Failed over to the replication server,
                                             running in failover mode */
#define DSM_RC_FAILOVER_MODE_FUNC_BLOCKED 179 /* function is blocked because
                                             session is in failover mode */

/*----------------------------------------------------------------------------*/
/* Return codes 180-199 are reserved for Policy Set handling                  */
/*----------------------------------------------------------------------------*/
#define DSM_RC_PS_MULTBCG          181 /* Multiple backup copy groups in 1 MC*/
#define DSM_RC_PS_MULTACG          182 /* Multiple arch.  copy groups in 1 MC*/
#define DSM_RC_PS_NODFLTMC         183 /* Default MC name not in policy set  */
#define DSM_RC_TL_NOBCG            184 /* Backup req, no backup copy group   */
#define DSM_RC_TL_EXCLUDED         185 /* Backup req, excl. by in/ex filter  */
#define DSM_RC_TL_NOACG            186 /* Archive req, no archive copy group */
#define DSM_RC_PS_INVALID_ARCHMC   187 /* Invalid MC name in archive override*/
#define DSM_RC_NO_PS_DATA          188 /* No policy set data on the server   */
#define DSM_RC_PS_INVALID_DIRMC    189 /* Invalid directory MC specified in
                                          the options file.               */
#define DSM_RC_PS_NO_CG_IN_DIR_MC  190 /* No backup copy group in directory MC.
                                          Must specify an MC using DirMC
                                          .                               */

#define DSM_RC_WIN32_UNSUPPORTED_FILE_TYPE 280 /* File is not of
                                             Win32 type FILE_TYPE_DISK  */

/*----------------------------------------------------------------------------*/
/* Return codes for the Trusted Communication Agent                           */
/*----------------------------------------------------------------------------*/
#define DSM_RC_TCA_NOT_ROOT        161 /* Access to TA is denied             */
#define DSM_RC_TCA_ATTACH_SHR_MEM_ERR  200 /* Error attaching shared memory  */
#define DSM_RC_TCA_SHR_MEM_BLOCK_ERR   200 /* Shared memory block error      */
#define DSM_RC_TCA_SHR_MEM_IN_USE      200 /* Shared memory block error      */
#define DSM_RC_TCA_SHARED_MEMORY_ERROR 200 /* Shared memory block error      */
#define DSM_RC_TCA_SEGMENT_MISMATCH    200 /* Shared memory block error      */
#define DSM_RC_TCA_FORK_FAILED     292 /* Error forking off TCA process      */
#define DSM_RC_TCA_DIED            294 /* TCA died unexpectedly              */
#define DSM_RC_TCA_INVALID_REQUEST 295 /* Invalid request sent to TCA        */
#define DSM_RC_TCA_SEMGET_ERROR    297 /* Error getting semaphores           */
#define DSM_RC_TCA_SEM_OP_ERROR    298 /* Error in semaphore set or wait     */
#define DSM_RC_TCA_NOT_ALLOWED     299 /* TCA not allowed (multi thread)     */

/*----------------------------------------------------------------------------*/
/* 400-430  for options                                                       */
/*----------------------------------------------------------------------------*/
#define DSM_RC_INVALID_OPT           400 /* invalid option                   */
#define DSM_RC_NO_HOST_ADDR          405 /* Not enuf info to connect server  */
#define DSM_RC_NO_OPT_FILE           406 /* No default user configuration file*/
#define DSM_RC_MACHINE_SAME          408 /* -MACHINENAME same as real name   */
#define DSM_RC_INVALID_SERVER        409 /* Invalid server name from client  */
#define DSM_RC_INVALID_KEYWORD       410 /* Invalid option keyword           */
#define DSM_RC_PATTERN_TOO_COMPLEX   411 /* Can't match Include/Exclude entry*/
#define DSM_RC_NO_CLOSING_BRACKET    412 /* Missing closing bracket inc/excl */
#define DSM_RC_OPT_CLIENT_NOT_ACCEPTING 417/* Client doesn't accept this option
                                          from the server                  */
#define DSM_RC_OPT_CLIENT_DOES_NOT_WANT 418/* Client doesn't want this value
                                          from the server                  */
#define DSM_RC_OPT_NO_INCLEXCL_FILE  419   /* inclexcl file not found        */
#define DSM_RC_OPT_OPEN_FAILURE      420   /* can't open file               */
```

```
#define DSM_RC_OPT_INV_NODENAME      421/* used for Windows if nodename=local
                                         machine when CLUSTERNODE=YES       */
#define DSM_RC_OPT_NODENAME_INVALID  423/* generic invalid nodename           */
#define DSM_RC_OPT_ERRORLOG_CONFLICT 424/* both logmax & retention specified  */
#define DSM_RC_OPT_SCHEDLOG_CONFLICT 425/* both logmax & retention specified  */
#define DSM_RC_CANNOT_OPEN_TRACEFILE 426/* cannot open trace file             */
#define DSM_RC_CANNOT_OPEN_LOGFILE   427/* cannot open error log file         */
#define DSM_RC_OPT_SESSINIT_LF_CONFLICT 428/* both sessioninit=server and
               enablelanfree=yes are specified*/
#define DSM_RC_OPT_OPTION_IGNORE     429/* option will be ignored            */
#define DSM_RC_OPT_DEDUP_CONFLICT    430/* cannot open error log file        */
#define DSM_RC_OPT_HSMLOG_CONFLICT 431/* both logmax & retention specified  */


/*----------------------------------------------------------------------------*/
/* 600 to 610 for volume label codes                                          */
/*----------------------------------------------------------------------------*/
#define DSM_RC_DUP_LABEL           600 /* duplicate volume label found       */
#define DSM_RC_NO_LABEL            601 /* drive has no label                 */

/*----------------------------------------------------------------------------*/
/* Return codes for message file processing                                   */
/*----------------------------------------------------------------------------*/
#define DSM_RC_NLS_CANT_OPEN_TXT   610 /* error trying to open msg txt file  */
#define DSM_RC_NLS_CANT_READ_HDR   611 /* error trying to read header        */
#define DSM_RC_NLS_INVALID_CNTL_REC 612 /* invalid control record            */
#define DSM_RC_NLS_INVALID_DATE_FMT 613 /* invalid default date format       */
#define DSM_RC_NLS_INVALID_TIME_FMT 614 /* invalid default time format       */
#define DSM_RC_NLS_INVALID_NUM_FMT 615 /* invalid default number format      */

/*----------------------------------------------------------------------------*/
/* Return codes 620-630 are reserved for log message return codes             */
/*----------------------------------------------------------------------------*/
#define DSM_RC_LOG_CANT_BE_OPENED  620 /* error trying to open error log     */
#define DSM_RC_LOG_ERROR_WRITING_TO_LOG 621 /* error occurred writing to
                                       log file                            */
#define DSM_RC_LOG_NOT_SPECIFIED   622 /* no error log file was specified    */


/*----------------------------------------------------------------------------*/
/* Return codes 900-999 TSM CLIENT ONLY                                       */
/*----------------------------------------------------------------------------*/
#define DSM_RC_NOT_ADSM_AUTHORIZED  927 /* Must be ADSM authorized to perform*/
                                    /* action : root user or pwd auth    */
#define DSM_RC_REJECT_USERID_UNKNOWN 940 /* userid unknown on server          */
#define DSM_RC_FILE_IS_SYMLINK      959 /* errorlog or trace is a symbolic
                                       link
                                                                        */

#define DSM_RC_DIRECT_STORAGE_AGENT_UNSUPPORTED 961 /* Direct connection to SA not supported */
#define DSM_RC_FS_NAMESPACE_DOWNLEVEL 963 /* Long namespace has been removed from
                                     from the Netware volume */
#define DSM_RC_CONTINUE_NEW_CONSUMER   972 /* Continue processing using a new consumer */
#define DSM_RC_CONTINUE_NEW_CONSUMER_NODEDUP   973 /* Continue processing using a new consumer no dedup*/
#define DSM_RC_CONTINUE_NEW_CONSUMER_NOCOMPRESS 976 /* Continue processing using a new consumer no compression */


#define DSM_RC_SERVER_SUPPORTS_FUNC    994 /* the server supports this function  */
#define DSM_RC_SERVER_AND_SA_SUPPORT_FUNC 995 /* Both server and SA support func */
#define DSM_RC_SERVER_DOWNLEVEL_FUNC   996 /* The server is downlevel for func   */
#define DSM_RC_STORAGEAGENT_DOWNLEVEL  997 /* the storage agent is downlevel      */
#define DSM_RC_SERVER_AND_SA_DOWNLEVEL 998 /* both server and SA downlevel       */


/* TCP/IP error codes */
#define DSM_RC_TCPIP_FAILURE       -50 /* TCP/IP communications failure      */
#define DSM_RC_CONN_TIMEDOUT       -51 /* TCP/IP connection attempt timedout */
#define DSM_RC_CONN_REFUSED        -52 /* TCP/IP connection refused by host  */
#define DSM_RC_BAD_HOST_NAME       -53 /* TCP/IP invalid host name specified */
#define DSM_RC_NETWORK_UNREACHABLE -54 /* TCP/IP host name unreachable       */
#define DSM_RC_WINSOCK_MISSING     -55 /* TCP/IP WINSOCK.DLL missing         */
#define DSM_RC_TCPIP_DLL_LOADFAILURE -56 /* Error from LoadLibrary           */
#define DSM_RC_TCPIP_LOADFAILURE   -57 /* Error from GetProcAddress   */
#define DSM_RC_TCPIP_USER_ABORT    -58 /* User aborted while in TCP/IP layer */

/*----------------------------------------------------------------------------*/
/* Return codes (-71)-(-90) are reserved for CommTSM error codes              */
/*----------------------------------------------------------------------------*/
#define DSM_RC_TSM_FAILURE         -71 /* TSM communications failure         */
```

```
#define DSM_RC_TSM_ABORT           -72 /* Session aborted abnormally         */

/*comm3270 error codes - no longer used*/
#define DSM_RC_COMM_TIMEOUT       2021   /* no longer used                    */
#define DSM_RC_EMULATOR_INACTIVE  2021   /* no longer used                    */
#define DSM_RC_BAD_HOST_ID        2021   /* no longer used                    */
#define DSM_RC_HOST_SESS_BUSY     2021   /* no longer used                    */
#define DSM_RC_3270_CONNECT_FAILURE 2021 /* no longer used                    */
#define DSM_RC_NO_ACS3ELKE_DLL    2021   /* no longer used                    */
#define DSM_RC_EMULATOR_ERROR     2021   /* no longer used                    */
#define DSM_RC_EMULATOR_BACKLEVEL 2021   /* no longer used                    */
#define DSM_RC_CKSUM_FAILURE      2021   /* no longer used                    */


/* The following Return codes are for EHLLAPI for Windows                    */
#define DSM_RC_3270COMMError_DLL        2021   /* no longer used             */
#define DSM_RC_3270COMMError_GetProc    2021   /* no longer used             */
#define DSM_RC_EHLLAPIError_DLL         2021   /* no longer used             */
#define DSM_RC_EHLLAPIError_GetProc     2021   /* no longer used             */
#define DSM_RC_EHLLAPIError_HostConnect 2021   /* no longer used             */
#define DSM_RC_EHLLAPIError_AllocBuff   2021   /* no longer used             */
#define DSM_RC_EHLLAPIError_SendKey     2021   /* no longer used             */
#define DSM_RC_EHLLAPIError_PacketChk   2021   /* no longer used             */
#define DSM_RC_EHLLAPIError_ChkSum      2021   /* no longer used             */
#define DSM_RC_EHLLAPIError_HostTimeOut 2021   /* no longer used             */
#define DSM_RC_EHLLAPIError_Send        2021   /* no longer used             */
#define DSM_RC_EHLLAPIError_Recv        2021   /* no longer used             */
#define DSM_RC_EHLLAPIError_General     2021   /* no longer used             */
#define DSM_RC_PC3270_MISSING_DLL       2021   /* no longer used             */
#define DSM_RC_3270COMM_MISSING_DLL     2021   /* no longer used             */


/* NETBIOS error codes */
#define DSM_RC_NETB_ERROR         -151 /* Could not add node to LAN          */
#define DSM_RC_NETB_NO_DLL        -152 /* The ACSNETB.DLL could not be loaded*/
#define DSM_RC_NETB_LAN_ERR       -155 /* LAN error detected                 */
#define DSM_RC_NETB_NAME_ERR      -158 /* Netbios error on Add Name          */
#define DSM_RC_NETB_TIMEOUT       -159 /* Netbios send timeout               */
#define DSM_RC_NETB_NOTINST       -160 /* Netbios not installed - DOS        */
#define DSM_RC_NETB_REBOOT        -161 /* Netbios config err - reboot DOS    */

/* Named Pipe error codes */
#define DSM_RC_NP_ERROR                       -190

/* CPIC error codes */
#define DSM_RC_CPIC_ALLOCATE_FAILURE          2021 /* no longer used      */
#define DSM_RC_CPIC_TYPE_MISMATCH             2021 /* no longer used      */
#define DSM_RC_CPIC_PIP_NOT_SPECIFY_ERR       2021 /* no longer used      */
#define DSM_RC_CPIC_SECURITY_NOT_VALID        2021 /* no longer used      */
#define DSM_RC_CPIC_SYNC_LVL_NO_SUPPORT       2021 /* no longer used      */
#define DSM_RC_CPIC_TPN_NOT_RECOGNIZED        2021 /* no longer used      */
#define DSM_RC_CPIC_TP_ERROR                  2021 /* no longer used      */
#define DSM_RC_CPIC_PARAMETER_ERROR           2021 /* no longer used      */
#define DSM_RC_CPIC_PROD_SPECIFIC_ERR         2021 /* no longer used      */
#define DSM_RC_CPIC_PROGRAM_ERROR             2021 /* no longer used      */
#define DSM_RC_CPIC_RESOURCE_ERROR            2021 /* no longer used      */
#define DSM_RC_CPIC_DEALLOCATE_ERROR          2021 /* no longer used      */
#define DSM_RC_CPIC_SVC_ERROR                 2021 /* no longer used      */
#define DSM_RC_CPIC_PROGRAM_STATE_CHECK       2021 /* no longer used      */
#define DSM_RC_CPIC_PROGRAM_PARAM_CHECK       2021 /* no longer used      */
#define DSM_RC_CPIC_UNSUCCESSFUL              2021 /* no longer used      */
#define DSM_RC_UNKNOWN_CPIC_PROBLEM           2021 /* no longer used      */
#define DSM_RC_CPIC_MISSING_LU                2021 /* no longer used      */
#define DSM_RC_CPIC_MISSING_TP                2021 /* no longer used      */
#define DSM_RC_CPIC_SNA6000_LOAD_FAIL         2021 /* no longer used      */
#define DSM_RC_CPIC_STARTUP_FAILURE           2021 /* no longer used      */

/*---------------------------------------------------------------------------*/
/* Return codes -300 to -307 are reserved for IPX/SPX communications         */
/*---------------------------------------------------------------------------*/
#define DSM_RC_TLI_ERROR                      2021 /* no longer used      */
#define DSM_RC_IPXSPX_FAILURE                 2021 /* no longer used      */
#define DSM_RC_TLI_DLL_MISSING                2021 /* no longer used      */
#define DSM_RC_DLL_LOADFAILURE                2021 /* no longer used      */
#define DSM_RC_DLL_FUNCTION_LOADFAILURE       2021 /* no longer used      */
#define DSM_RC_IPXCONN_REFUSED                2021 /* no longer used      */
#define DSM_RC_IPXCONN_TIMEDOUT               2021 /* no longer used      */
#define DSM_RC_IPXADDR_UNREACHABLE            2021 /* no longer used      */
#define DSM_RC_CPIC_MISSING_DLL               2021 /* no longer used      */
```

```
#define DSM_RC_CPIC_DLL_LOADFAILURE           2021 /* no longer used      */
#define DSM_RC_CPIC_FUNC_LOADFAILURE          2021 /* no longer used      */

/*=== Shared Memory Protocol error codes   ===*/
#define DSM_RC_SHM_TCPIP_FAILURE              -450
#define DSM_RC_SHM_FAILURE                    -451
#define DSM_RC_SHM_NOTAUTH                    -452

#define DSM_RC_NULL_OBJNAME        2000 /* Object name pointer is NULL      */
#define DSM_RC_NULL_DATABLKPTR     2001 /* dataBlkPtr is NULL               */
#define DSM_RC_NULL_MSG            2002 /* msg parm in dsmRCMsg is NULL     */

#define DSM_RC_NULL_OBJATTRPTR     2004 /* Object Attr Pointer is NULL      */

#define DSM_RC_NO_SESS_BLK         2006 /* no server session info           */
#define DSM_RC_NO_POLICY_BLK       2007 /* no policy hdr     info           */
#define DSM_RC_ZERO_BUFLEN         2008 /* bufferLen is zero for dataBlkPtr */
#define DSM_RC_NULL_BUFPTR         2009 /* bufferPtr is NULL for dataBlkPtr */

#define DSM_RC_INVALID_OBJTYPE     2010 /* invalid object type              */
#define DSM_RC_INVALID_VOTE        2011 /* invalid vote                     */
#define DSM_RC_INVALID_ACTION      2012 /* invalid action                   */
#define DSM_RC_INVALID_DS_HANDLE   2014 /* invalid ADSM handle              */
#define DSM_RC_INVALID_REPOS       2015 /* invalid value for repository     */
#define DSM_RC_INVALID_FSNAME      2016 /* fs should start with dir delim   */
#define DSM_RC_INVALID_OBJNAME     2017 /* invalid full path name           */
#define DSM_RC_INVALID_LLNAME      2018 /* ll should start with dir delim   */
#define DSM_RC_INVALID_OBJOWNER    2019 /* invalid object owner name        */
#define DSM_RC_INVALID_ACTYPE      2020 /* invalid action type              */
#define DSM_RC_INVALID_RETCODE     2021 /* dsmRC in dsmRCMsg is invalid     */
#define DSM_RC_INVALID_SENDTYPE    2022 /* invalid send type                */
#define DSM_RC_INVALID_PARAMETER   2023 /* invalid parameter                */
#define DSM_RC_INVALID_OBJSTATE    2024 /* active, inactive, or any match?  */
#define DSM_RC_INVALID_MCNAME      2025 /* Mgmt class name not found        */
#define DSM_RC_INVALID_DRIVE_CHAR  2026 /* Drive letter is not alphabet     */
#define DSM_RC_NULL_FSNAME         2027 /* Filespace name is NULL           */
#define DSM_RC_INVALID_HLNAME      2028 /* hl should start with dir delim   */

#define DSM_RC_NUMOBJ_EXCEED       2029 /* BeginGetData num objs exceeded   */

#define DSM_RC_NEWPW_REQD          2030 /* new password is required         */
#define DSM_RC_OLDPW_REQD          2031 /* old password is required         */
#define DSM_RC_NO_OWNER_REQD       2032 /* owner not allowed. Allow default */
#define DSM_RC_NO_NODE_REQD        2033 /* node not allowed w/ pw=generate  */
#define DSM_RC_KEY_MISSING         2034 /* key file can't be found          */
#define DSM_RC_KEY_BAD             2035 /* content of key file is bad       */

#define DSM_RC_BAD_CALL_SEQUENCE   2041 /* Sequence of DSM calls not allowed*/
#define DSM_RC_INVALID_TSMBUFFER   2042 /* invalid value for tsmbuffhandle or dataPtr */
#define DSM_RC_TOO_MANY_BYTES      2043 /* too many bytes copied to buffer  */
#define DSM_RC_MUST_RELEASE_BUFFER 2044 /* cant exit app needs to release buffers */
#define DSM_RC_BUFF_ARRAY_ERROR    2045 /* internal buff array error        */
#define DSM_RC_INVALID_DATABLK     2046 /* using tsmbuff datablk should be null */
#define DSM_RC_ENCR_NOT_ALLOWED    2047 /* when using tsmbuffers encription not allowed */
#define DSM_RC_OBJ_COMPRESSED      2048 /* Can't restore using tsmBuff on compressed object */
#define DSM_RC_OBJ_ENCRYPTED       2049 /* Cant restore using tsmbuff an encr obj */
#define DSM_RC_WILDCHAR_NOTALLOWED 2050 /* Wild card not allowed for hl,ll  */
#define DSM_RC_POR_NOT_ALLOWED     2051 /* Can't use partial object restore with tsmBuffers */
#define DSM_RC_NO_ENCRYPTION_KEY   2052 /* Encryption key not found*/
#define DSM_RC_ENCR_CONFLICT       2053 /* mutually exclusive options */

#define DSM_RC_FSNAME_NOTFOUND     2060 /* Filespace name not found         */
#define DSM_RC_FS_NOT_REGISTERED   2061 /* Filespace name not registered    */
#define DSM_RC_FS_ALREADY_REGED    2062 /* Filespace already registered     */
#define DSM_RC_OBJID_NOTFOUND      2063 /* No object id to restore          */
#define DSM_RC_WRONG_VERSION       2064 /* Wrong level of code              */
#define DSM_RC_WRONG_VERSION_PARM  2065 /* Wrong level of parameter struct  */

#define DSM_RC_NEEDTO_ENDTXN       2070 /* Need to call dsmEndTxn           */

#define DSM_RC_OBJ_EXCLUDED        2080 /* Object is excluded by MC         */
#define DSM_RC_OBJ_NOBCG           2081 /* Object has no backup copy group  */
#define DSM_RC_OBJ_NOACG           2082 /* Object has no archive copy group */

#define DSM_RC_APISYSTEM_ERROR     2090 /* API internal error               */

#define DSM_RC_DESC_TOOLONG             2100 /* description is too long          */
#define DSM_RC_OBJINFO_TOOLONG          2101 /* object attr objinfo too long     */
#define DSM_RC_HL_TOOLONG               2102 /* High level qualifier is too long */
```

```
#define DSM_RC_PASSWD_TOOLONG           2103 /* password is too long            */
#define DSM_RC_FILESPACE_TOOLONG        2104 /* filespace name is too long      */
#define DSM_RC_LL_TOOLONG               2105 /* Low level qualifier is too long  */
#define DSM_RC_FSINFO_TOOLONG           2106 /* filespace length is too big     */
#define DSM_RC_SENDDATA_WITH_ZERO_SIZE  2107 /* send data w/ zero est           */

/*=== new return codes for dsmaccess ===*/
#define DSM_RC_INVALID_ACCESS_TYPE 2110 /* invalid access type             */
#define DSM_RC_QUERY_COMM_FAILURE  2111 /* communication error during query */
#define DSM_RC_NO_FILES_BACKUP     2112 /* No backed up files for this fs   */
#define DSM_RC_NO_FILES_ARCHIVE    2113 /* No archived files for this fs    */
#define DSM_RC_INVALID_SETACCESS   2114 /* invalid set access format        */

/*=== new return codes for dsmaccess ===*/
#define DSM_RC_STRING_TOO_LONG     2120 /* String parameter too long        */

#define DSM_RC_MORE_DATA           2200 /* There are more data to restore   */

#define DSM_RC_BUFF_TOO_SMALL      2210 /* DataBlk buffer too small for qry */

#define DSM_RC_NO_API_CONFIGFILE   2228 /*specified API confg file not found*/
#define DSM_RC_NO_INCLEXCL_FILE    2229 /* specified inclexcl file not found*/
#define DSM_RC_NO_SYS_OR_INCLEXCL  2230 /* either dsm.sys or inclexcl file
                                           specified in dsm.sys not found   */
#define DSM_RC_REJECT_NO_POR_SUPPORT 2231 /* server doesn't have POR support*/

#define DSM_RC_NEED_ROOT           2300 /* API caller must be root          */
#define DSM_RC_NEEDTO_CALL_BINDMC  2301 /* dsmBindMC must be called first   */
#define DSM_RC_CHECK_REASON_CODE   2302 /* check reason code from dsmEndTxn */
#define DSM_RC_NEEDTO_ENDTXN_DEDUP_SIZE_EXCEEDED 2303 /* max dedup bytes exceeded */

/*=== return codes 2400 - 2410 used by lic file see agentrc.h ===*/

/*=== return codes 2410 - 2430 used by Oracle agent see agentrc.h ===*/

#define DSM_RC_ENC_WRONG_KEY       4580 /* the key provided is incorrect    */
#define DSM_RC_ENC_NOT_AUTHORIZED  4582 /* user is not allowed to decrypt   */
#define DSM_RC_ENC_TYPE_UNKNOWN    4584 /* encryption type unknown          */

/*============================================================================
   Return codes (4600)-(4624) are reserved for clustering
============================================================================*/
#define DSM_RC_CLUSTER_INFO_LIBRARY_NOT_LOADED        4600
#define DSM_RC_CLUSTER_LIBRARY_INVALID                4601
#define DSM_RC_CLUSTER_LIBRARY_NOT_LOADED             4602
#define DSM_RC_CLUSTER_NOT_MEMBER_OF_CLUSTER          4603
#define DSM_RC_CLUSTER_NOT_ENABLED                    4604
#define DSM_RC_CLUSTER_NOT_SUPPORTED                  4605
#define DSM_RC_CLUSTER_UNKNOWN_ERROR                  4606


/*============================================================================
   Return codes (5701)-(5749) are reserved for proxy
============================================================================*/
#define DSM_RC_PROXY_REJECT_NO_RESOURCES              5702
#define DSM_RC_PROXY_REJECT_DUPLICATE_ID              5705
#define DSM_RC_PROXY_REJECT_ID_IN_USE                 5710
#define DSM_RC_PROXY_REJECT_INTERNAL_ERROR            5717
#define DSM_RC_PROXY_REJECT_NOT_AUTHORIZED            5722
#define DSM_RC_PROXY_INVALID_FROMNODE                 5746
#define DSM_RC_PROXY_INVALID_SERVERFREE               5747
#define DSM_RC_PROXY_INVALID_CLUSTER                  5748
#define DSM_RC_PROXY_INVALID_FUNCTION                 5749

/*============================================================================
   Return codes 5801 - 5849 are reserved for cryptography/security
============================================================================*/

#define DSM_RC_CRYPTO_ICC_ERROR                       5801
#define DSM_RC_CRYPTO_ICC_CANNOT_LOAD                 5802
#define DSM_RC_SSL_NOT_SUPPORTED                      5803
#define DSM_RC_SSL_INIT_FAILED                        5804
#define DSM_RC_SSL_KEYFILE_OPEN_FAILED                5805
#define DSM_RC_SSL_KEYFILE_BAD_PASSWORD               5806
#define DSM_RC_SSL_BAD_CERTIFICATE                    5807

/*============================================================================
   Return codes 6300 - 6399 are reserved for client-side deduplication
============================================================================*/
```

```
#define DSM_RC_DIGEST_VALIDATION_ERROR              6300  /* End-to-end digest validation err  */
#define DSM_RC_DATA_FINGERPRINT_ERROR               6301 /* Failure in Rabin fingeprinting    */
#define DSM_RC_DATA_DEDUP_ERROR                     6302 /* Error converting data into chunks */


#endif /* _H_DSMRC */
```

Référence associée:

Codes retour d'API

Annexe B. Fichiers source des définitions de type d'API

Cette annexe contient les définitions de structure, les définitions de type et les constantes pour l'API. Les premiers fichiers d'en-tête, `dsmapitd.h` et `tsmapitd.h`, illustrent les définitions qui sont communes à tous les systèmes d'exploitation.

Le deuxième fichier d'en-tête, `dsmapips.h`, fournit un exemple des définitions qui sont spécifiques à un système d'exploitation particulier ; dans cet exemple, la plateforme Windows.

Le troisième fichier d'en-tête, `release.h`, inclut les informations de version et d'édition.

Les informations fournies ici contiennent une copie ponctuelle des fichiers distribués avec l'API. Affichez les fichiers dans l'API Distribution Package pour la version la plus récente.

```
/*************************************************************************
* Tivoli Storage Manager                                *
* API Client Component                                  *
*                                                                       *
* (C) Copyright IBM Corporation 1993,2010                               *
*************************************************************************/

/*************************************************************************
* Header File Name: dsmapitd.h
*
* Environment:     ***********************************************
*                  ** This is a platform-independent source file **
*
*                  ***********************************************
*
* Design Notes:    This file contains basic data types and constants
*                  includable by all client source files. The constants
*                  within this file should be set properly for the
*                  particular machine and operating system on which the
*                  client software is to be run.
*
*                  Platform specific definitions are included in dsmapips.h
*
* Descriptive-name: Definitions for Tivoli Storage manager API constants
*-------------------------------------------------------------------------*/

#ifndef _H_DSMAPITD
#define _H_DSMAPITD


#include "dsmapips.h"      /* Platform specific definitions*/
#include "release.h"

/*=== set the structure alignment to pack the structures ===*/
#if (_OPSYS_TYPE == DS_WINNT) && !defined(_WIN64)
#pragma pack(1)
#endif

#ifdef _MAC
/*=========================================================================
  choices are:
http://developer.apple.com/documentation/DeveloperTools/Conceptual/PowerPCRuntime/Data/chapter_2_section_3.html

#pragma option align=<mode>
where <mode> is power, mac68k, natural, or packed.
=========================================================================*/
```

```
#pragma options align = packed
#endif

typedef char osChar_t;

/*<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>*/
/*                      D E F I N E S                                   */
/*<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>*/
/*-------------------------------------------------------------------------+
|  API Version, Release, and Level to use in dsmApiVersion on dsmInit()    |
+-------------------------------------------------------------------------*/
#define DSM_API_VERSION     COMMON_VERSION
#define DSM_API_RELEASE     COMMON_RELEASE
#define DSM_API_LEVEL       COMMON_LEVEL
#define DSM_API_SUBLEVEL    COMMON_SUBLEVEL

/*-------------------------------------------------------------------------+
| Maximum field lengths                                                    |
+-------------------------------------------------------------------------*/
#define DSM_MAX_CG_DEST_LENGTH        30        /* copy group destination */
#define DSM_MAX_CG_NAME_LENGTH        30        /* copy group name        */
#define DSM_MAX_DESCR_LENGTH          255       /* archive description    */
#define DSM_MAX_DOMAIN_LENGTH         30        /* policy domain name     */
#define DSM_MAX_FSINFO_LENGTH         500       /* filespace info         */
#define DSM_MAX_USER_FSINFO_LENGTH    480       /* max user filespace info*/
#define DSM_MAX_FSNAME_LENGTH         1024      /* filespace name         */
#define DSM_MAX_FSTYPE_LENGTH         32        /* filespace type         */
#define DSM_MAX_HL_LENGTH             1024      /* object high level name */
#define DSM_MAX_ID_LENGTH             64        /* session node name      */
#define DSM_MAX_LL_LENGTH             256       /* object low level name  */
#define DSM_MAX_MC_NAME_LENGTH        30        /* management class name  */
#define DSM_MAX_OBJINFO_LENGTH        255       /* object info            */
#define DSM_MAX_EXT_OBJINFO_LENGTH    1500      /* Extended object info   */
#define DSM_MAX_OWNER_LENGTH          64        /* object owner name      */
#define DSM_MAX_PLATFORM_LENGTH       16        /* application type       */
#define DSM_MAX_PS_NAME_LENGTH        30        /* policy set name        */
#define DSM_MAX_SERVERTYPE_LENGTH     32        /* server platform type   */
#define DSM_MAX_VERIFIER_LENGTH       64        /* password               */
#define DSM_PATH_MAX                  1024      /* API config file path   */
#define DSM_NAME_MAX                  255       /* API config file name   */
#define DSM_MAX_NODE_LENGTH           64        /* node/machine name      */
#define DSM_MAX_RC_MSG_LENGTH         1024      /* msg parm for dsmRCMsg  */
#define DSM_MAX_SERVER_ADDRESS        1024     /* server address */

#define DSM_MAX_MC_DESCR_LENGTH       DSM_MAX_DESCR_LENGTH /* mgmt class  */
#define DSM_MAX_SERVERNAME_LENGTH     DSM_MAX_ID_LENGTH /* server name    */
#define DSM_MAX_GET_OBJ               4080     /* max objs on BeginGetData */
#define DSM_MAX_PARTIAL_GET_OBJ       1300 /* max partial objs on BeginGetData */
#define DSM_MAX_COMPRESSTYPE_LENGTH   32       /* max compression algorithm name */

/*-------------------------------------------------------------------------+
| Minimum field lengths                                                    |
+-------------------------------------------------------------------------*/
#define DSM_MIN_COMPRESS_SIZE  2048 /* minimum number of bytes an object  */
                                    /* needs before compression is allowed*/

/*-------------------------------------------------------------------------+
|  Values for mtFlag in dsmSetup call                                      |
+-------------------------------------------------------------------------*/
#define DSM_MULTITHREAD      bTrue
#define DSM_SINGLETHREAD     bFalse

/*-------------------------------------------------------------------------+
|  Values for object type in dsmObjName structure                          |
|  Note: These values must be kept in sync with dsmcomm.h                  |
+-------------------------------------------------------------------------*/
#define DSM_OBJ_FILE                  0x01 /*object has attrib info & data*/
#define DSM_OBJ_DIRECTORY             0x02 /*obj has only attribute info  */
#define DSM_OBJ_RESERVED1             0x04 /* for future use              */
#define DSM_OBJ_RESERVED2             0x05 /* for future use              */
```

```
#define DSM_OBJ_RESERVED3            0x06 /* for future use             */
#define DSM_OBJ_WILDCARD             0xFE /* Any object type            */
#define DSM_OBJ_ANY_TYPE             0xFF /* for future use             */

/*-------------------------------------------------------------------------+
| Type definition for compressedState in QryResp                           |
+-------------------------------------------------------------------------*/
#define DSM_OBJ_COMPRESSED_UNKNOWN    0
#define DSM_OBJ_COMPRESSED_YES        1
#define DSM_OBJ_COMPRESSED_NO         2

/*-------------------------------------------------------------------------+
| Definitions for "group type" field in tsmGrouphandlerIn_t                |
+-------------------------------------------------------------------------*/

#define  DSM_GROUPTYPE_NONE         0x00   /* Not a group member           */
#define  DSM_GROUPTYPE_RESERVED1    0x01   /* for future use               */
#define  DSM_GROUPTYPE_PEER         0x02   /* Peer group                   */
#define  DSM_GROUPTYPE_RESERVED2    0x03   /* for future use               */

/*-------------------------------------------------------------------------+
| Definitions for "member type" field in tsmGrouphandlerIn_t               |
+-------------------------------------------------------------------------*/

#define  DSM_MEMBERTYPE_LEADER      0x01   /* group leader  */
#define  DSM_MEMBERTYPE_MEMBER      0x02   /* group member  */

/*-------------------------------------------------------------------------+
| Definitions for "operation type" field in tsmGrouphandlerIn_t            |
+-------------------------------------------------------------------------*/
#define DSM_GROUP_ACTION_BEGIN         0x01
#define DSM_GROUP_ACTION_OPEN          0x02 /* create new group             */
#define DSM_GROUP_ACTION_CLOSE         0x03 /* commit and save an open group */
#define DSM_GROUP_ACTION_ADD           0x04 /* Append to a group */
#define DSM_GROUP_ACTION_ASSIGNTO      0x05 /* Assign to a another group */
#define DSM_GROUP_ACTION_REMOVE        0x06 /* remove a member from a group */

/*-------------------------------------------------------------------------+
| Values for copySer in DetailCG structures for Query Mgmt Class response  |
+-------------------------------------------------------------------------*/
#define Copy_Serial_Static          1   /*Copy Serialization Static       */
#define Copy_Serial_Shared_Static   2   /*Copy Serialization Shared Static*/
#define Copy_Serial_Shared_Dynamic  3   /*Copy Serialization Shared Dynamic*/
#define Copy_Serial_Dynamic         4   /*Copy Serialization Dynamic      */

/*-------------------------------------------------------------------------+
| Values for copyMode in DetailCG structures for Query Mgmt Class response |
+-------------------------------------------------------------------------*/
#define Copy_Mode_Modified          1   /*Copy Mode Modified              */
#define Copy_Mode_Absolute          2   /*Copy Mode Absolute              */

/*-------------------------------------------------------------------------+
|  Values for objState in qryBackupData structure                          |
+-------------------------------------------------------------------------*/
#define DSM_ACTIVE            0x01      /* query only active objects      */
#define DSM_INACTIVE          0x02      /* query only inactive objects    */
#define DSM_ANY_MATCH         0xFF      /* query all backup objects       */

/*-------------------------------------------------------------------------+
| Boundary values for dsmDate.year field in qryArchiveData structure       |
+-------------------------------------------------------------------------*/
#define DATE_MINUS_INFINITE         0x0000      /* lowest boundary        */
#define DATE_PLUS_INFINITE          0xFFFF      /* highest upper boundary */

/*-------------------------------------------------------------------------+
| Bits masks for update action parameter on dsmUpdateFS()                  |
+-------------------------------------------------------------------------*/
#define DSM_FSUPD_FSTYPE              ((unsigned) 0x00000002)
#define DSM_FSUPD_FSINFO              ((unsigned) 0x00000004)
#define DSM_FSUPD_BACKSTARTDATE       ((unsigned) 0x00000008)
```

```
#define DSM_FSUPD_BACKCOMPLETEDATE    ((unsigned) 0x00000010)
#define DSM_FSUPD_OCCUPANCY           ((unsigned) 0x00000020)
#define DSM_FSUPD_CAPACITY            ((unsigned) 0x00000040)
#define DSM_FSUPD_RESERVED1           ((unsigned) 0x00000100)

/*-------------------------------------------------------------------------+
| Bits mask  for backup update action parameter on dsmUpdateObj()          |
+-------------------------------------------------------------------------*/
#define DSM_BACKUPD_OWNER              ((unsigned) 0x00000001)
#define DSM_BACKUPD_OBJINFO            ((unsigned) 0x00000002)
#define DSM_BACKUPD_MC                 ((unsigned) 0x00000004)

#define DSM_ARCHUPD_OWNER              ((unsigned) 0x00000001)
#define DSM_ARCHUPD_OBJINFO            ((unsigned) 0x00000002)
#define DSM_ARCHUPD_DESCR              ((unsigned) 0x00000004)

/*-------------------------------------------------------------------------+
|  Values for repository parameter on dsmDeleteFS()                        |
+-------------------------------------------------------------------------*/
#define DSM_ARCHIVE_REP       0x0A      /* archive repository             */
#define DSM_BACKUP_REP        0x0B      /* backup repository              */
#define DSM_REPOS_ALL         0x01      /* all respository types          */

/*-------------------------------------------------------------------------+
|  Values for vote parameter on dsmEndTxn()                                |
+-------------------------------------------------------------------------*/
#define DSM_VOTE_COMMIT  1             /* commit current transaction      */
#define DSM_VOTE_ABORT   2             /* roll back current transaction   */

/*-------------------------------------------------------------------------+
|  Values for various flags returned in ApiSessInfo structure.             |
+-------------------------------------------------------------------------*/
/* Client compression field codes */
#define COMPRESS_YES    1    /* client must compress data             */
#define COMPRESS_NO     2    /* client must NOT compress data         */
#define COMPRESS_CD     3    /* client determined                     */

/* Archive delete permission codes. */
#define ARCHDEL_YES     1    /* archive delete allowed                */
#define ARCHDEL_NO      2    /* archive delete NOT allowed            */

/* Backup delete permission codes. */
#define BACKDEL_YES     1    /* backup delete allowed                 */
#define BACKDEL_NO      2    /* backup delete NOT allowed             */


/*-------------------------------------------------------------------------+
  Values for various flags returned in optStruct structure.              |
--------------------------------------------------------------------------*/
#define DSM_PASSWD_GENERATE  1
#define DSM_PASSWD_PROMPT    0

#define DSM_COMM_TCP        1    /* tcpip         */
#define DSM_COMM_NAMEDPIPE  2    /* Named pipes   */
#define DSM_COMM_SHM        3    /* Shared Memory */

/* obsolete commmethods */
#define DSM_COMM_PVM_IUCV   12
#define DSM_COMM_3270       12
#define DSM_COMM_IUCV       12
#define DSM_COMM_PWSCS      12
#define DSM_COMM_SNA_LU6_2  12
#define DSM_COMM_IPXSPX     12    /* For IPX/SPX support */
#define DSM_COMM_NETBIOS    12    /* NETBIOS */
#define DSM_COMM_400COMM    12
#define DSM_COMM_CLIO      12    /* CLIO/S */
/*-------------------------------------------------------------------------+
|  Values for userNameAuthorities in dsmInitEx for future use              |
+-------------------------------------------------------------------------*/
#define DSM_USERAUTH_NONE     ((dsInt16_t)0x0000)
```

```
#define DSM_USERAUTH_ACCESS   ((dsInt16_t)0x0001)
#define DSM_USERAUTH_OWNER    ((dsInt16_t)0x0002)
#define DSM_USERAUTH_POLICY   ((dsInt16_t)0x0004)
#define DSM_USERAUTH_SYSTEM   ((dsInt16_t)0x0008)

/*-------------------------------------------------------------------------+
|  Values for encryptionType on dsmEndSendObjEx, queryResp                 |
+-------------------------------------------------------------------------*/
#define DSM_ENCRYPT_NO              ((dsUint8_t)0x00)
#define DSM_ENCRYPT_USER            ((dsUint8_t)0x01)
#define DSM_ENCRYPT_CLIENTENCRKEY   ((dsUint8_t)0x02)
#define DSM_ENCRYPT_DES_56BIT       ((dsUint8_t)0x04)
#define DSM_ENCRYPT_AES_128BIT      ((dsUint8_t)0x08)
#define DSM_ENCRYPT_AES_256BIT      ((dsUint8_t)0x10)

/*-------------------------------------------------------------------------+
| Definitions for mediaClass field.                                        |
+-------------------------------------------------------------------------*/
/*
 *  The following constants define a hierarchy of media access classes.
 *  Lower numbers indicate media which can supply faster access to data.
 */

/* Fixed: represents the class of on-line, fixed media (such as
          hard disks). */
#define  MEDIA_FIXED          0x10

/* Library: represents the class of mountable media accessible
            through a mechanical mounting device. */
#define  MEDIA_LIBRARY        0x20

/* future use */
#define  MEDIA_NETWORK        0x30

/* future use */
#define  MEDIA_SHELF          0x40

/* future use */
#define  MEDIA_OFFSITE        0x50

/* future use */
#define  MEDIA_UNAVAILABLE    0xF0


/*-------------------------------------------------------------------------+
| Type definition for partial object data for dsmBeginGetData()            |
+-------------------------------------------------------------------------*/
typedef struct
{
    dsUint16_t     stVersion;           /* Structure version               */
    dsStruct64_t   partialObjOffset;   /* offset into object to begin reading   */
    dsStruct64_t   partialObjLength;   /* amount of object to read              */
} PartialObjData ;              /* partial object data                  */

#define PartialObjDataVersion 1  /*                                      */

/*-------------------------------------------------------------------------+
|  Type definition for date structure                                      |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t    year;                 /* année, entier 16 bits (ex. 1990) */
   dsUint8_t     month;                /* month, 8-bit integer (1 - 12)     */
   dsUint8_t     day;                  /* day. 8-bit integer (1 - 31)       */
   dsUint8_t     hour;                 /* hour, 8-bit integer (0 - 23)      */
   dsUint8_t     minute;               /* minute, 8-bit integer (0 - 59)    */
   dsUint8_t     second;               /* second, b-bit integer (0 - 59)    */
}dsmDate ;

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

```
|  Type definition for Object ID on dsmGetObj() and in dsmGetList structure|
+-------------------------------------------------------------------------*/
typedef dsStruct64_t  ObjID ;

/*-------------------------------------------------------------------------+
| Type definition for dsmQueryBuff on dsmBeginQuery()                      |
+-------------------------------------------------------------------------*/
typedef void dsmQueryBuff ;

/*-------------------------------------------------------------------------+
| Type definition for dsmGetType parameter on dsmBeginGetData()            |
+-------------------------------------------------------------------------*/
typedef enum
{
        gtBackup = 0x00,                    /* Backup processing type    */
        gtArchive                           /* Archive processing type   */
} dsmGetType ;

/*-------------------------------------------------------------------------+
|  Type definition for dsmQueryType parameter on dsmBeginQuery()           |
+-------------------------------------------------------------------------*/
typedef enum
{
   qtArchive = 0x00,                /* Archive query type                */
   qtBackup,                        /* Backup query type                 */
   qtBackupActive,                  /* Fast query for active backup files */
   qtFilespace,                     /* Filespace query type              */
   qtMC,                            /* Mgmt. class query type            */
   qtReserved1,                     /* future use                        */
   qtReserved2,                     /* future use                        */
   qtReserved3,                     /* future use                        */
   qtReserved4,                     /* future use                        */
   qtBackupGroups,                  /* group leaders in a specific fs    */
   qtOpenGroups,                    /* Open groups in a specific fs      */
   qtReserved5,                     /* future use                        */
   qtProxyNodeAuth,                 /* nodes that his node can proxy to  */
   qtProxyNodePeer,                 /* Peer nodes with the same target   */
   qtReserved6,                     /* future use                        */
   qtReserved7,                     /* future use                        */
   qtReserved8                      /* future use                        */
}dsmQueryType ;

/*-------------------------------------------------------------------------+
|  Type definition sendType parameter on dsmBindMC() and dsmSendObj()      |
+-------------------------------------------------------------------------*/
typedef enum
{
   stBackup = 0x00,                        /* Backup processing type    */
   stArchive,                              /* Archive processing type   */
   stBackupMountWait,           /* Backup processing with mountwait on   */
   stArchiveMountWait           /* Archive processing with mountwait on  */
}dsmSendType ;

/*-------------------------------------------------------------------------+
|  Type definition for delType parameter on dsmDeleteObj()                 |
+-------------------------------------------------------------------------*/
typedef enum
{
   dtArchive = 0x00,                             /* Archive delete type */
   dtBackup,                          /* Backup delete (deactivate) type */
   dtBackupID                        /* Backup delete (remove)     type */
}dsmDelType ;

/*-------------------------------------------------------------------------+
|  Type definition sendType parameter on dsmSetAccess()                    |
+-------------------------------------------------------------------------*/
typedef enum
{
   atBackup = 0x00,                        /* Backup processing type    */
   atArchive                              /* Archive processing type   */
```

```
}dsmAccessType;

/*-------------------------------------------------------------------------+
|  Type definition for API Version on dsmInit() and dsmQueryApiVersion()   |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t version;       /* API version                    */
   dsUint16_t release;       /* API release                    */
   dsUint16_t level;         /* API level                      */
}dsmApiVersion;

/*-------------------------------------------------------------------------+
|  Type definition for API Version on dsmInit() and dsmQueryApiVersion()   |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t stVersion;     /* Structure version              */
   dsUint16_t version;       /* API version                    */
   dsUint16_t release;       /* API release                    */
   dsUint16_t level;         /* API level                      */
   dsUint16_t subLevel;      /* API sub level                  */
   dsmBool_t  unicode;       /* API unicode?                   */
}dsmApiVersionEx;


#define apiVersionExVer       2

/*-------------------------------------------------------------------------+
|  Type definition for Application Version on dsmInit()                    |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t     stVersion;           /* Structure version               */
   dsUint16_t   applicationVersion;   /* application version number   */
   dsUint16_t   applicationRelease;   /* application release number   */
   dsUint16_t   applicationLevel;     /* application level number     */
   dsUint16_t   applicationSubLevel;  /* application sub level number */
} dsmAppVersion;

#define appVersionVer      1

/*-------------------------------------------------------------------------+
|  Type definition for object name used on BindMC, Send, Delete, Query     |
+-------------------------------------------------------------------------*/
typedef struct S_dsmObjName
{
   char       fs[DSM_MAX_FSNAME_LENGTH + 1] ;              /* Filespace name */
   char       hl[DSM_MAX_HL_LENGTH + 1] ;                 /* High level name */
   char       ll[DSM_MAX_LL_LENGTH + 1] ;                  /* Low level name */
   dsUint8_t  objType;          /* for object type values, see defines above */
}dsmObjName;


/*-------------------------------------------------------------------------+
|  Type definition for Backup delete info on dsmDeleteObj()                |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t        stVersion ;                   /* structure version      */
   dsmObjName       *objNameP ;                   /* object name            */
   dsUint32_t        copyGroup ;                  /* copy group             */
}delBack ;

#define  delBackVersion    1

/*-------------------------------------------------------------------------+
|  Type definition for Archive delete info on dsmDeleteObj()               |
+-------------------------------------------------------------------------*/
typedef struct
```

```
{
   dsUint16_t        stVersion ;                    /* structure version      */
   dsStruct64_t       objId ;                        /* object ID             */
}delArch ;

#define  delArchVersion  1

/*--------------------------------------------------------------------------+
|  Type definition for Backup ID delete info on dsmDeleteObj()              |
+--------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t        stVersion ;                    /* structure version      */
   dsStruct64_t       objId ;                        /* object ID             */
}delBackID;

#define  delBackIDVersion  1

/*--------------------------------------------------------------------------+
|  Type definition for delete info on dsmDeleteObj()                        |
+--------------------------------------------------------------------------*/
typedef union
{
   delBack   backInfo ;
   delArch   archInfo ;
   delBackID backIDInfo ;
}dsmDelInfo ;

/*--------------------------------------------------------------------------+
|  Type definition for Object Attribute parameter on dsmSendObj()           |
+--------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t   stVersion;               /* Structure version                */
   char         owner[DSM_MAX_OWNER_LENGTH + 1];      /* object owner */
   dsStruct64_t sizeEstimate;            /* Size estimate in bytes of the object */
   dsmBool_t    objCompressed;                 /* Is object already compressed? */
   dsUint16_t   objInfoLength;           /* length of object-dependent info   */
   char         *objInfo;      /* object-dependent info */
   char         *mcNameP;               /* mgmnt class name for override  */
   dsmBool_t    disableDeduplication;         /* force no dedup for this object    */
   dsmBool_t    useExtObjInfo;                /* use ext obj info up to 1536 */
}ObjAttr;

#define ObjAttrVersion  4



/*--------------------------------------------------------------------------+
| Type definition for mcBindKey returned on dsmBindMC()                     |
+--------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t         stVersion;              /* structure version             */
   char        mcName[DSM_MAX_MC_NAME_LENGTH + 1];
                                      /* Name of mc bound to object. */
   dsmBool_t   backup_cg_exists;                        /* True/false */
   dsmBool_t   archive_cg_exists;                       /* True/false */
   char        backup_copy_dest[DSM_MAX_CG_DEST_LENGTH + 1];
                                           /* Backup copy dest. name */
   char        archive_copy_dest[DSM_MAX_CG_DEST_LENGTH + 1];
                                           /* Arch copy dest.name */
}mcBindKey;

#define mcBindKeyVersion  1


/*--------------------------------------------------------------------------+
| Type definition for object list on dsmBeginGetData()                      |
+--------------------------------------------------------------------------*/
```

```
typedef struct
{
   dsUint16_t        stVersion ;                   /* structure version      */
   dsUint32_t       numObjId ;          /* number of object IDs in the list */
   ObjID           *objId ;             /* list of object IDs to restore*/
   PartialObjData  *partialObjData;     /*list of partial obj data info */
}dsmGetList ;

#define dsmGetListVersion     2  /* default if not using Partial Obj data */
#define dsmGetListPORVersion  3  /* version if using Partial Obj data     */

/*-------------------------------------------------------------------------+
|  Type definition for DataBlk used to Get or Send data                    |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t        stVersion ;                   /* structure version      */
   dsUint32_t bufferLen;                /* Length of buffer passed below */
   dsUint32_t numBytes;                 /* Actual number of bytes read from */
                                        /* or written to the buffer */
   char       *bufferPtr;               /* Data buffer */
   dsUint32_t numBytesCompressed;       /* on send actual bytes compressed */
   dsUint16_t reserved;                 /* for future use                  */
}DataBlk;

#define DataBlkVersion  3


/*-------------------------------------------------------------------------+
|  Type definition for Mgmt Class queryBuffer on dsmBeginQuery()           |
+-------------------------------------------------------------------------*/
typedef struct S_qryMCData
{
   dsUint16_t         stVersion;              /* structure version              */
   char         *mcName;                      /* Mgmt class name */
                     /* single name to get one or empty string to get all*/
   dsmBool_t    mcDetail;                           /* Want details or not? */
}qryMCData;

#define qryMCDataVersion  1

/*=== values for RETINIT ===*/
#define ARCH_RETINIT_CREATE 0
#define ARCH_RETINIT_EVENT  1

/*-------------------------------------------------------------------------+
|  Type definition for Archive Copy Group details on Query MC response     |
+-------------------------------------------------------------------------*/
typedef struct S_archDetailCG
{
   char         cgName[DSM_MAX_CG_NAME_LENGTH + 1];       /* Copy group name */
   dsUint16_t   frequency;                     /* Copy (archive) frequency */
   dsUint16_t   retainVers;                              /* Retain version */
   dsUint8_t    copySer;     /* for copy serialization values, see defines */
   dsUint8_t    copyMode;       /* for copy mode values, see defines above */
   char         destName[DSM_MAX_CG_DEST_LENGTH + 1];      /* Copy dest name */
   dsmBool_t    bLanFreeDest;           /* Destination has lan free path?   */
   dsmBool_t    reserved;                             /* Not currently used */
   dsUint8_t    retainInit;              /* possible values see above        */
   dsUint16_t   retainMin;              /* if retInit is EVENT num of days */
   dsmBool_t    bDeduplicate;            /* destination has dedup enabled    */
}archDetailCG;

/*-------------------------------------------------------------------------+
|  Type definition for Backup Copy Group details on Query MC response      |
+-------------------------------------------------------------------------*/
typedef struct S_backupDetailCG
{
   char         cgName[DSM_MAX_CG_NAME_LENGTH + 1];       /* Copy group name */
   dsUint16_t   frequency;                               /* Backup frequency */
```

```
   dsUint16_t   verDataExst;                        /* Versions data exists */
   dsUint16_t   verDataDltd;                       /* Versions data deleted */
   dsUint16_t   retXtraVers;                       /* Retain extra versions */
   dsUint16_t   retOnlyVers;                        /* Retain only versions */
   dsUint8_t    copySer;     /* for copy serialization values, see defines */
   dsUint8_t    copyMode;       /* for copy mode values, see defines above */
   char         destName[DSM_MAX_CG_DEST_LENGTH + 1];      /* Copy dest name */
   dsmBool_t    bLanFreeDest;           /* Destination has lan free path?   */
   dsmBool_t    reserved;                          /* Not currently used */
   dsmBool_t    bDeduplicate;            /* destination has dedup enabled    */
}backupDetailCG;


/*-------------------------------------------------------------------------+
|  Type definition for Query Mgmt Class detail response on dsmGetNextQObj()|
+-------------------------------------------------------------------------*/
typedef struct S_qryRespMCDetailData
{
   dsUint16_t         stVersion;               /* structure version              */
   char            mcName[DSM_MAX_MC_NAME_LENGTH + 1];            /* mc name */
   char            mcDesc[DSM_MAX_MC_DESCR_LENGTH + 1]; /*mc description */
   archDetailCG    archDet;                 /* Archive copy group detail */
   backupDetailCG  backupDet;                /* Backup copy group detail */
}qryRespMCDetailData;

#define qryRespMCDetailDataVersion  4

/*-------------------------------------------------------------------------+
| Type definition for Query Mgmt Class summary response on dsmGetNextQObj()|
+-------------------------------------------------------------------------*/
typedef struct S_qryRespMCData
{
   dsUint16_t         stVersion;               /* structure version              */
   char            mcName[DSM_MAX_MC_NAME_LENGTH + 1];            /* mc name */
   char         mcDesc[DSM_MAX_MC_DESCR_LENGTH + 1];       /* mc description */
}qryRespMCData;

#define qryRespMCDataVersion  1



/*-------------------------------------------------------------------------+
|  Type definition for Archive queryBuffer on dsmBeginQuery()              |
+-------------------------------------------------------------------------*/
typedef struct S_qryArchiveData
{
   dsUint16_t         stVersion;               /* structure version              */
   dsmObjName   *objName;                     /* Full dsm name of object */
   char         *owner;                                  /* owner name */
                       /* for maximum date boundaries, see defines above */
   dsmDate      insDateLowerBound;      /* low bound archive insert date */
   dsmDate      insDateUpperBound;       /* hi bound archive insert date */
   dsmDate      expDateLowerBound;          /* low bound expiration date */
   dsmDate      expDateUpperBound;           /* hi bound expiration date */
   char         *descr;                   /* archive description */
} qryArchiveData;

#define qryArchiveDataVersion  1



/*=== values for retentionInitiated field ===*/
#define DSM_ARCH_RETINIT_UNKNOWN 0 /* ret init is unknown (down-level srv)  */
#define DSM_ARCH_RETINIT_STARTED 1 /* retention clock is started            */
#define DSM_ARCH_RETINIT_PENDING 2 /* retention clock is not started        */


/*=== Values for objHeld ===*/
#define DSM_ARCH_HELD_UNKNOWN 0    /* unknown hold status (down-level srv)  */
#define DSM_ARCH_HELD_FALSE   1    /* object is NOT in a delete hold state  */
```

```
#define DSM_ARCH_HELD_TRUE    2    /* object is in a delete hold state      */

/*-------------------------------------------------------------------------+
| Type definition for Query Archive response on dsmGetNextQObj()           |
+-------------------------------------------------------------------------*/
typedef struct S_qryRespArchiveData
{
   dsUint16_t         stVersion;               /* structure version              */
   dsmObjName      objName;                      /* Filespace name qualifier */
   dsUint32_t  copyGroup;                           /* copy group number */
   char            mcName[DSM_MAX_MC_NAME_LENGTH + 1];            /* mc name */
   char            owner[DSM_MAX_OWNER_LENGTH + 1];            /* owner name */
   dsStruct64_t    objId;                                    /* Unique copy id */
   dsStruct64_t    reserved;                        /* backward compatability */
   dsUint8_t       mediaClass;                          /* media access class */
   dsmDate         insDate;                         /* archive insertion date */
   dsmDate         expDate;                     /* expiration date for object */
   char            descr[DSM_MAX_DESCR_LENGTH + 1];   /* archive description */
   dsUint16_t  objInfolen;            /* length of object-dependent info*/
   char            reservedObjInfo[DSM_MAX_OBJINFO_LENGTH];  /*object-dependent info */
   dsUint160_t     restoreOrderExt;                           /* restore order */
   dsStruct64_t    sizeEstimate;              /* size estimate stored by user*/
   dsUint8_t       compressType;                           /* Compression flag*/
   dsUint8_t       retentionInitiated;   /* object waiting on retention event*/
   dsUint8_t       objHeld;  /*object is on retention "hold" see values above*/
   dsUint8_t       encryptionType;                     /* type of encryption */
   dsmBool_t       clientDeduplicated;             /* obj deduplicated by API*/
   char            objInfo[DSM_MAX_EXT_OBJINFO_LENGTH];  /*object-dependent info */
   char            compressAlg[DSM_MAX_COMPRESSTYPE_LENGTH + 1];  /* compression algorithm name */
}qryRespArchiveData;

#define qryRespArchiveDataVersion  7

/*-------------------------------------------------------------------------+
|  Type definition for Archive sendBuff parameter on dsmSendObj()          |
+-------------------------------------------------------------------------*/
typedef struct S_sndArchiveData
{
   dsUint16_t         stVersion;               /* structure version              */
   char         *descr;                   /* archive description */
}sndArchiveData;

#define sndArchiveDataVersion  1

/*-------------------------------------------------------------------------+
|  Type definition for Backup queryBuffer on dsmBeginQuery()               |
+-------------------------------------------------------------------------*/
typedef struct S_qryBackupData
{
   dsUint16_t         stVersion;               /* structure version              */
   dsmObjName  *objName;          /* full dsm name of object */
   char         *owner;                                  /* owner name */
   dsUint8_t   objState;          /* object state selector */
   dsmDate     pitDate;           /* Date value for point in time restore */
                                  /* for possible values, see defines above */
}qryBackupData;

#define qryBackupDataVersion  2

typedef struct
{
  dsUint8_t    reserved1;
  dsStruct64_t reserved2;
} reservedInfo_t;                 /* for future use */

/*-------------------------------------------------------------------------+
| Type definition for Query Backup response on dsmGetNextQObj()            |
+-------------------------------------------------------------------------*/
typedef struct S_qryRespBackupData
{
```

```
   dsUint16_t         stVersion;               /* structure version             */
   dsmObjName      objName;                       /* full dsm name of object */
   dsUint32_t  copyGroup;                           /* copy group number */
   char            mcName[DSM_MAX_MC_NAME_LENGTH + 1];            /* mc name */
   char            owner[DSM_MAX_OWNER_LENGTH + 1];             /* owner name */
   dsStruct64_t    objId;                                  /* Unique object id */
   dsStruct64_t    reserved;                        /* backward compatability */
   dsUint8_t       mediaClass;                          /* media access class */
   dsUint8_t       objState;                       /* Obj state, active, etc. */
   dsmDate     insDate;                          /* backup insertion date */
   dsmDate         expDate;                     /* expiration date for object */
   dsUint16_t  objInfolen;             /* length of object-dependent info*/
   char            reservedObjInfo[DSM_MAX_OBJINFO_LENGTH];  /*object-dependent info */
   dsUint160_t     restoreOrderExt;                          /* restore order */
   dsStruct64_t    sizeEstimate;             /* size estimate stored by user */
   dsStruct64_t    baseObjId;
   dsUint16_t      baseObjInfolen;             /* length of base object-dependent info*/
   dsUint8_t       baseObjInfo[DSM_MAX_OBJINFO_LENGTH];  /* base object-dependent info */
   dsUint160_t     baseRestoreOrder;                         /* restore order */
   dsUint32_t     fsID;
   dsUint8_t       compressType;
   dsmBool_t       isGroupLeader;
   dsmBool_t       isOpenGroup;
   dsUint8_t       reserved1;               /* for future use */
   dsmBool_t       reserved2;                  /* for future use */
   dsUint16_t      reserved3;               /* for future use */
   reservedInfo_t  *reserved4;              /* for future use */
   dsUint8_t       encryptionType;                    /* type of encryption */
   dsmBool_t       clientDeduplicated;             /* obj deduplicated by API*/
   char            objInfo[DSM_MAX_EXT_OBJINFO_LENGTH];  /*object-dependent info */
   char            compressAlg[DSM_MAX_COMPRESSTYPE_LENGTH + 1];  /* compression algorithm name */
}qryRespBackupData;

#define qryRespBackupDataVersion  8

/*--------------------------------------------------------------------------+
|  Type definition for Active Backup queryBuffer on dsmBeginQuery()
|
|  Notes:  For the active backup query, only the fs (filespace) and objType
|          fields of objName need be set.  objType can only be set to
|          DSM_OBJ_FILE or DSM_OBJ_DIRECTORY.  DSM_OBJ_ANY_TYPE will not
|          find a match on the query.
+--------------------------------------------------------------------------*/
typedef struct S_qryABackupData
{
   dsUint16_t         stVersion;               /* structure version             */
   dsmObjName      *objName;                    /* Only fs and objtype used */
}qryABackupData;

#define qryABackupDataVersion  1

/*--------------------------------------------------------------------------+
| Type definition for Query Active Backup response on dsmGetNextQObj()     |
+--------------------------------------------------------------------------*/
typedef struct S_qryARespBackupData
{
   dsUint16_t         stVersion;               /* structure version             */
   dsmObjName      objName;                       /* full dsm name of object */
   dsUint32_t  copyGroup;                           /* copy group number */
   char        mcName[DSM_MAX_MC_NAME_LENGTH + 1];/*management class name*/
   char            owner[DSM_MAX_OWNER_LENGTH + 1];             /* owner name */
   dsmDate     insDate;                          /* backup insertion date */
   dsUint16_t  objInfolen;             /* length of object-dependent info*/
   char        reservedObjInfo[DSM_MAX_OBJINFO_LENGTH];  /*object-dependent info */
   char        objInfo[DSM_MAX_EXT_OBJINFO_LENGTH];  /*object-dependent info */
}qryARespBackupData;

#define qryARespBackupDataVersion  2

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

```
|  Type definition for Backup queryBuffer on dsmBeginQuery()               |
+--------------------------------------------------------------------------*/
typedef struct qryBackupGroups
{
   dsUint16_t         stVersion;              /* structure version             */
   dsUint8_t    groupType;
   char         *fsName;
   char           *owner;
   dsStruct64_t groupLeaderObjId;
   dsUint8_t    objType;
   dsmBool_t    noRestoreOrder;
   dsmBool_t    noGroupInfo;
   char         *hl;
}qryBackupGroups;

#define qryBackupGroupsVersion 3


/*--------------------------------------------------------------------------+
|  Type definition for proxynode queryBuffer on dsmBeginQuery()             |
+--------------------------------------------------------------------------*/
typedef struct qryProxyNodeData
{
   dsUint16_t         stVersion;              /* structure version             */
   char        *targetNodeName;            /* target node name  */
}qryProxyNodeData;

#define qryProxyNodeDataVersion  1

/*--------------------------------------------------------------------------+
| Type definition for qryRespProxyNodeData  parameter used on dsmGetNextQObj()|
+--------------------------------------------------------------------------*/

typedef struct
{
   dsUint16_t        stVersion ;                    /* structure version      */
   char            targetNodeName[DSM_MAX_ID_LENGTH+1]; /* target node name  */
   char            peerNodeName[DSM_MAX_ID_LENGTH+1];   /* Peer node name    */
   char            hlAddress[DSM_MAX_ID_LENGTH+1];      /* peer hlAddress    */
   char            llAddress[DSM_MAX_ID_LENGTH+1];      /* peer hlAddress    */
}qryRespProxyNodeData;

#define qryRespProxyNodeDataVersion  1


/*--------------------------------------------------------------------------+
|  Type definition for WINNT and OS/2 Filespace attributes                  |
+--------------------------------------------------------------------------*/
typedef struct
{
   char         driveLetter ;          /* drive letter for filespace    */
   dsUint16_t   fsInfoLength;          /* fsInfo length used            */
   char         fsInfo[DSM_MAX_FSINFO_LENGTH];/*caller-determined data  */
}dsmDosFSAttrib ;

/*--------------------------------------------------------------------------+
|  Type definition for UNIX Filespace attributes                            |
+--------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t   fsInfoLength;          /* fsInfo length used            */
   char         fsInfo[DSM_MAX_FSINFO_LENGTH];/*caller-determined data  */
}dsmUnixFSAttrib ;

/*--------------------------------------------------------------------------+
|  Type definition for NetWare Filespace attributes                         |
+--------------------------------------------------------------------------*/
typedef dsmUnixFSAttrib dsmNetwareFSAttrib;

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

```
|  Type definition for Filespace attributes on all Filespace calls         |
+--------------------------------------------------------------------------*/
typedef union
{
   dsmNetwareFSAttrib  netwareFSAttr;
   dsmUnixFSAttrib     unixFSAttr ;
   dsmDosFSAttrib      dosFSAttr ;
}dsmFSAttr ;

/*--------------------------------------------------------------------------+
|  Type definition for fsUpd parameter on dsmUpdateFS()
+--------------------------------------------------------------------------*/
typedef struct S_dsmFSUpd
{
   dsUint16_t        stVersion ;                    /* structure version      */
   char            *fsType ;              /* filespace type                 */
   dsStruct64_t    occupancy ;            /* occupancy estimate             */
   dsStruct64_t    capacity  ;            /* capacity estimate              */
   dsmFSAttr       fsAttr ;                   /* platform specific attributes */
}dsmFSUpd ;

#define dsmFSUpdVersion  1

/*--------------------------------------------------------------------------+
|  Type definition for Filespace queryBuffer on dsmBeginQuery()            |
+--------------------------------------------------------------------------*/
typedef struct S_qryFSData
{
   dsUint16_t         stVersion;              /* structure version             */
   char        *fsName;                  /* File space name */
}qryFSData;

#define qryFSDataVersion  1

/*--------------------------------------------------------------------------+
| Type definition for Query Filespace response on dsmGetNextQObj()         |
+--------------------------------------------------------------------------*/
typedef struct S_qryRespFSData
{
   dsUint16_t         stVersion;              /* structure version             */
   char           fsName[DSM_MAX_FSNAME_LENGTH + 1];         /* Filespace name */
   char           fsType[DSM_MAX_FSTYPE_LENGTH + 1] ;        /* Filespace type */
   dsStruct64_t    occupancy;                     /* Occupancy est. in bytes. */
   dsStruct64_t    capacity;                      /* Capacity est. in bytes. */
   dsmFSAttr       fsAttr ;                   /* platform specific attributes */
   dsmDate        backStartDate;           /* start backup date             */
   dsmDate        backCompleteDate;        /* end backup Date               */
   dsmDate        reserved1;                 /* For future use               */
   dsmDate        lastReplStartDate;        /* The last time replication was started */
   dsmDate        lastReplCmpltDate;        /* The last time replication completed   */
                                            /*   (could have had a failure,          */
                                            /*    but it still completes)            */
   dsmDate        lastBackOpDateFromServer; /* The last store time stamp the client  */
                                            /*    saved on the server               */
   dsmDate        lastArchOpDateFromServer; /* The last store time stamp the client  */
                                            /*    saved on the server               */
   dsmDate        lastSpMgOpDateFromServer; /* The last store time stamp the client  */
                                            /*    saved on the server               */
   dsmDate        lastBackOpDateFromLocal;  /* The last store time stamp the client  */
                                            /*    saved on the Local                */
   dsmDate        lastArchOpDateFromLocal;  /* The last store time stamp the client  */
                                            /*    saved on the Local                */
   dsmDate        lastSpMgOpDateFromLocal;  /* The last store time stamp the client  */
                                            /*    saved on the Local                */
   dsInt32_t      failOverWriteDelay;       /* Minutes for client to wait before  allowed  */
                                            /* to store to this Repl srvr, Specail codes:  */
                                            /* NO_ACCESS(-1), ACCESS_RDONLY (-2)           */
}qryRespFSData;

#define qryRespFSDataVersion 4
```

```
/*-------------------------------------------------------------------------+
|  Type definition for regFilespace parameter on dsmRegisterFS()
+-------------------------------------------------------------------------*/
typedef struct S_regFSData
{
   dsUint16_t         stVersion;              /* structure version             */
   char            *fsName;                    /* Filespace name */
   char            *fsType;                    /* Filespace type */
   dsStruct64_t    occupancy;                  /* Occupancy est. in bytes. */
   dsStruct64_t    capacity;                    /* Capacity est. in bytes. */
   dsmFSAttr       fsAttr ;                  /* platform specific attributes */
}regFSData;

#define regFSDataVersion  1

/*-------------------------------------------------------------------------+
| Type definition for dedupType used in apisessInfo                        |
+-------------------------------------------------------------------------*/
typedef enum
{
   dedupServerOnly= 0x00,             /* dedup only done on server    */
   dedupClientOrServer               /* dedup can be done on client or server */
}dsmDedupType ;

/*-------------------------------------------------------------------------+
 | Type definition for fail over configuration and status
 -------------------------------------------------------------------------*/
typedef enum
{
   failOvrNotConfigured = 0x00,
   failOvrConfigured,
   failOvrConnectedToReplServer
}dsmFailOvrCfgType ;

/*-------------------------------------------------------------------------+
|  Type definition for session info response on dsmQuerySessionInfo()      |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t   stVersion;               /* Structure version                */
      /*-----------------------------------------------------------------*/
      /*           Server information                                    */
      /*-----------------------------------------------------------------*/
   char           serverHost[DSM_MAX_SERVERNAME_LENGTH+1];
                                  /* Network host name of DSM server  */
   dsUint16_t     serverPort;             /* Server comm port on host         */
   dsmDate        serverDate;             /* Server's date/time               */
   char           serverType[DSM_MAX_SERVERTYPE_LENGTH+1];
                                 /* Server's execution platform      */
   dsUint16_t     serverVer;              /* Server's version number          */
   dsUint16_t     serverRel;              /* Server's release number          */
   dsUint16_t     serverLev;              /* Server's level number            */
   dsUint16_t     serverSubLev;           /* Server's sublevel number         */
      /*-----------------------------------------------------------------*/
      /*           Client Defaults                                       */
      /*-----------------------------------------------------------------*/
   char           nodeType[DSM_MAX_PLATFORM_LENGTH+1]; /*node/application type*/
   char           fsdelim;                /* File space delimiter             */
   char           hldelim;                /* Delimiter betw highlev & lowlev  */
   dsUint8_t      compression;            /* Compression flag                 */
   dsUint8_t      archDel;                /* Archive delete permission        */
   dsUint8_t      backDel;                /* Backup  delete permission        */
   dsUint32_t     maxBytesPerTxn;         /* for future use                   */
   dsUint16_t     maxObjPerTxn;           /* The max objects allowed in a txn */
      /*-----------------------------------------------------------------*/
      /*           Session Information                                   */
      /*-----------------------------------------------------------------*/
   char       id[DSM_MAX_ID_LENGTH+1];    /* Sign-in id node name         */
   char       owner[DSM_MAX_OWNER_LENGTH+1]; /* Sign-in owner             */
```

```
                                    /*   (for multi-user platforms)     */
   char       confFile[DSM_PATH_MAX + DSM_NAME_MAX +1];
                                   /* len is platform dep             */
                                   /* dsInit name of appl config file  */
   dsUint8_t      opNoTrace;              /* dsInit option - NoTrace = 1      */
      /*---------------------------------------------------------------*/
      /*           Policy Data                                         */
      /*---------------------------------------------------------------*/
   char           domainName[DSM_MAX_DOMAIN_LENGTH+1]; /* Domain name         */
   char           policySetName[DSM_MAX_PS_NAME_LENGTH+1];
                                    /* Active policy set name           */
   dsmDate        polActDate;             /* Policy set activation date       */
   char           dfltMCName[DSM_MAX_MC_NAME_LENGTH+1];/* Default Mgmt Class  */
   dsUint16_t     gpBackRetn;             /* Grace-period backup retention    */
   dsUint16_t     gpArchRetn;             /* Grace-period archive retention   */
   char           adsmServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* adsm server name */
   dsmBool_t      archiveRetentionProtection; /* is server Retention protection enabled */
   dsStruct64_t   maxBytesPerTxn_64;         /* for future use                */
   dsmBool_t      lanFreeEnabled;           /* lan free option is set         */
   dsmDedupType   dedupType;                /* server or clientOrServer       */
   char           accessNode[DSM_MAX_ID_LENGTH+1];    /* as node node name         */

   /*---------------------------------------------------------------*/
   /*           Replication and fail over information               */
   /*---------------------------------------------------------------*/
   dsmFailOvrCfgType failOverCfgType; /* status of fail over */
   char           replServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* repl server name */
   char           homeServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* home server name */
   char           replServerHost[DSM_MAX_SERVERNAME_LENGTH+1]; /* Network host name of DSM server  */
   dsInt32_t      replServerPort;                              /* Server comm port on host         */

}ApiSessInfo;

#define ApiSessInfoVersion  6



/*-------------------------------------------------------------------------+
| Type definition for Query options response on dsmQueryCliOptions()       |
|      and dsmQuerySessOptions()                                           |
+-------------------------------------------------------------------------*/

typedef struct
{
   char            dsmiDir[DSM_PATH_MAX + DSM_NAME_MAX +1];
   char            dsmiConfig[DSM_PATH_MAX + DSM_NAME_MAX +1];
   char        serverName[DSM_MAX_SERVERNAME_LENGTH+1];
   dsInt16_t   commMethod;
   char        serverAddress[DSM_MAX_SERVER_ADDRESS];
   char        nodeName[DSM_MAX_NODE_LENGTH+1];
   dsmBool_t   compression;
   dsmBool_t   compressalways;
   dsmBool_t   passwordAccess;
}optStruct ;

/*-------------------------------------------------------------------------+
| Type definition for LogType used in logInfo                              |
+-------------------------------------------------------------------------*/
typedef enum
{
   logServer = 0x00,            /* log msg only to server    */
   logLocal,                    /* log msg only to local error log */
   logBoth,                      /* log msg to server and to local error log */
   logNone
}dsmLogType ;

/*-------------------------------------------------------------------------+
| Type definition for logInfo parameter used on dsmLogEvent()              |
+-------------------------------------------------------------------------*/
```

```
typedef struct
{
   char               *message;   /* text of message to be logged */
   dsmLogType         logType;    /* log type : local, server, both */
}logInfo;

/*-------------------------------------------------------------------------+
| Type definition for qryRespAccessData  parameter used on dsmQueryAccess()|
+-------------------------------------------------------------------------*/

typedef struct
{
   dsUint16_t        stVersion ;                    /* structure version      */
   char              node[DSM_MAX_ID_LENGTH+1];     /* node name              */
   char              owner[DSM_MAX_OWNER_LENGTH+1]; /* owner                  */
   dsmObjName        objName ;                      /* object name            */
   dsmAccessType     accessType;                    /* archive or backup     */
   dsUint32_t        ruleNumber ;                   /* Access rule id         */
}qryRespAccessData;

#define qryRespAccessDataVersion  1

/*-------------------------------------------------------------------------+
|  Type definition for envSetUp parameter on dsmSetUp()
+-------------------------------------------------------------------------*/
typedef struct S_envSetUp
{
   dsUint16_t         stVersion;              /* structure version             */
   char            dsmiDir[DSM_PATH_MAX + DSM_NAME_MAX +1];
   char            dsmiConfig[DSM_PATH_MAX + DSM_NAME_MAX +1];
   char            dsmiLog[DSM_PATH_MAX + DSM_NAME_MAX +1];
   char            **argv;  /* for executables name argv[0] */
   char            logName[DSM_NAME_MAX +1];
   dsmBool_t       reserved1;         /* for future use */
   dsmBool_t       reserved2;                 /* for future use */
}envSetUp;

#define envSetUpVersion  4

/*-------------------------------------------------------------------------+
|  Type definition for dsmInitExIn_t
+-------------------------------------------------------------------------*/
typedef struct dsmInitExIn_t
{
   dsUint16_t         stVersion;              /* structure version             */
   dsmApiVersionEx    *apiVersionExP;
   char               *clientNodeNameP;
   char               *clientOwnerNameP;
   char               *clientPasswordP;
   char               *userNameP;
   char               *userPasswordP;
   char               *applicationTypeP;
   char               *configfile;
   char               *options;
   char               dirDelimiter;
   dsmBool_t          useUnicode;
   dsmBool_t          bCrossPlatform;
   dsmBool_t          bService;
   dsmBool_t          bEncryptKeyEnabled;
   char               *encryptionPasswordP;
   dsmBool_t          useTsmBuffers;
   dsUint8_t          numTsmBuffers;
   dsmAppVersion      *appVersionP;
}dsmInitExIn_t;

#define dsmInitExInVersion 5

/*-------------------------------------------------------------------------+
|  Type definition for dsmInitExOut_t
+-------------------------------------------------------------------------*/
```

```
typedef struct dsmInitExOut_t
{
   dsUint16_t         stVersion;              /* structure version             */
   dsInt16_t          userNameAuthorities;
   dsInt16_t          infoRC;        /* error return code if encountered */
   char               adsmServerName[DSM_MAX_SERVERNAME_LENGTH+1];
   dsUint16_t     serverVer;              /* Server's version number          */
   dsUint16_t     serverRel;              /* Server's release number          */
   dsUint16_t     serverLev;              /* Server's level number            */
   dsUint16_t     serverSubLev;           /* Server's sublevel number         */

   dsmBool_t          bIsFailOverMode; /* true if failover has occured */
   char               replServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* repl server name */
   char               homeServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* home server name */

}dsmInitExOut_t;

#define dsmInitExOutVersion 3

/*--------------------------------------------------------------------------+
| Type definition for LogType used in logInfo                               |
+--------------------------------------------------------------------------*/
typedef enum
{
   logSevInfo = 0x00,       /* information ANE4991 */
   logSevWarning,           /* warning     ANE4992 */
   logSevError,             /* Error       ANE4993 */
   logSevSevere,            /* severe      ANE4994 */
   logSevLicense,           /* License     ANE4995 */
   logSevTryBuy             /* try Buy     ANE4996 */
}dsmLogSeverity ;


/*--------------------------------------------------------------------------+
|  Type definition for dsmLogExIn_t
+--------------------------------------------------------------------------*/
typedef struct dsmLogExIn_t
{
   dsUint16_t         stVersion;              /* structure version             */
   dsmLogSeverity     severity;
   char               appMsgID[8];
   dsmLogType         logType;    /* log type : local, server, both */
   char               *message;   /* text of message to be logged */
   char               appName[DSM_MAX_PLATFORM_LENGTH];
   char               osPlatform[DSM_MAX_PLATFORM_LENGTH];
   char               appVersion[DSM_MAX_PLATFORM_LENGTH];
}dsmLogExIn_t;

#define dsmLogExInVersion 2

/*--------------------------------------------------------------------------+
|  Type definition for dsmlogExOut_t
+--------------------------------------------------------------------------*/
typedef struct dsmLogExOut_t
{
   dsUint16_t         stVersion;              /* structure version             */
}dsmLogExOut_t;

#define dsmLogExOutVersion 1

/*--------------------------------------------------------------------------+
|  Type definition for dsmRenameIn_t
+--------------------------------------------------------------------------*/
typedef struct dsmRenameIn_t
{
   dsUint16_t         stVersion;              /* structure version             */
   dsUint32_t       dsmHandle;                  /* handle for session */
   dsUint8_t        repository;                 /* Backup or Archive  */
   dsmObjName       *objNameP ;                 /* object name            */
   char             newHl[DSM_MAX_HL_LENGTH + 1];  /* new High level name */
```

```
   char             newLl[DSM_MAX_LL_LENGTH + 1];  /* new Low level name */
   dsmBool_t        merge;                        /* merge into existing name*/
   ObjID            objId;                         /* objId for Archive */
}dsmRenameIn_t;

#define dsmRenameInVersion 1

/*--------------------------------------------------------------------------+
|  Type definition for dsmRenameOut_t
+--------------------------------------------------------------------------*/
typedef struct dsmRenameOut_t
{
   dsUint16_t         stVersion;              /* structure version             */
}dsmRenameOut_t;

#define dsmRenameOutVersion 1

/*--------------------------------------------------------------------------+
|  Type definition for dsmEndSendObjExIn_t
+--------------------------------------------------------------------------*/
typedef struct dsmEndSendObjExIn_t
{
   dsUint16_t         stVersion;              /* structure version             */
   dsUint32_t       dsmHandle;                    /* handle for session */
}dsmEndSendObjExIn_t;

#define dsmEndSendObjExInVersion 1

/*--------------------------------------------------------------------------+
|  Type definition for dsmEndSendObjExOut_t
+--------------------------------------------------------------------------*/
typedef struct dsmEndSendObjExOut_t
{
   dsUint16_t         stVersion;              /* structure version             */
   dsStruct64_t       totalBytesSent;      /* total bytes read from app */
   dsmBool_t          objCompressed;       /* was object compressed */
   dsStruct64_t       totalCompressSize;   /* total size after compress */
   dsStruct64_t       totalLFBytesSent;    /* total bytes sent Lan Free */
   dsUint8_t          encryptionType;      /* type of encryption used   */
   dsmBool_t          objDeduplicated;     /* was object processed for dist. data dedup */
   dsStruct64_t       totalDedupSize;      /* total size after de-dup */
}dsmEndSendObjExOut_t;

#define dsmEndSendObjExOutVersion 3
/*--------------------------------------------------------------------------+
|  Type definition for dsmGroupHandlerIn_t
+--------------------------------------------------------------------------*/
typedef struct dsmGroupHandlerIn_t
{
   dsUint16_t         stVersion;              /* structure version             */
   dsUint32_t       dsmHandle;                    /* handle for session */
   dsUint8_t        groupType;        /* Type of group                         */
   dsUint8_t        actionType;       /* Type of group operation               */
   dsUint8_t        memberType;       /* Type of member: Leader or member      */
   dsStruct64_t    leaderObjId;      /* OBJID of the groupleader when manipulating a member */
   char            *uniqueGroupTagP; /* Unique group identifier               */
   dsmObjName      *objNameP ;       /* group leader object name */
   dsmGetList       memberObjList;    /* list of objects to remove, assign     */
}dsmGroupHandlerIn_t;

#define dsmGroupHandlerInVersion 1

/*--------------------------------------------------------------------------+
|  Type definition for dsmGroupHandlerExOut_t
+--------------------------------------------------------------------------*/
typedef struct dsmGroupHandlerOut_t
{
   dsUint16_t         stVersion;              /* structure version             */
}dsmGroupHandlerOut_t;
```

```
#define dsmGroupHandlerOutVersion 1

/*-------------------------------------------------------------------------+
|  Type definition for dsmEndTxnExIn_t
+-------------------------------------------------------------------------*/
typedef struct dsmEndTxnExIn_t
{
   dsUint16_t         stVersion;              /* structure version             */
   dsUint32_t       dsmHandle;                    /* handle for session */
   dsUint8_t        vote;
}dsmEndTxnExIn_t;

#define dsmEndTxnExInVersion 1

/*-------------------------------------------------------------------------+
|  Type definition for dsmEndTxnExOut_t
+-------------------------------------------------------------------------*/
typedef struct dsmEndTxnExOut_t
{
   dsUint16_t         stVersion;              /* structure version             */
   dsUint16_t         reason;                 /* reason code                   */
   dsStruct64_t       groupLeaderObjId;       /* groupLeader obj id returned on */
                                              /* DSM_ACTION_OPEN               */
   dsUint8_t          reserved1;              /* future use                    */
   dsUint16_t         reserved2;              /* future use                    */
}dsmEndTxnExOut_t;

#define dsmEndTxnExOutVersion 1

/*-------------------------------------------------------------------------+
|  Type definition for dsmEndGetDataExIn_t
+-------------------------------------------------------------------------*/
typedef struct dsmEndGetDataExIn_t
{
   dsUint16_t         stVersion;              /* structure version             */
   dsUint32_t       dsmHandle;                    /* handle for session */
}dsmEndGetDataExIn_t;

#define dsmEndGetDataExInVersion 1

/*-------------------------------------------------------------------------+
|  Type definition for dsmEndGetDataExOut_t
+-------------------------------------------------------------------------*/
typedef struct dsmEndGetDataExOut_t
{
   dsUint16_t         stVersion;              /* structure version             */
   dsUint16_t         reason;                 /* reason code                   */
   dsStruct64_t   totalLFBytesRecv;  /* total lan free bytes recieved */
}dsmEndGetDataExOut_t;

#define dsmEndGetDataExOutVersion 1

/*-------------------------------------------------------------------------+
| Type definition for object list on dsmRetentionEvent()                   |
+-------------------------------------------------------------------------*/
typedef struct dsmObjList
{
   dsUint16_t         stVersion;              /* structure version             */
   dsUint32_t       numObjId;          /* number of object IDs in the list */
   ObjID            *objId;            /* list of object IDs to signal     */
}dsmObjList_t ;

#define dsmObjlistVersion 1

/*-------------------------------------------------------------------------+
| Type definition eventType used on dsmRetentionEvent                       |
+-------------------------------------------------------------------------*/
typedef enum
{
   eventRetentionActivate = 0x00,   /* signal the server that the event has occured */
```

```
   eventHoldObj,                   /* suspend delete/expire of the object          */
   eventReleaseObj                 /* Resume normal delete/expire processing       */
}dsmEventType_t;

/*-------------------------------------------------------------------------+
|  Type definition for  on dsmRetentionEvent()                             |
+-------------------------------------------------------------------------*/
typedef struct dsmRetentionEventIn_t
{
   dsUint16_t         stVersion;              /* structure version             */
   dsUint32_t         dsmHandle;                   /* session Handle     */
   dsmEventType_t     eventType;                   /* Event type         */
   dsmObjList_t       objList;                     /* object ID          */
}dsmRetentionEventIn_t;

#define dsmRetentionEventInVersion 1

/*-------------------------------------------------------------------------+
|  Type definition for  on dsmRetentionEvent()                |            |
+-------------------------------------------------------------------------*/
typedef struct dsmRetentionEventOut_t
{
   dsUint16_t        stVersion ;                  /* structure version      */
}dsmRetentionEventOut_t;

#define dsmRetentionEventOutVersion 1


/*-------------------------------------------------------------------------+
|  Type definition for  on dsmRequestBuffer()                               |
+-------------------------------------------------------------------------*/
typedef struct requestBufferIn_t
{
   dsUint16_t         stVersion;              /* structure version             */
   dsUint32_t         dsmHandle;                   /* session Handle     */
}requestBufferIn_t;

#define requestBufferInVersion 1

/*-------------------------------------------------------------------------+
|  Type definition for  on dsmRequestBuffer()                               |
+-------------------------------------------------------------------------*/
typedef struct requestBufferOut_t
{
   dsUint16_t        stVersion ;                  /* structure version      */
   dsUint8_t          tsmBufferHandle;       /* handle to tsm Data buffer */
   char               *dataPtr;                   /* Address to write data to */
   dsUint32_t       bufferLen;                /* Max length of data to be written */
}requestBufferOut_t;

#define requestBufferOutVersion 1

/*-------------------------------------------------------------------------+
|  Type definition for  on dsmReleaseBuffer()                              |
+-------------------------------------------------------------------------*/
typedef struct releaseBufferIn_t
{
   dsUint16_t         stVersion;              /* structure version             */
   dsUint32_t         dsmHandle;                   /* session Handle     */
   dsUint8_t          tsmBufferHandle;       /* handle to tsm Data buffer */
   char               *dataPtr;                   /* Address to write data to */
}releaseBufferIn_t;

#define releaseBufferInVersion 1

/*-------------------------------------------------------------------------+
|  Type definition for  on dsmReleaseBuffer()                              |
+-------------------------------------------------------------------------*/
typedef struct releaseBufferOut_t
{
```

```
   dsUint16_t        stVersion ;                    /* structure version      */
}releaseBufferOut_t;

#define releaseBufferOutVersion 1

/*------------------------------------------------------------------------+
|  Type definition for  on dsmGetBufferData()                             |
+------------------------------------------------------------------------*/
typedef struct getBufferDataIn_t
{
   dsUint16_t         stVersion;               /* structure version             */
   dsUint32_t         dsmHandle;                    /* session Handle     */
}getBufferDataIn_t;

#define getBufferDataInVersion 1

/*------------------------------------------------------------------------+
|  Type definition for  on dsmGetBufferData()                              |
+------------------------------------------------------------------------*/
typedef struct getBufferDataOut_t
{
   dsUint16_t        stVersion ;                    /* structure version      */
   dsUint8_t          tsmBufferHandle;       /* handle to tsm Data buffer */
   char             *dataPtr;              /* Address of actual data to read */
   dsUint32_t       numBytes;              /* Actual number of bytes to read from dataPtr*/
}getBufferDataOut_t;

#define getBufferDataOutVersion 1

/*------------------------------------------------------------------------+
|  Type definition for  on dsmSendBufferData()                            |
+------------------------------------------------------------------------*/
typedef struct sendBufferDataIn_t
{
   dsUint16_t         stVersion;               /* structure version             */
   dsUint32_t         dsmHandle;                    /* session Handle     */
   dsUint8_t          tsmBufferHandle;       /* handle to tsm Data buffer */
   char               *dataPtr;              /* Address of actual data to send */
   dsUint32_t         numBytes;              /* Actual number of bytes to send from dataPtr*/
}sendBufferDataIn_t;

#define sendBufferDataInVersion 1

/*------------------------------------------------------------------------+
|  Type definition for  on dsmSendBufferData()                            |
+------------------------------------------------------------------------*/
typedef struct sendBufferDataOut_t
{
   dsUint16_t        stVersion ;                    /* structure version      */
}sendBufferDataOut_t;

#define sendBufferDataOutVersion 1

/*------------------------------------------------------------------------+
|  Type definition for dsmUpdateObjExIn_t
+------------------------------------------------------------------------*/
typedef struct dsmUpdateObjExIn_t
{
   dsUint16_t         stVersion;               /* structure version             */
   dsUint32_t         dsmHandle;                    /* session Handle     */
   dsmSendType     sendType;             /* send type back/arch */
   char            *descrP;              /* archive description */
   dsmObjName      *objNameP;            /* objName             */
   ObjAttr         *objAttrPtr;          /* attribute           */
   dsUint32_t      objUpdAct;            /* update action       */
   ObjID           archObjId;            /* objId for archive   */
}dsmUpdateObjExIn_t;

#define dsmUpdateObjExInVersion 1
```

```
/*-------------------------------------------------------------------------+
|  Type definition for dsmUpdateObjExOut_t
+-------------------------------------------------------------------------*/
typedef struct dsmUpdateObjExOut_t
{
    dsUint16_t         stVersion;              /* structure version             */
}dsmUpdateObjExOut_t;

#define dsmUpdateObjExOutVersion 1

#if (_OPSYS_TYPE == DS_WINNT) && !defined(_WIN64)
#pragma pack()
#endif


#ifdef _MAC
#pragma options align = reset
#endif
#endif /* _H_DSMAPITD */
/*************************************************************************
 * Tivoli Storage Manager                                *
 * API Client Component                                  *
 *                                                                       *
 * (C) Copyright IBM Corporation 1993,2010                               *
 *************************************************************************/

/*************************************************************************
 * Header File Name: tsmapitd.h
 *
 * Environment:     ************************************************
 *                  ** This is a platform-independent source file **
 *
 *                  ************************************************
 *
 * Design Notes:    This file contains basic data types and constants
 *                  includable by all client source files. The constants
 *                  within this file should be set properly for the
 *                  particular machine and operating system on which the
 *                  client software is to be run.
 *
 *                  Platform specific definitions are included in dsmapips.h
 *
 * Descriptive-name: Definitions for Tivoli Storage manager API constants
 *-------------------------------------------------------------------------*/

#ifndef _H_TSMAPITD
#define _H_TSMAPITD


/*=== set the structure alignment to pack the structures ===*/
#if _OPSYS_TYPE == DS_WINNT
#ifdef _WIN64
#pragma pack(8)
#else
#pragma pack(1)
#endif
#endif

#ifdef _MAC
#pragma options align = packed
#endif

/*============================================================
 Win32 applications using the tsm interface must use the
 -DUNICODE flag during compilation.
 ============================================================*/
#if _OPSYS_TYPE == DS_WINNT && !defined(DSMAPILIB)
#ifndef UNICODE
```

```
#error "Win32 applications using the TSM interface MUST be compiled with the -DUNICODE flag"
#endif
#endif

/*============================================================
 Mac OS X applications using the tsm interface must use the
 -DUNICODE flag during compilation.
 ============================================================*/
#if _OPSYS_TYPE == DS_MACOS && !defined(DSMAPILIB)
#ifndef UNICODE
#error "Mac OS X applications using the TSM interface MUST be compiled with the -DUNICODE flag"
#endif
#endif

/*-------------------------------------------------------------------------+
 | Type definition for dsmGetType parameter on tsmBeginGetData()           |
 +-------------------------------------------------------------------------*/
typedef enum
{
   gtTsmBackup = 0x00,                    /* Backup processing type    */
   gtTsmArchive                           /* Archive processing type   */
} tsmGetType ;

/*-------------------------------------------------------------------------+
 |  Type definition for dsmQueryType parameter on tsmBeginQuery()          |
 +-------------------------------------------------------------------------*/
typedef enum
{
   qtTsmArchive = 0x00,                    /* Archive query type        */
   qtTsmBackup,                            /* Backup query type         */
   qtTsmBackupActive,                      /* Fast query for active backup files */
   qtTsmFilespace,                         /* Filespace query type      */
   qtTsmMC,                                /* Mgmt. class query type    */
    qtTsmReserved1,                         /* future use                */
    qtTsmReserved2,                         /* future use                */
    qtTsmReserved3,                         /* future use                */
    qtTsmReserved4,                         /* future use                */
   qtTsmBackupGroups,                      /* All group leaders in a specific filespace */
   qtTsmOpenGroups,                        /* All group members associated with a leader */
    qtTsmReserved5,                         /* future use                */
   qtTsmProxyNodeAuth,                     /* nodes that this node can proxy to   */
    qtTsmProxyNodePeer,                     /* peer nodes under this target node  */
    qtTsmReserved6,                         /* future use                */
    qtTsmReserved7,                         /* future use                */
    qtTsmReserved8                          /* future use                */
} tsmQueryType ;

/*-------------------------------------------------------------------------+
 |  Type definition sendType parameter on tsmBindMC() and tsmSendObj()     |
 +-------------------------------------------------------------------------*/
typedef enum
{
   stTsmBackup = 0x00,                        /* Backup processing type    */
   stTsmArchive,                              /* Archive processing type   */
   stTsmBackupMountWait,           /* Backup processing with mountwait on   */
   stTsmArchiveMountWait           /* Archive processing with mountwait on  */
} tsmSendType ;

/*-------------------------------------------------------------------------+
 |  Type definition for delType parameter on tsmDeleteObj()                |
 +-------------------------------------------------------------------------*/
typedef enum
{
   dtTsmArchive = 0x00,                             /* Archive delete type */
   dtTsmBackup,                           /* Backup delete (deactivate) type */
   dtTsmBackupID                         /* Backup delete (remove)     type */
} tsmDelType ;
```

```
/*-------------------------------------------------------------------------+
 |  Type definition sendType parameter on tsmSetAccess()                   |
 +-------------------------------------------------------------------------*/
typedef enum
{
   atTsmBackup = 0x00,                         /* Backup processing type    */
   atTsmArchive                               /* Archive processing type   */
}tsmAccessType;

/*-------------------------------------------------------------------------+
 |  Type definition for Overwrite  parameter on tsmSendObj()               |
 +-------------------------------------------------------------------------*/
typedef enum
{
   owIGNORE = 0x00,
   owYES,
   owNO
}tsmOwType;


/*-------------------------------------------------------------------------+
 |  Type definition for API Version on tsmInit() and tsmQueryApiVersion()  |
 +-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t stVersion;     /* Structure version               */
   dsUint16_t version;       /* API version                     */
   dsUint16_t release;       /* API release                     */
   dsUint16_t level;         /* API level                       */
   dsUint16_t subLevel;      /* API sub level                   */
   dsmBool_t  unicode;       /* API unicode?                    */
} tsmApiVersionEx;

#define tsmApiVersionExVer       2

/*-------------------------------------------------------------------------+
|  Type definition for Application Version on tsmInit()                    |
+-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t     stVersion;           /* Structure version               */
   dsUint16_t   applicationVersion;   /* application version number   */
   dsUint16_t   applicationRelease;   /* application release number   */
   dsUint16_t   applicationLevel;     /* application level number     */
   dsUint16_t   applicationSubLevel;  /* application sub level number */
} tsmAppVersion;

#define tsmAppVersionVer      1


/*-------------------------------------------------------------------------+
 |  Type definition for object name used on BindMC, Send, Delete, Query    |
 +-------------------------------------------------------------------------*/

typedef struct tsmObjName
{
   dsChar_t   fs[DSM_MAX_FSNAME_LENGTH + 1] ;              /* Filespace name */
   dsChar_t   hl[DSM_MAX_HL_LENGTH + 1] ;                 /* High level name */
   dsChar_t   ll[DSM_MAX_LL_LENGTH + 1] ;                  /* Low level name */
   dsUint8_t  objType;          /* for object type values, see defines above */
   dsChar_t           dirDelimiter;
} tsmObjName;

/*-------------------------------------------------------------------------+
 |  Type definition for Backup delete info on dsmDeleteObj()               |
 +-------------------------------------------------------------------------*/
```

```
typedef struct tsmDelBack
{
   dsUint16_t        stVersion ;                   /* structure version      */
   tsmObjName       *objNameP ;                   /* object name            */
   dsUint32_t        copyGroup ;                  /* copy group             */
} tsmDelBack ;

#define  tsmDelBackVersion    1

/*-------------------------------------------------------------------------+
 |  Type definition for Archive delete info on dsmDeleteObj()              |
 +-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t        stVersion ;                   /* structure version      */
   dsStruct64_t       objId ;                       /* object ID              */
} tsmDelArch ;

#define  tsmDelArchVersion  1

/*-------------------------------------------------------------------------+
 |  Type definition for Backup ID delete info on dsmDeleteObj()            |
 +-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t        stVersion ;                   /* structure version      */
   dsStruct64_t       objId ;                       /* object ID              */
} tsmDelBackID;

#define  tsmDelBackIDVersion  1


/*-------------------------------------------------------------------------+
 |  Type definition for delete info on dsmDeleteObj()                      |
 +-------------------------------------------------------------------------*/
typedef union
{
   tsmDelBack   backInfo ;
   tsmDelArch   archInfo ;
   tsmDelBackID backIDInfo;
} tsmDelInfo ;

/*-------------------------------------------------------------------------+
 |  Type definition for Object Attribute parameter on dsmSendObj()         |
 +-------------------------------------------------------------------------*/
typedef struct tsmObjAttr
{
   dsUint16_t   stVersion;               /* Structure version               */
   dsChar_t     owner[DSM_MAX_OWNER_LENGTH + 1];      /* object owner         */
   dsStruct64_t sizeEstimate;            /* Size estimate in bytes of the object */
   dsmBool_t    objCompressed;                  /* Is object already compressed?     */
   dsUint16_t   objInfoLength;           /* length of object-dependent info   */
   char         *objInfo;                /* object-dependent info byte buffer */
   dsChar_t     *mcNameP;                /* mgmnt class name for override     */
   tsmOwType    reserved1;               /* for future use                    */
   tsmOwType    reserved2;               /* for future use                    */
   dsmBool_t    disableDeduplication;        /* force no dedup for this object    */
   dsmBool_t    useExtObjInfo;            /* use ext objinfo up to 1536        */
} tsmObjAttr;

#define tsmObjAttrVersion 5

/*-------------------------------------------------------------------------+
 | Type definition for mcBindKey returned on dsmBindMC()                   |
 +-------------------------------------------------------------------------*/
typedef struct tsmMcBindKey
{
```

```
   dsUint16_t         stVersion;               /* structure version             */
   dsChar_t    mcName[DSM_MAX_MC_NAME_LENGTH + 1];
   /* Name of mc bound to object. */
   dsmBool_t   backup_cg_exists;                          /* True/false */
   dsmBool_t   archive_cg_exists;                         /* True/false */
   dsChar_t    backup_copy_dest[DSM_MAX_CG_DEST_LENGTH + 1];
   /* Backup copy dest. name */
   dsChar_t    archive_copy_dest[DSM_MAX_CG_DEST_LENGTH + 1];
   /* Arch copy dest.name */
} tsmMcBindKey;

#define tsmMcBindKeyVersion  1


/*-------------------------------------------------------------------------+
 |  Type definition for Mgmt Class queryBuffer on dsmBeginQuery()          |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryMCData
{
   dsUint16_t         stVersion;               /* structure version             */
   dsChar_t     *mcName;                       /* Mgmt class name */
   /* single name to get one or empty string to get all*/
   dsmBool_t    mcDetail;                           /* Want details or not? */
} tsmQryMCData;

#define tsmQryMCDataVersion  1


/*-------------------------------------------------------------------------+
 |  Type definition for Archive Copy Group details on Query MC response    |
 +-------------------------------------------------------------------------*/
typedef struct tsmArchDetailCG
{
   dsChar_t     cgName[DSM_MAX_CG_NAME_LENGTH + 1];       /* Copy group name */
   dsUint16_t   frequency;                     /* Copy (archive) frequency */
   dsUint16_t   retainVers;                              /* Retain version */
   dsUint8_t    copySer;     /* for copy serialization values, see defines */
   dsUint8_t    copyMode;       /* for copy mode values, see defines above */
   dsChar_t     destName[DSM_MAX_CG_DEST_LENGTH + 1];      /* Copy dest name */
   dsmBool_t    bLanFreeDest;           /* Destination has lan free path?    */
   dsmBool_t    reserved;                               /* Not currently used */
   dsUint8_t    retainInit;             /* possible values see dsmapitd.h  */
   dsUint16_t   retainMin;              /* if retInit is EVENT num of days */
   dsmBool_t    bDeduplicate;             /* destination has dedup enabled   */
}tsmArchDetailCG;

/*-------------------------------------------------------------------------+
 |  Type definition for Backup Copy Group details on Query MC response     |
 +-------------------------------------------------------------------------*/
typedef struct tsmBackupDetailCG
{
   dsChar_t     cgName[DSM_MAX_CG_NAME_LENGTH + 1];       /* Copy group name */
   dsUint16_t   frequency;                              /* Backup frequency */
   dsUint16_t   verDataExst;                         /* Versions data exists */
   dsUint16_t   verDataDltd;                        /* Versions data deleted */
   dsUint16_t   retXtraVers;                        /* Retain extra versions */
   dsUint16_t   retOnlyVers;                         /* Retain only versions */
   dsUint8_t    copySer;     /* for copy serialization values, see defines */
   dsUint8_t    copyMode;       /* for copy mode values, see defines above */
   dsChar_t     destName[DSM_MAX_CG_DEST_LENGTH + 1];      /* Copy dest name */
   dsmBool_t    bLanFreeDest;           /* Destination has lan free path?    */
   dsmBool_t    reserved;                               /* Not currently used */
   dsmBool_t    bDeduplicate;             /* destination has dedup enabled   */
}tsmBackupDetailCG;


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

```
 |  Type definition for Query Mgmt Class detail response on dsmGetNextQObj()|
 +-------------------------------------------------------------------------*/
typedef struct tsmQryRespMCDetailData
{
   dsUint16_t         stVersion;               /* structure version              */
   dsChar_t        mcName[DSM_MAX_MC_NAME_LENGTH + 1];           /* mc name */
   dsChar_t        mcDesc[DSM_MAX_MC_DESCR_LENGTH + 1]; /*mc description */
   archDetailCG    archDet;                /* Archive copy group detail */
   backupDetailCG  backupDet;               /* Backup copy group detail */
} tsmQryRespMCDetailData;

#define tsmQryRespMCDetailDataVersion  4

/*-------------------------------------------------------------------------+
 | Type definition for Query Mgmt Class summary response on dsmGetNextQObj()|
 +-------------------------------------------------------------------------*/
typedef struct tsmQryRespMCData
{
   dsUint16_t         stVersion;               /* structure version              */
   dsChar_t        mcName[DSM_MAX_MC_NAME_LENGTH + 1];           /* mc name */
   dsChar_t     mcDesc[DSM_MAX_MC_DESCR_LENGTH + 1];       /* mc description */
}tsmQryRespMCData;

#define tsmQryRespMCDataVersion  1

/*-------------------------------------------------------------------------+
 |  Type definition for Archive queryBuffer on tsmBeginQuery()              |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryArchiveData
{
   dsUint16_t         stVersion;               /* structure version              */
   tsmObjName   *objName;                     /* Full dsm name of object */
   dsChar_t     *owner;                                 /* owner name */
   /* for maximum date boundaries, see defines above */
   dsmDate      insDateLowerBound;      /* low bound archive insert date */
   dsmDate      insDateUpperBound;       /* hi bound archive insert date */
   dsmDate      expDateLowerBound;          /* low bound expiration date */
   dsmDate      expDateUpperBound;           /* hi bound expiration date */
   dsChar_t     *descr;                    /* archive description */
} tsmQryArchiveData;

#define tsmQryArchiveDataVersion  1

/*-------------------------------------------------------------------------+
| Type definition for Query Archive response on dsmGetNextQObj()           |
+-------------------------------------------------------------------------*/
typedef struct tsmQryRespArchiveData
{
    dsUint16_t         stVersion;               /* structure version              */
    tsmObjName      objName;                      /* Filespace name qualifier */
    dsUint32_t  copyGroup;                           /* copy group number */
    dsChar_t        mcName[DSM_MAX_MC_NAME_LENGTH + 1];           /* mc name */
    dsChar_t    owner[DSM_MAX_OWNER_LENGTH + 1];           /* owner name */
    dsStruct64_t    objId;                                 /* Unique copy id */
    dsStruct64_t    reserved;                        /* backward compatability */
    dsUint8_t       mediaClass;                          /* media access class */
    dsmDate         insDate;                        /* archive insertion date */
    dsmDate         expDate;                     /* expiration date for object */
    dsChar_t        descr[DSM_MAX_DESCR_LENGTH + 1];   /* archive description */
    dsUint16_t  objInfolen;             /* length of object-dependent info*/
    dsUint8_t       reservedObjInfo[DSM_MAX_OBJINFO_LENGTH];  /*object-dependent info */
    dsUint160_t     restoreOrderExt;                          /* restore order */
    dsStruct64_t    sizeEstimate;              /* size estimate stored by user*/
    dsUint8_t       compressType;                          /* Compression flag */
    dsUint8_t       retentionInitiated;   /* object waiting on retention event*/
    dsUint8_t       objHeld; /* object is on "hold" see dsmapitd.h for values */
    dsUint8_t       encryptionType;                    /* type of encryption */
```

```
    dsmBool_t       clientDeduplicated;             /* obj deduplicated by API*/
    dsUint8_t       objInfo[DSM_MAX_EXT_OBJINFO_LENGTH];  /*object-dependent info */
    dsChar_t        compressAlg[DSM_MAX_COMPRESSTYPE_LENGTH + 1];  /* compression algorithm name */
} tsmQryRespArchiveData;

#define tsmQryRespArchiveDataVersion  7

/*------------------------------------------------------------------------+
 |  Type definition for Archive sendBuff parameter on dsmSendObj()        |
 +------------------------------------------------------------------------*/
typedef struct tsmSndArchiveData
{
   dsUint16_t         stVersion;              /* structure version             */
   dsChar_t     *descr;                   /* archive description */
} tsmSndArchiveData;

#define tsmSndArchiveDataVersion  1

/*------------------------------------------------------------------------+
 |  Type definition for Backup queryBuffer on dsmBeginQuery()              |
 +------------------------------------------------------------------------*/
typedef struct tsmQryBackupData
{
   dsUint16_t         stVersion;              /* structure version             */
   tsmObjName  *objName;          /* full dsm name of object */
   dsChar_t     *owner;                                  /* owner name */
   dsUint8_t   objState;          /* object state selector */
   dsmDate     pitDate;           /* Date value for point in time restore */
   /* for possible values, see defines above */
    dsUint32_t  reserved1;
    dsUint32_t  reserved2;
} tsmQryBackupData;

#define tsmQryBackupDataVersion  3

/*------------------------------------------------------------------------+
 | Type definition for Query Backup response on dsmGetNextQObj()           |
 +------------------------------------------------------------------------*/
typedef struct tsmQryRespBackupData
{
   dsUint16_t         stVersion;              /* structure version             */
   tsmObjName  objName;                      /* full dsm name of object */
   dsUint32_t  copyGroup;                         /* copy group number */
   dsChar_t        mcName[DSM_MAX_MC_NAME_LENGTH + 1];           /* mc name */
   dsChar_t    owner[DSM_MAX_OWNER_LENGTH + 1];           /* owner name */
   dsStruct64_t    objId;                               /* Unique object id */
   dsStruct64_t    reserved;                       /* backward compatability */
   dsUint8_t       mediaClass;                         /* media access class */
   dsUint8_t       objState;                      /* Obj state, active, etc. */
   dsmDate     insDate;                        /* backup insertion date */
   dsmDate         expDate;                    /* expiration date for object */
   dsUint16_t  objInfolen;             /* length of object-dependent info*/
   dsUint8_t       reservedObjInfo[DSM_MAX_OBJINFO_LENGTH];  /*object-dependent info */
   dsUint160_t     restoreOrderExt;                        /* restore order */
   dsStruct64_t    sizeEstimate;            /* size estimate stored by user */
   dsStruct64_t    baseObjId;
   dsUint16_t      baseObjInfolen;             /* length of base object-dependent info*/
   dsUint8_t       baseObjInfo[DSM_MAX_OBJINFO_LENGTH];  /* base object-dependent info */
   dsUint160_t     baseRestoreOrder;                        /* restore order */
   dsUint32_t     fsID;
   dsUint8_t       compressType;
   dsmBool_t       isGroupLeader;
   dsmBool_t       isOpenGroup;
   dsUint8_t       reserved1;               /* for future use */
   dsmBool_t       reserved2;                  /* for future use */
   dsUint16_t      reserved3;               /* for future use */
   reservedInfo_t  *reserved4;              /* for future use */
```

```
   dsUint8_t       encryptionType;                    /* type of encryption */
   dsmBool_t       clientDeduplicated;             /* obj deduplicated by API*/
   dsUint8_t       objInfo[DSM_MAX_EXT_OBJINFO_LENGTH];  /*object-dependent info */
   dsChar_t        compressAlg[DSM_MAX_COMPRESSTYPE_LENGTH + 1];  /* compression algorithm name */
} tsmQryRespBackupData;

#define tsmQryRespBackupDataVersion  8
/*-------------------------------------------------------------------------+
 |  Type definition for Active Backup queryBuffer on dsmBeginQuery()
 |
 |  Notes:  For the active backup query, only the fs (filespace) and objType
 |          fields of objName need be set.  objType can only be set to
 |          DSM_OBJ_FILE or DSM_OBJ_DIRECTORY.  DSM_OBJ_ANY_TYPE will not
 |          find a match on the query.
 +-------------------------------------------------------------------------*/
typedef struct tsmQryABackupData
{
   dsUint16_t         stVersion;               /* structure version              */
   tsmObjName      *objName;                     /* Only fs and objtype used */
} tsmQryABackupData;

#define tsmQryABackupDataVersion  1

/*-------------------------------------------------------------------------+
 | Type definition for Query Active Backup response on dsmGetNextQObj()     |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryARespBackupData
{
   dsUint16_t         stVersion;               /* structure version              */
   tsmObjName  objName;                       /* full dsm name of object */
   dsUint32_t  copyGroup;                          /* copy group number */
   dsChar_t    mcName[DSM_MAX_MC_NAME_LENGTH + 1];/*management class name*/
   dsChar_t    owner[DSM_MAX_OWNER_LENGTH + 1];            /* owner name */
   dsmDate     insDate;                         /* backup insertion date */
   dsUint16_t  objInfolen;             /* length of object-dependent info*/
   dsUint8_t   reservedObjInfo[DSM_MAX_OBJINFO_LENGTH];  /*object-dependent info */
   dsUint8_t  objInfo[DSM_MAX_EXT_OBJINFO_LENGTH];  /*object-dependent info */
} tsmQryARespBackupData;

#define tsmQryARespBackupDataVersion  2

/*-------------------------------------------------------------------------+
 |  Type definition for Backup queryBuffer on dsmBeginQuery()               |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryBackupGroups
{
   dsUint16_t         stVersion;               /* structure version              */
   dsUint8_t    groupType;
   dsChar_t     *fsName;
   dsChar_t     *owner;
   dsStruct64_t groupLeaderObjId;
   dsUint8_t    objType;
   dsUint32_t   reserved1;
   dsUint32_t   reserverd2;
   dsmBool_t    noRestoreOrder;
   dsmBool_t    noGroupInfo;
   dsChar_t     *hl;
} tsmQryBackupGroups;

#define tsmQryBackupGroupsVersion 4

/*-------------------------------------------------------------------------+
 |  Type definition for proxynode queryBuffer on tsmBeginQuery()            |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryProxyNodeData
{
   dsUint16_t         stVersion;               /* structure version              */
```

```
   dsChar_t    *targetNodeName;            /* target node name      */
}tsmQryProxyNodeData;

#define tsmQryProxyNodeDataVersion  1

/*-------------------------------------------------------------------------+
 | Type definition for qryRespProxyNodeData  parameter used on tsmGetNextQObj()|
 +-------------------------------------------------------------------------*/

typedef struct tsmQryRespProxyNodeData
{
   dsUint16_t        stVersion ;                   /* structure version      */
   dsChar_t          targetNodeName[DSM_MAX_ID_LENGTH+1];  /* target node name  */
   dsChar_t          peerNodeName[DSM_MAX_ID_LENGTH+1];    /* peer node name    */
   dsChar_t          hlAddress[DSM_MAX_ID_LENGTH+1];       /* peer hlAddress    */
   dsChar_t          llAddress[DSM_MAX_ID_LENGTH+1];       /* peer llAddress    */
}tsmQryRespProxyNodeData;

#define tsmQryRespProxyNodeDataVersion  1

/*-------------------------------------------------------------------------+
 |  Type definition for WINNT and OS/2 Filespace attributes                |
 +-------------------------------------------------------------------------*/
typedef struct tsmDosFSAttrib
{
   osChar_t     driveLetter ;          /* drive letter for filespace    */
   dsUint16_t   fsInfoLength;          /* fsInfo length used            */
   osChar_t     fsInfo[DSM_MAX_FSINFO_LENGTH];/*caller-determined data  */
} tsmDosFSAttrib ;

/*-------------------------------------------------------------------------+
 |  Type definition for UNIX Filespace attributes                          |
 +-------------------------------------------------------------------------*/
typedef struct tsmUnixFSAttrib
{
   dsUint16_t   fsInfoLength;          /* fsInfo length used            */
   osChar_t     fsInfo[DSM_MAX_FSINFO_LENGTH];/*caller-determined data  */
} tsmUnixFSAttrib ;

/*-------------------------------------------------------------------------+
 |  Type definition for NetWare Filespace attributes                       |
 +-------------------------------------------------------------------------*/
typedef tsmUnixFSAttrib tsmNetwareFSAttrib;

/*-------------------------------------------------------------------------+
 |  Type definition for Filespace attributes on all Filespace calls        |
 +-------------------------------------------------------------------------*/
typedef union
{
   tsmNetwareFSAttrib  netwareFSAttr;
   tsmUnixFSAttrib     unixFSAttr ;
   tsmDosFSAttrib      dosFSAttr ;
} tsmFSAttr ;

/*-------------------------------------------------------------------------+
 |  Type definition for fsUpd parameter on dsmUpdateFS()
 +-------------------------------------------------------------------------*/
typedef struct   tsmFSUpd
{
   dsUint16_t        stVersion ;                   /* structure version      */
   dsChar_t        *fsType ;              /* filespace type                */
   dsStruct64_t    occupancy ;            /* occupancy estimate            */
   dsStruct64_t    capacity  ;            /* capacity estimate             */
   tsmFSAttr       fsAttr ;               /* platform specific attributes */
} tsmFSUpd ;

#define tsmFSUpdVersion  1
```

```
/*-------------------------------------------------------------------------+
 |  Type definition for Filespace queryBuffer on dsmBeginQuery()           |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryFSData
{
   dsUint16_t         stVersion;              /* structure version             */
   dsChar_t    *fsName;                  /* File space name */
} tsmQryFSData;

#define tsmQryFSDataVersion  1

/*-------------------------------------------------------------------------+
 | Type definition for Query Filespace response on dsmGetNextQObj()        |
 +-------------------------------------------------------------------------*/
typedef struct tsmQryRespFSData
{
   dsUint16_t         stVersion;              /* structure version             */
   dsChar_t       fsName[DSM_MAX_FSNAME_LENGTH + 1];     /* Filespace name */
   dsChar_t       fsType[DSM_MAX_FSTYPE_LENGTH + 1] ;     /* Filespace type */
   dsStruct64_t    occupancy;                      /* Occupancy est. in bytes.             */
   dsStruct64_t    capacity;                        /* Capacity est. in bytes.              */
   tsmFSAttr       fsAttr ;                /* platform specific attributes */
   dsmDate        backStartDate;           /* start backup date            */
   dsmDate        backCompleteDate;        /* end backup Date              */
    dsmDate        reserved1;                  /* For future use               */
   dsmBool_t      bIsUnicode;
   dsUint32_t     fsID;
   dsmDate        lastReplStartDate;        /* The last time replication was started */
   dsmDate        lastReplCmpltDate;        /* The last time replication completed   */
                                            /*   (could have had a failure,          */
                                            /*    but it still completes)            */
   dsmDate        lastBackOpDateFromServer; /* The last store time stamp the client  */
                                            /*    saved on the server               */
   dsmDate        lastArchOpDateFromServer; /* The last store time stamp the client  */
                                            /*    saved on the server               */
   dsmDate        lastSpMgOpDateFromServer; /* The last store time stamp the client  */
                                            /*    saved on the server               */
   dsmDate        lastBackOpDateFromLocal;  /* The last store time stamp the client  */
                                            /*    saved on the Local                */
   dsmDate        lastArchOpDateFromLocal;  /* The last store time stamp the client  */
                                            /*    saved on the Local                */
   dsmDate        lastSpMgOpDateFromLocal;  /* The last store time stamp the client  */
                                            /*    saved on the Local                */
   dsInt32_t      failOverWriteDelay;       /* Minutes for client to wait before  allowed  */
                                            /* to store to this Repl srvr, Specail codes:  */
                                            /* NO_ACCESS(-1), ACCESS_RDONLY (-2)           */
} tsmQryRespFSData;

#define tsmQryRespFSDataVersion 5

/*-------------------------------------------------------------------------+
 |  Type definition for regFilespace parameter on dsmRegisterFS()
 +-------------------------------------------------------------------------*/
typedef struct tsmRegFSData
{
   dsUint16_t         stVersion;              /* structure version             */
   dsChar_t        *fsName;                    /* Filespace name */
   dsChar_t        *fsType;                    /* Filespace type */
   dsStruct64_t    occupancy;                  /* Occupancy est. in bytes. */
   dsStruct64_t    capacity;                    /* Capacity est. in bytes. */
   tsmFSAttr       fsAttr ;                /* platform specific attributes */
} tsmRegFSData;

#define tsmRegFSDataVersion  1

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

```
 |  Type definition for session info response on dsmQuerySessionInfo()      |
 +-------------------------------------------------------------------------*/
typedef struct
{
   dsUint16_t   stVersion;               /* Structure version                */
   /*-----------------------------------------------------------------*/
   /*           Server information                                    */
   /*-----------------------------------------------------------------*/
   dsChar_t       serverHost[DSM_MAX_SERVERNAME_LENGTH+1];
   /* Network host name of DSM server  */
   dsUint16_t     serverPort;            /* Server comm port on host         */
   dsmDate        serverDate;            /* Server's date/time               */
   dsChar_t       serverType[DSM_MAX_SERVERTYPE_LENGTH+1];
   /* Server's execution platform      */
   dsUint16_t     serverVer;             /* Server's version number          */
   dsUint16_t     serverRel;             /* Server's release number          */
   dsUint16_t     serverLev;             /* Server's level number            */
   dsUint16_t     serverSubLev;          /* Server's sublevel number         */
   /*-----------------------------------------------------------------*/
   /*           Client Defaults                                       */
   /*-----------------------------------------------------------------*/
   dsChar_t       nodeType[DSM_MAX_PLATFORM_LENGTH+1]; /*node/application type*/
   dsChar_t       fsdelim;               /* File space delimiter             */
   dsChar_t       hldelim;               /* Delimiter betw highlev & lowlev  */
   dsUint8_t      compression;           /* Compression flag                 */
   dsUint8_t      archDel;               /* Archive delete permission        */
   dsUint8_t      backDel;               /* Backup  delete permission        */
   dsUint32_t     maxBytesPerTxn;        /* for future use                   */
   dsUint16_t     maxObjPerTxn;          /* The max objects allowed in a txn */
   /*-----------------------------------------------------------------*/
   /*           Session Information                                   */
   /*-----------------------------------------------------------------*/
   dsChar_t       id[DSM_MAX_ID_LENGTH+1];    /* Sign-in id node name         */
   dsChar_t       owner[DSM_MAX_OWNER_LENGTH+1]; /* Sign-in owner             */
   /*   (for multi-user platforms)     */
   dsChar_t       confFile[DSM_PATH_MAX + DSM_NAME_MAX +1];
   /* len is platform dep              */
   /* dsInit name of appl config file  */
   dsUint8_t      opNoTrace;             /* dsInit option - NoTrace = 1      */
   /*-----------------------------------------------------------------*/
   /*           Policy Data                                           */
   /*-----------------------------------------------------------------*/
   dsChar_t       domainName[DSM_MAX_DOMAIN_LENGTH+1]; /* Domain name         */
   dsChar_t       policySetName[DSM_MAX_PS_NAME_LENGTH+1];
   /* Active policy set name           */
   dsmDate        polActDate;            /* Policy set activation date       */
   dsChar_t       dfltMCName[DSM_MAX_MC_NAME_LENGTH+1];/* Default Mgmt Class  */
   dsUint16_t     gpBackRetn;            /* Grace-period backup retention    */
   dsUint16_t     gpArchRetn;            /* Grace-period archive retention   */
   dsChar_t       adsmServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* adsm server name */
  dsmBool_t      archiveRetentionProtection; /* is server Retention protection enabled */
   dsUint64_t     maxBytesPerTxn_64;         /* for future use                  */
   dsmBool_t      lanFreeEnabled;            /* lan free option is set         */
   dsmDedupType   dedupType;                 /* server or clientOrServer       */
   dsChar_t       accessNode[DSM_MAX_ID_LENGTH+1];    /* as node node name         */

   /*-----------------------------------------------------------------*/
   /*           Replication and fail over information                 */
   /*-----------------------------------------------------------------*/
   dsmFailOvrCfgType failOverCfgType; /* status of fail over */
   dsChar_t           replServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* repl server name */
   dsChar_t           homeServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* home server name */
   dsChar_t        replServerHost[DSM_MAX_SERVERNAME_LENGTH+1]; /* Network host name of DSM server  */
   dsInt32_t      replServerPort;                              /* Server comm port on host         */

} tsmApiSessInfo;
```

```
#define tsmApiSessInfoVersion  6

/*--------------------------------------------------------------------------+
 | Type definition for Query options response on dsmQueryCliOptions()       |
 |      and dsmQuerySessOptions()                                           |
 +--------------------------------------------------------------------------*/

typedef struct
{
   dsUint16_t  stVersion;
   dsChar_t        dsmiDir[DSM_PATH_MAX + DSM_NAME_MAX +1];
   dsChar_t        dsmiConfig[DSM_PATH_MAX + DSM_NAME_MAX +1];
   dsChar_t    serverName[DSM_MAX_SERVERNAME_LENGTH+1];
   dsInt16_t   commMethod;
   dsChar_t    serverAddress[DSM_MAX_SERVER_ADDRESS];
   dsChar_t    nodeName[DSM_MAX_NODE_LENGTH+1];
   dsmBool_t   compression;
   dsmBool_t   compressalways;
   dsmBool_t   passwordAccess;
}tsmOptStruct ;

#define tsmOptStructVersion  1

/*--------------------------------------------------------------------------+
 | Type definition for qryRespAccessData  parameter used on dsmQueryAccess()|
 +--------------------------------------------------------------------------*/

typedef struct
{
   dsUint16_t        stVersion ;                   /* structure version      */
   dsChar_t          node[DSM_MAX_ID_LENGTH+1];    /* node name              */
   dsChar_t          owner[DSM_MAX_OWNER_LENGTH+1]; /* owner                 */
   tsmObjName        objName ;                     /* object name            */
   dsmAccessType     accessType;                   /* archive or backup     */
   dsUint32_t        ruleNumber ;                  /* Access rule id         */

}tsmQryRespAccessData;

#define tsmQryRespAccessDataVersion  1

/*--------------------------------------------------------------------------+
 |  Type definition for envSetUp parameter on dsmSetUp()
 +--------------------------------------------------------------------------*/
typedef struct tsmEnvSetUp
{
   dsUint16_t         stVersion;               /* structure version             */
   dsChar_t        dsmiDir[DSM_PATH_MAX + DSM_NAME_MAX +1];
   dsChar_t        dsmiConfig[DSM_PATH_MAX + DSM_NAME_MAX +1];
   dsChar_t        dsmiLog[DSM_PATH_MAX + DSM_NAME_MAX +1];
   char            **argv;  /* for executables name argv[0] */
   dsChar_t        logName[DSM_NAME_MAX +1];
   dsmBool_t       reserved1;         /* for future use */
   dsmBool_t       reserved2;                /* for future use */
} tsmEnvSetUp;

#define tsmEnvSetUpVersion  4

/*--------------------------------------------------------------------------+
 |  Type definition for dsmInitExIn_t
 +--------------------------------------------------------------------------*/
typedef struct tsmInitExIn_t
{
   dsUint16_t         stVersion;               /* structure version             */
   tsmApiVersionEx    *apiVersionExP;
   dsChar_t           *clientNodeNameP;
   dsChar_t           *clientOwnerNameP;
   dsChar_t           *clientPasswordP;
```

```
   dsChar_t           *userNameP;
   dsChar_t           *userPasswordP;
   dsChar_t           *applicationTypeP;
   dsChar_t           *configfile;
   dsChar_t           *options;
   dsChar_t           dirDelimiter;
   dsmBool_t          useUnicode;
   dsmBool_t          bCrossPlatform;
   dsmBool_t          bService;
   dsmBool_t          bEncryptKeyEnabled;
   dsChar_t           *encryptionPasswordP;
   dsmBool_t          useTsmBuffers;
   dsUint8_t          numTsmBuffers;
   tsmAppVersion      appVersionP;
} tsmInitExIn_t;

#define tsmInitExInVersion 5

/*-------------------------------------------------------------------------+
 |  Type definition for dsmInitExOut_t
 +-------------------------------------------------------------------------*/
typedef struct tsmInitExOut_t
{
   dsUint16_t         stVersion;              /* structure version             */
   dsInt16_t          userNameAuthorities;
   dsInt16_t          infoRC;        /* error return code if encountered */
   /* adsm server name                */
   dsChar_t           adsmServerName[DSM_MAX_SERVERNAME_LENGTH+1];
   dsUint16_t     serverVer;              /* Server's version number          */
   dsUint16_t     serverRel;              /* Server's release number          */
   dsUint16_t     serverLev;              /* Server's level number            */
   dsUint16_t     serverSubLev;           /* Server's sublevel number         */
   dsmBool_t          bIsFailOverMode; /* true if failover has occured */
   dsChar_t           replServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* repl server name */
   dsChar_t           homeServerName[DSM_MAX_SERVERNAME_LENGTH+1]; /* home server name */
} tsmInitExOut_t;

#define tsmInitExOutVersion 3

/*-------------------------------------------------------------------------+
 |  Type definition for dsmLogExIn_t
 +-------------------------------------------------------------------------*/
typedef struct tsmLogExIn_t
{
   dsUint16_t         stVersion;              /* structure version             */
   dsmLogSeverity     severity;
   dsChar_t           appMsgID[8];
   dsmLogType         logType;    /* log type : local, server, both */
   dsChar_t           *message;   /* text of message to be logged */
   dsChar_t           appName[DSM_MAX_PLATFORM_LENGTH];
   dsChar_t           osPlatform[DSM_MAX_PLATFORM_LENGTH];
   dsChar_t           appVersion[DSM_MAX_PLATFORM_LENGTH];

} tsmLogExIn_t;

#define tsmLogExInVersion 2

/*-------------------------------------------------------------------------+
 |  Type definition for dsmlogExOut_t
 +-------------------------------------------------------------------------*/
typedef struct tsmLogExOut_t
{
   dsUint16_t         stVersion;              /* structure version             */
} tsmLogExOut_t;

#define tsmLogExOutVersion 1
```

```
/*-------------------------------------------------------------------------+
 |  Type definition for dsmRenameIn_t
 +-------------------------------------------------------------------------*/
typedef struct tsmRenameIn_t
{
   dsUint16_t         stVersion;               /* structure version             */
   dsUint32_t       tsmHandle;        /* handle for session                      */
   dsUint8_t        repository;                 /* Backup or Archive  */
   tsmObjName       *objNameP ;                 /* object name            */
   dsChar_t         newHl[DSM_MAX_HL_LENGTH + 1];  /* new High level name */
   dsChar_t         newLl[DSM_MAX_LL_LENGTH + 1];  /* new Low level name */
   dsmBool_t        merge;                     /* merge into existing name*/
   ObjID            objId;                      /* objId for Archive */
} tsmRenameIn_t;

#define tsmRenameInVersion 1

/*-------------------------------------------------------------------------+
 |  Type definition for dsmRenameOut_t
 +-------------------------------------------------------------------------*/
typedef struct tsmRenameOut_t
{
   dsUint16_t         stVersion;               /* structure version             */
} tsmRenameOut_t;

#define tsmRenameOutVersion 1

/*-------------------------------------------------------------------------+
 |  Type definition for tsmEndSendObjExIn_t
 +-------------------------------------------------------------------------*/
typedef struct tsmEndSendObjExIn_t
{
   dsUint16_t         stVersion;               /* structure version             */
   dsUint32_t       tsmHandle;        /* handle for session                      */
} tsmEndSendObjExIn_t;

#define tsmEndSendObjExInVersion 1

/*-------------------------------------------------------------------------+
 |  Type definition for dsmEndSendObjExOut_t
 +-------------------------------------------------------------------------*/
typedef struct tsmEndSendObjExOut_t
{
   dsUint16_t         stVersion;               /* structure version             */
   dsStruct64_t       totalBytesSent;      /* total bytes read from app */
   dsmBool_t          objCompressed;       /* was object compressed */
   dsStruct64_t       totalCompressSize;   /* total size after compress */
   dsStruct64_t       totalLFBytesSent;    /* total bytes sent Lan Free */
   dsUint8_t          encryptionType;      /* type of encryption used   */
   dsmBool_t          objDeduplicated;     /* was object processed for dist. data dedup */
   dsStruct64_t       totalDedupSize;      /* total size after de-dup */
}tsmEndSendObjExOut_t;

#define tsmEndSendObjExOutVersion 3

/*-------------------------------------------------------------------------+
 |  Type definition for tsmGroupHandlerIn_t
 +-------------------------------------------------------------------------*/
typedef struct tsmGroupHandlerIn_t
{
   dsUint16_t         stVersion;               /* structure version             */
   dsUint32_t       tsmHandle;        /* handle for session                      */
   dsUint8_t        groupType;        /* Type of group                           */
   dsUint8_t        actionType;       /* Type of group operation                 */
   dsUint8_t        memberType;       /* Type of member: Leader or member        */
   dsStruct64_t     leaderObjId;      /* OBJID of the groupleader                */
   dsChar_t         *uniqueGroupTagP; /* Unique group identifier                 */
```

```
   tsmObjName       *objNameP ;       /* group leader object name               */
   dsmGetList       memberObjList;    /* list of objects to remove, assign      */
} tsmGroupHandlerIn_t;

#define tsmGroupHandlerInVersion 1

/*-------------------------------------------------------------------------+
 |  Type definition for tsmGroupHandlerExOut_t
 +-------------------------------------------------------------------------*/
typedef struct tsmGroupHandlerOut_t
{
   dsUint16_t         stVersion;              /* structure version           */
} tsmGroupHandlerOut_t;

#define tsmGroupHandlerOutVersion 1

/*-------------------------------------------------------------------------+
 |  Type definition for tsmEndTxnExIn_t
 +-------------------------------------------------------------------------*/
typedef struct tsmEndTxnExIn_t
{
   dsUint16_t         stVersion;              /* structure version           */
   dsUint32_t       tsmHandle;        /* handle for session                   */
   dsUint8_t        vote;
} tsmEndTxnExIn_t;

#define tsmEndTxnExInVersion 1

/*-------------------------------------------------------------------------+
 |  Type definition for tsmEndTxnExOut_t
 +-------------------------------------------------------------------------*/
typedef struct tsmEndTxnExOut_t
{
   dsUint16_t         stVersion;              /* structure version           */
   dsUint16_t         reason;                 /* reason code                 */
   dsStruct64_t       groupLeaderObjId;       /* groupLeader obj id returned on */
   /* DSM_ACTION_OPEN                */
    dsUint8_t          reserved1;             /* future use                  */
    dsUint16_t         reserved2;             /* future use                  */
} tsmEndTxnExOut_t;

#define tsmEndTxnExOutVersion 1

/*-------------------------------------------------------------------------+
 |  Type definition for tsmEndGetDataExIn_t
 +-------------------------------------------------------------------------*/
typedef struct tsmEndGetDataExIn_t
{
   dsUint16_t         stVersion;              /* structure version           */
   dsUint32_t       tsmHandle;        /* handle for session                   */
}tsmEndGetDataExIn_t;

#define tsmEndGetDataExInVersion 1

/*-------------------------------------------------------------------------+
 |  Type definition for tsmEndGetDataExOut_t
 +-------------------------------------------------------------------------*/
typedef struct tsmEndGetDataExOut_t
{
   dsUint16_t         stVersion;              /* structure version           */
   dsUint16_t         reason;                 /* reason code                 */
   dsStruct64_t   totalLFBytesRecv;  /* total lan free bytes recieved */
}tsmEndGetDataExOut_t;

#define tsmEndGetDataExOutVersion 1

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

```
 |  Type definition for  on tsmRetentionEvent()               |
 +-------------------------------------------------------------------------*/
typedef struct tsmRetentionEventIn_t
{
   dsUint16_t         stVersion;               /* structure version             */
   dsUint32_t         tsmHandle;                    /* session Handle     */
   dsmEventType_t     eventType;                    /* Event type         */
   dsmObjList_t       objList;                      /* object ID          */
}tsmRetentionEventIn_t;

#define tsmRetentionEventInVersion 1

/*-------------------------------------------------------------------------+
 |  Type definition for  on tsmRetentionEvent()               |
 +-------------------------------------------------------------------------*/
typedef struct tsmRetentionEventOut_t
{
   dsUint16_t        stVersion ;                   /* structure version      */
}tsmRetentionEventOut_t;

#define tsmRetentionEventOutVersion 1

/*-------------------------------------------------------------------------+
 |  Type definition for tsmUpdateObjExIn_t
 +-------------------------------------------------------------------------*/
typedef struct tsmUpdateObjExIn_t
{
   dsUint16_t         stVersion;               /* structure version             */
   dsUint32_t         tsmHandle;                    /* session Handle     */
   tsmSendType     sendType;             /* send type back/arch */
   dsChar_t        *descrP;              /* archive description */
   tsmObjName      *objNameP;            /* objName             */
   tsmObjAttr      *objAttrPtr;          /* attribute           */
   dsUint32_t      objUpdAct;            /* update action       */
   ObjID           archObjId;            /* objId for archive   */
}tsmUpdateObjExIn_t;

#define tsmUpdateObjExInVersion 1

/*-------------------------------------------------------------------------+
 |  Type definition for tsmUpdateObjExOut_t
 +-------------------------------------------------------------------------*/
typedef struct tsmUpdateObjExOut_t
{
   dsUint16_t         stVersion;               /* structure version             */
}tsmUpdateObjExOut_t;

#define tsmUpdateObjExOutVersion 1

#if _OPSYS_TYPE == DS_WINNT
#pragma pack()
#endif


#ifdef _MAC
#pragma options align = reset
#endif
#endif /* _H_TSMAPITD */
/*************************************************************************
* Tivoli Storage Manager                                 *
* API Client Component                                   *
*                                                                        *
* (C) Copyright IBM Corporation 1993,2010                                *
*************************************************************************/

/*************************************************************************
* Header File Name: dsmapips.h
```

```
*
* Environment:      *********************************************
*                   ** This is a platform-specific source file **
*                   ** versioned for Windows NT                **
*
*                   *********************************************
*
* Design Notes:     This file includes platform dependent definitions
*
* Descriptive-name: Definitions for Tivoli Storage Manager typedefs and LINKAGE
*-------------------------------------------------------------------------*/

#ifndef _H_DSMAPIPS
#define _H_DSMAPIPS


#ifndef _WIN64
#pragma pack(1)
#endif

/*<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>*/
/*                     T Y P E D E F S                                    */
/*<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>*/

/* new typedef file for Version 3   */


#if !defined(DSMAPILIB) || defined (XOPEN_BUILD)

/* support for linkage */
#include <windows.h>
#define DSMLINKAGE WINAPI


#define DS_WINNT    22
#define _OPSYS_TYPE DS_WINNT

   typedef signed   char  dsInt8_t;
   typedef unsigned char  dsUint8_t;
   typedef signed   short dsInt16_t;
   typedef unsigned short dsUint16_t;
   typedef signed   long  dsInt32_t;
   typedef unsigned long  dsUint32_t;

/*=== Character and string types ===*/
#ifdef UNICODE
  typedef wchar_t dsChar_t;
  #define dsTEXT(x)       L##x
#else
  typedef char dsChar_t;
  #define dsTEXT(x)       x
#endif  /* !UNICODE */

/*=== Common typedefs and defines derived from dsChar_t ===*/
typedef dsChar_t            *dsString_t;

  /* added for the extended restore order */
   typedef struct
   {
     dsUint32_t top;
     dsUint32_t hi_hi;
     dsUint32_t hi_lo;
     dsUint32_t lo_hi;
     dsUint32_t lo_lo;
   } dsUint160_t ;

#if defined(_LONG_LONG)
```

```
   typedef __int64              dsInt64_t;
   typedef unsigned __int64     dsUint64_t;
   /*=== A "true" unsigned 64-bit integer ===*/
   typedef __int64              dsLongLong_t;
#else
typedef struct tagUINT64_t
   {
      dsUint32_t hi;        /* Most significant 32 bits. */
      dsUint32_t lo;        /* Least significant 32 bits. */
   } dsUint64_t;
#endif

/*-------------------------------------------------------------------------+
| Type definition for bool_t                                               |
+-------------------------------------------------------------------------*/
/*
 *  Had to create a Boolean type that didn't clash with any other predefined
 *  version in any operating system or windowing system.
 */
typedef enum
{
   dsmFalse = 0x00,
   dsmTrue  = 0x01
}dsmBool_t ;


/*===  for backward compatability  ===*/
#define uint8  dsUint8_t
#define int8   dsInt8_t
#define uint16 dsUint16_t
#define int16  dsInt16_t
#define uint32 dsUint32_t
#define int32  dsInt32_t
#define uint64 dsStruct64_t
#define bool_t dsBool_t
#define dsBool_t dsmBool_t
#define bTrue    dsmTrue
#define bFalse   dsmFalse

typedef struct
{
   dsUint32_t hi;        /* Most significant 32 bits. */
   dsUint32_t lo;        /* Least significant 32 bits. */
}dsStruct64_t ;

#endif /* DSMAPILIB */



#ifndef _WIN64
#pragma pack()
#endif
#endif  /* _H_DSMAPIPS */
/*************************************************************************
* Tivoli Storage Manager                                *
* Common Source Component                                                *
*                                                                        *
* (C) Copyright IBM Corporation 1993,2016                                *
*************************************************************************/

/*************************************************************************
* Header File Name: release.h
*
* Environment:     ***********************************************
*                  ** This is a platform-independent source file **
*                  ***********************************************
*
```

```
* Design Notes:    This file contains the common information about
*                  the actual version.release.level.sublevel
*
* Descriptive-name: Definitions for Tivoli Storage manager version
*
* Note: This file should contain no LOG or CMVC information. It is
*       shipped with the API code.
*
*-----------------------------------------------------------------------*/

#ifndef _H_RELEASE
#define _H_RELEASE

#define COMMON_VERSION         8
#define COMMON_RELEASE         1
#define COMMON_LEVEL           0
#define COMMON_SUBLEVEL        0
#define COMMON_DRIVER          dsTEXT("")

#define COMMON_VERSIONTXT "8.1.0.0"

#define SHIPYEARTXT  "2016"
#define SHIPYEARTXTW dsTEXT("2016")
#define TSMPRODTXT  "IBM Tivoli Storage Manager"

/*=====================================================================
   The following string definitions are used for VERSION information
   and should not be converted to dsTEXT or osTEXT.  They are used
   only at link time.

   These are also used when the Jar file is built on Unix.  See the
   the perl script tools/unx/mzbuild/createReleaseJava
=====================================================================*/
#define COMMON_VERSION_STR    "8"
#define COMMON_RELEASE_STR    "1"
#define COMMON_LEVEL_STR      "0"
#define COMMON_SUBLEVEL_STR   "0"
#define COMMON_DRIVER_STR     ""

/*=== product names definitions ===*/
#define COMMON_NAME_DFDSM      1
#define COMMON_NAME_ADSM       2
#define COMMON_NAME_TSM        3
#define COMMON_NAME_ITSM       4
#define COMMON_NAME            COMMON_NAME_ITSM

/*=====================================================================
   Internal version, release, and level (build) version.  This
   should be unique for every version+release+ptf of a product.
   This information is recorded in the file attributes and data
   stream for diagnostic purposes.
   NOTE: DO NOT MODIFY THESE VALUES. YOU CAN ONLY ADD NEW ENTRIES!
=====================================================================*/
#define COMMON_BUILD_TSM_510   1
#define COMMON_BUILD_TSM_511   2
#define COMMON_BUILD_TSM_515   3
#define COMMON_BUILD_TSM_516   4
#define COMMON_BUILD_TSM_520   5
#define COMMON_BUILD_TSM_522   6
#define COMMON_BUILD_TSM_517   7
#define COMMON_BUILD_TSM_523   8
#define COMMON_BUILD_TSM_530   9
#define COMMON_BUILD_TSM_524   10
#define COMMON_BUILD_TSM_532   11
#define COMMON_BUILD_TSM_533   12
#define COMMON_BUILD_TSM_525   13
#define COMMON_BUILD_TSM_534   14
```

```
#define COMMON_BUILD_TSM_540   15
#define COMMON_BUILD_TSM_535   16
#define COMMON_BUILD_TSM_541   17
#define COMMON_BUILD_TSM_550   18
#define COMMON_BUILD_TSM_542   19
#define COMMON_BUILD_TSM_551   20
#define COMMON_BUILD_TSM_610   21
#define COMMON_BUILD_TSM_552   22
#define COMMON_BUILD_TSM_611   23
#define COMMON_BUILD_TSM_543   24
#define COMMON_BUILD_TSM_620   25
#define COMMON_BUILD_TSM_612   26
#define COMMON_BUILD_TSM_553   27
#define COMMON_BUILD_TSM_613   28
#define COMMON_BUILD_TSM_621   29
#define COMMON_BUILD_TSM_622   30
#define COMMON_BUILD_TSM_614   31
#define COMMON_BUILD_TSM_623   32
#define COMMON_BUILD_TSM_630   33
#define COMMON_BUILD_TSM_615   34
#define COMMON_BUILD_TSM_624   35
#define COMMON_BUILD_TSM_631   36
#define COMMON_BUILD_TSM_640   37
#define COMMON_BUILD_TSM_710   38
#define COMMON_BUILD_TSM_625   39
#define COMMON_BUILD_TSM_641   40
#define COMMON_BUILD_TSM_711   41
#define COMMON_BUILD_TSM_712   42
#define COMMON_BUILD_TSM_713   43
#define COMMON_BUILD_TSM_714   44
#define COMMON_BUILD_TSM_720   45
#define COMMON_BUILD_TSM_721   46
#define COMMON_BUILD_TSM_642   47
#define COMMON_BUILD_TSM_643   48
#define COMMON_BUILD_TSM_715   49
#define COMMON_BUILD_TSM_716   50
#define COMMON_BUILD_TSM_810   51
#define COMMON_BUILD           COMMON_BUILD_TSM_810

/*=== define VRL as an Int for bitmap version compares ====*/
static const  int  VRL_712 = 712;
static const  int  VRL_713 = 713;
static const  int  VRL_714 = 714;
static const  int  VRL_715 = 715;
static const  int  VRL_716 = 716;
static const  int  VRL_810 = 810;

#define   TDP4VE_PLATFORM_STRING_MBCS     "TDP VMware"
#define   TDP4VE_PLATFORM_STRING          dsTEXT("TDP VMware")

#define   TDP4HYPERV_PLATFORM_STRING_MBCS "TDP HyperV"
#define   TDP4HYPERV_PLATFORM_STRING      dsTEXT("TDP HyperV")

#endif /* _H_RELEASE */
```

Annexe C. Fichier source des définitions de type d'API

Cette annexe contient le fichier d'en-tête `dsmapifp.h`, de sorte à visualiser les définitions de fonction de l'API.

Remarque : **`DSMLINKAGE`** est défini différemment pour chaque système d'exploitation. Pour votre système d'exploitation particulier, voir les définitions contenues dans le fichier `dsmapips.h`.

Les informations fournies ici contiennent une copie ponctuelle des fichiers distribués avec l'API. Affichez les fichiers dans l'API Distribution Package pour la version la plus récente.

```
/*************************************************************************
* Tivoli Storage Manager                                *
* API Client Component                                  *
*                                                                  *
* (C) Copyright IBM Corporation 1993,2002               *
*************************************************************************/

/*************************************************************************/
/* Header File Name: dsmapifp.h                                          */
/*                                                                       */
/* Descriptive-name: Tivoli Storage Manager API function prototypes      */
/*************************************************************************/
#ifndef _H_DSMAPIFP
#define _H_DSMAPIFP

#if defined(__cplusplus)
extern "C" {
#endif

#ifdef DYNALOAD_DSMAPI

/* function will be dynamically loaded */
#include "dsmapidl.h"

#else

/* functions will be implicitly loaded from library */

/*************************************************************************/
/*              P U B L I C   F U N C T I O N S                          */
/*************************************************************************/


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

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

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

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

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

extern dsInt16_t DSMLINKAGE dsmCleanUp(
       dsBool_t               mtFlag
);


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

extern dsInt16_t DSMLINKAGE  dsmDeleteObj(
       dsUint32_t             dsmHandle,
       dsmDelType             delType,
       dsmDelInfo             delInfo
);

extern dsInt16_t DSMLINKAGE  dsmDeleteFS(
       dsUint32_t             dsmHandle,
       char                   *fsName,
       dsUint8_t              repository
);

extern dsInt16_t DSMLINKAGE  dsmEndGetData(
       dsUint32_t             dsmHandle
);

extern dsInt16_t DSMLINKAGE  dsmEndGetDataEx(
       dsmEndGetDataExIn_t  *dsmEndGetDataExInP,
       dsmEndGetDataExOut_t *dsmEndGetDataExOutP
);

extern dsInt16_t DSMLINKAGE  dsmEndGetObj(
       dsUint32_t             dsmHandle
);

extern dsInt16_t DSMLINKAGE  dsmEndQuery(
       dsUint32_t             dsmHandle
);

extern dsInt16_t DSMLINKAGE  dsmEndSendObj(
       dsUint32_t             dsmHandle
);

extern dsInt16_t DSMLINKAGE  dsmEndSendObjEx(
       dsmEndSendObjExIn_t   *dsmEndSendObjExInP,
       dsmEndSendObjExOut_t  *dsmEndSendObjExOutP
);

extern dsInt16_t DSMLINKAGE  dsmEndTxnEx(
       dsmEndTxnExIn_t      *dsmEndTxnExInP,
       dsmEndTxnExOut_t     *dsmEndTxnExOutP
);
```

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

extern dsInt16_t DSMLINKAGE  dsmGetData(
       dsUint32_t             dsmHandle,
       DataBlk                *dataBlkPtr
);

extern dsInt16_t DSMLINKAGE  dsmGetBufferData(
       getBufferDataIn_t     *dsmGetBufferDataInP,
       getBufferDataOut_t    *dsmGetBufferDataOutP
);

extern dsInt16_t DSMLINKAGE  dsmGetNextQObj(
       dsUint32_t             dsmHandle,
       DataBlk                *dataBlkPtr
) ;

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

extern dsInt16_t DSMLINKAGE  dsmGroupHandler(
       dsmGroupHandlerIn_t   *dsmGroupHandlerInP,
       dsmGroupHandlerOut_t  *dsmGroupHandlerOutP
);

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

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


extern dsInt16_t DSMLINKAGE  dsmLogEvent(
       dsUint32_t             dsmHandle,
       logInfo               *lopInfoP
);

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

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

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

extern void DSMLINKAGE       dsmQueryApiVersionEx(
       dsmApiVersionEx       *apiVersionP
);

extern dsInt16_t DSMLINKAGE  dsmQueryCliOptions(
       optStruct             *optstructP
);

extern dsInt16_t DSMLINKAGE  dsmQuerySessInfo(
       dsUint32_t             dsmHandle,
       ApiSessInfo           *SessInfoP
);

extern dsInt16_t DSMLINKAGE  dsmQuerySessOptions(
       dsUint32_t             dsmHandle,
       optStruct             *optstructP
);

extern dsInt16_t DSMLINKAGE  dsmRCMsg(
       dsUint32_t             dsmHandle,
       dsInt16_t             dsmRC,
       char                  *msg
);

extern dsInt16_t DSMLINKAGE  dsmRegisterFS(
       dsUint32_t             dsmHandle,
       regFSData             *regFilespaceP
);

extern dsInt16_t DSMLINKAGE   dsmReleaseBuffer(
       releaseBufferIn_t      *dsmReleaseBufferInP,
       releaseBufferOut_t     *dsmReleaseBufferOutP
);

extern dsInt16_t DSMLINKAGE   dsmRenameObj(
       dsmRenameIn_t          *dsmRenameInP,
       dsmRenameOut_t         *dsmRenameOutP
);

extern dsInt16_t DSMLINKAGE  dsmRequestBuffer(
       requestBufferIn_t     *dsmRequestBufferInP,
       requestBufferOut_t    *dsmRequestBufferOutP
);

extern dsInt16_t DSMLINKAGE   dsmRetentionEvent(
       dsmRetentionEventIn_t  *dsmRetentionEventInP,
       dsmRetentionEventOut_t *dsmRetentionEventOutP
);

extern dsInt16_t DSMLINKAGE   dsmSendBufferData(
       sendBufferDataIn_t     *dsmSendBufferDataInP,
       sendBufferDataOut_t    *dsmSendBufferDataOutP
);

extern dsInt16_t DSMLINKAGE   dsmSendData(
       dsUint32_t             dsmHandle,
       DataBlk                *dataBlkPtr
) ;

extern dsInt16_t DSMLINKAGE   dsmSendObj(
       dsUint32_t             dsmHandle,
       dsmSendType            sendType,
       void                  *sendBuff,
```

```
       dsmObjName             *objNameP,
       ObjAttr                *objAttrPtr,
       DataBlk                *dataBlkPtr
);

extern dsInt16_t DSMLINKAGE   dsmSetAccess(
       dsUint32_t             dsmHandle,
       dsmAccessType          accessType,
       dsmObjName             *objNameP,
       char                   *node,
       char                   *owner
);

extern dsInt16_t DSMLINKAGE   dsmSetUp(
       dsBool_t              mtFlag,
       envSetUp               *envSetUpP
);

extern dsInt16_t DSMLINKAGE   dsmTerminate(
       dsUint32_t             dsmHandle
);

extern dsInt16_t DSMLINKAGE   dsmUpdateFS(
       dsUint32_t             dsmHandle,
       char                   *fs,
       dsmFSUpd               *fsUpdP,
       dsUint32_t            fsUpdAct
);

extern dsInt16_t DSMLINKAGE   dsmUpdateObj(
       dsUint32_t             dsmHandle,
       dsmSendType            sendType,
       void                  *sendBuff,
       dsmObjName             *objNameP,
       ObjAttr                *objAttrPtr,
       dsUint32_t            objUpdAct

);

extern dsInt16_t DSMLINKAGE   dsmUpdateObjEx(
       dsmUpdateObjExIn_t     *dsmUpdateObjExInP,
       dsmUpdateObjExOut_t    *dsmUpdateObjExOutP
);


#endif /* ifdef DYNALOAD */

#if defined(__cplusplus)
   }
#endif

#endif /* _H_DSMAPIFP */
```

Cette section contient les définitions de fonction pour l'API. C'est une copie du fichier d'en-tête `tsmapifp.h`.

Remarque : **DSMLINKAGE** est défini différemment pour chaque système d'exploitation. Pour votre système d'exploitation particulier, voir les définitions contenues dans le fichier `tsmapips.h`.

```
/*************************************************************************
* Tivoli Storage Manager                                 *
* API Client Component                                   *
*                                                        *
* (C) Copyright IBM Corporation 1993,2002                *
*************************************************************************/
```

```
/*****************************************************************************/
/* Header File Name: tsmapifp.h                                              */
/*                                                                           */
/* Descriptive-name: Tivoli Storage Manager API function prototypes       */
/*****************************************************************************/
#ifndef _H_TSMAPIFP
#define _H_TSMAPIFP

#if defined(__cplusplus)
extern "C" {
#endif

#ifdef DYNALOAD_DSMAPI

/* function will be dynamically loaded */
#include "dsmapidl.h"

#else

/* functions will be implicitly loaded from library */

/*===========================================================================*/
/*P U B L I C   F U N C T I O N S                                            */
/*===========================================================================*/


typedef void tsmQueryBuff;

extern dsInt16_t DSMLINKAGE  tsmBeginGetData(
       dsUint32_t            tsmHandle,
       dsBool_t              mountWait,
       tsmGetType            getType,
       dsmGetList            *dsmGetObjListP
);

extern dsInt16_t DSMLINKAGE  tsmBeginQuery(
       dsUint32_t            tsmHandle,
       tsmQueryType          queryType,
       tsmQueryBuff          *queryBuffer
);

extern dsInt16_t DSMLINKAGE  tsmBeginTxn(
       dsUint32_t            tsmHandle
);

extern dsInt16_t DSMLINKAGE  tsmBindMC(
       dsUint32_t            tsmHandle,
       tsmObjName            *objNameP,
       tsmSendType           sendType,
       tsmMcBindKey          *mcBindKeyP
);

extern dsInt16_t DSMLINKAGE  tsmChangePW(
       dsUint32_t            tsmHandle,
       dsChar_t              *oldPW,
       dsChar_t              *newPW
);

extern dsInt16_t DSMLINKAGE tsmCleanUp(
       dsBool_t              mtFlag
);


extern dsInt16_t DSMLINKAGE  tsmDeleteAccess(
       dsUint32_t            tsmHandle,
       dsUint32_t            ruleNum
```

```
);

extern dsInt16_t DSMLINKAGE  tsmDeleteObj(
       dsUint32_t            tsmHandle,
       tsmDelType            delType,
       tsmDelInfo            delInfo
);

extern dsInt16_t DSMLINKAGE  tsmDeleteFS(
       dsUint32_t            tsmHandle,
       dsChar_t              *fsName,
       dsUint8_t             repository
);

extern dsInt16_t DSMLINKAGE  tsmEndGetData(
       dsUint32_t            tsmHandle
);

extern dsInt16_t DSMLINKAGE  tsmEndGetDataEx(
       tsmEndGetDataExIn_t      *tsmEndGetDataExInP,
       tsmEndGetDataExOut_t     *tsmEndGetDataExOutP
);

extern dsInt16_t DSMLINKAGE  tsmEndGetObj(
       dsUint32_t            tsmHandle
);

extern dsInt16_t DSMLINKAGE  tsmEndQuery(
       dsUint32_t            tsmHandle
);

extern dsInt16_t DSMLINKAGE  tsmEndSendObj(
       dsUint32_t            tsmHandle
);

extern dsInt16_t DSMLINKAGE  tsmEndSendObjEx(
       tsmEndSendObjExIn_t      *tsmEndSendObjExInP,
       tsmEndSendObjExOut_t     *tsmEndSendObjExOutP
);

extern dsInt16_t DSMLINKAGE  tsmEndTxn(
       dsUint32_t            tsmHandle,
       dsUint8_t             vote,
       dsUint16_t            *reason
);

extern dsInt16_t DSMLINKAGE  tsmEndTxnEx(
       tsmEndTxnExIn_t      *tsmEndTxnExInP,
       tsmEndTxnExOut_t     *tsmEndTxnExOutP
);

extern dsInt16_t DSMLINKAGE  tsmGetData(
       dsUint32_t            tsmHandle,
       DataBlk*dataBlkPtr
);

extern dsInt16_t DSMLINKAGE  tsmGetBufferData(
       getBufferDataIn_t     *tsmGetBufferDataInP,
       getBufferDataOut_t    *tsmGetBufferDataOutP
);

extern dsInt16_t DSMLINKAGE  tsmGetNextQObj(
       dsUint32_t            tsmHandle,
       DataBlk*dataBlkPtr
) ;

extern dsInt16_t DSMLINKAGE  tsmGetObj(
```

```
       dsUint32_t            tsmHandle,
       ObjID                 *objIdP,
       DataBlk                *dataBlkPtr
);

extern dsInt16_t DSMLINKAGE  tsmGroupHandler(
       tsmGroupHandlerIn_t   *tsmGroupHandlerInP,
       tsmGroupHandlerOut_t  *tsmGroupHandlerOutP
);

extern dsInt16_t DSMLINKAGE  tsmInitEx(
       dsUint32_t            *tsmHandleP,
       tsmInitExIn_t         *tsmInitExInP,
       tsmInitExOut_t        *tsmInitExOutP
);


extern dsInt16_t DSMLINKAGE  tsmLogEventEx(
       dsUint32_t            tsmHandle,
       tsmLogExIn_t          *tsmLogExInP,
       tsmLogExOut_t         *tsmLogExOutP
);

extern dsInt16_t DSMLINKAGE  tsmQueryAccess(
       dsUint32_t            tsmHandle,
       tsmQryRespAccessData  **accessListP,
       dsUint16_t            *numberOfRules
);


extern void DSMLINKAGE       tsmQueryApiVersionEx(
       tsmApiVersionEx       *apiVersionP
);

extern dsInt16_t DSMLINKAGE  tsmQueryCliOptions(
       tsmOptStruct          *optstructP
);

extern dsInt16_t DSMLINKAGE  tsmQuerySessInfo(
       dsUint32_t            tsmHandle,
       tsmApiSessInfo        *SessInfoP
);

extern dsInt16_t DSMLINKAGE  tsmQuerySessOptions(
       dsUint32_t            tsmHandle,
       tsmOptStruct          *optstructP
);

extern dsInt16_t DSMLINKAGE  tsmRCMsg(
       dsUint32_t            tsmHandle,
       dsInt16_t             tsmRC,
       dsChar_t              *msg
);

extern dsInt16_t DSMLINKAGE  tsmRegisterFS(
       dsUint32_t            tsmHandle,
       tsmRegFSData          *regFilespaceP
);

extern dsInt16_t DSMLINKAGE  tsmReleaseBuffer(
       releaseBufferIn_t     *tsmReleaseBufferInP,
       releaseBufferOut_t    *tsmReleaseBufferOutP
);

extern dsInt16_t DSMLINKAGE  tsmRenameObj(
       tsmRenameIn_t         *tsmRenameInP,
       tsmRenameOut_t        *tsmRenameOutP
```

```
);

extern dsInt16_t DSMLINKAGE  tsmRequestBuffer(
       requestBufferIn_t     *tsmRequestBufferInP,
       requestBufferOut_t    *tsmRequestBufferOutP
);

extern dsInt16_t DSMLINKAGE   tsmRetentionEvent(
       tsmRetentionEventIn_t  *tsmRetentionEventInP,
       tsmRetentionEventOut_t *tsmRetentionEventOutP
);

extern dsInt16_t DSMLINKAGE   tsmSendBufferData(
       sendBufferDataIn_t     *tsmSendBufferDataInP,
       sendBufferDataOut_t    *tsmSendBufferDataOutP
);

extern dsInt16_t DSMLINKAGE  tsmSendData(
       dsUint32_t             tsmHandle,
       DataBlk                 *dataBlkPtr
);

extern dsInt16_t DSMLINKAGE  tsmSendObj(
       dsUint32_t             tsmHandle,
       tsmSendType            sendType,
       void                   *sendBuff,
       tsmObjName             *objNameP,
       tsmObjAttr             *objAttrPtr,
       DataBlk                 *dataBlkPtr
);

extern dsInt16_t DSMLINKAGE  tsmSetAccess(
       dsUint32_t             tsmHandle,
       tsmAccessType          accessType,
       tsmObjName             *objNameP,
       dsChar_t               *node,
       dsChar_t               *owner
);

extern dsInt16_t DSMLINKAGE tsmSetUp(
       dsBool_t              mtFlag,
       tsmEnvSetUp           *envSetUpP
);

extern dsInt16_t DSMLINKAGE  tsmTerminate(
       dsUint32_t             tsmHandle
);

extern dsInt16_t DSMLINKAGE  tsmUpdateFS(
       dsUint32_t             tsmHandle,
       dsChar_t               *fs,
       tsmFSUpd               *fsUpdP,
       dsUint32_t             fsUpdAct
);

extern dsInt16_t DSMLINKAGE  tsmUpdateObj(
       dsUint32_t             tsmHandle,
       tsmSendType            sendType,
       void                   *sendBuff,
       tsmObjName             *objNameP,
       tsmObjAttr             *objAttrPtr,
       dsUint32_t             objUpdAct

);

extern dsInt16_t DSMLINKAGE tsmUpdateObjEx(
       tsmUpdateObjExIn_t         *tsmUpdateObjExInP,
```

```
       tsmUpdateObjExOut_t        *tsmUpdateObjExOutP
);


#endif /* ifdef DYNALOAD */

#if defined(__cplusplus)
   }
#endif

#endif /* _H_TSMAPIFP */
```

Annexe D. Fonctions d'accessibilité de la famille de produits IBM Spectrum Protect

Les fonctions d'accessibilité permettent aux utilisateurs souffrant d'un handicap physique, comme une mobilité réduite ou une déficience visuelle, d'utiliser correctement les contenus informatiques.

Présentation

La famille de produits IBM Spectrum Protect comprend les fonctions d'accessibilité majeures suivantes :

- Fonctionnement à l'aide d'un clavier uniquement
- Opérations qui utilisent un lecteur d'écran

La famille IBM Spectrum Protect de produits utilise la toute dernière norme W3C, WAI-ARIA 1.0 (www.w3.org/TR/wai-aria/), pour assurer la conformité à US Section 508 (www.access-board.gov/guidelines-and-standards/communications-and-it/about-the-section-508-standards/section-508-standards) et Web Content Accessibility Guidelines (WCAG) 2.0 (www.w3.org/TR/WCAG20/). Pour bénéficier des fonctionnalités d'accessibilité, utilisez la dernière édition de votre lecteur d'écran et le dernier navigateur Web pris en charge par le produit.

La documentation du produit dans IBM Knowledge Center est activée pour l'accessibilité. Les fonctions d'accessibilité du centre IBM Knowledge Center sont décrites dans la section Accessibilité de l'aide d'IBM Knowledge Center (www.ibm.com/support/knowledgecenter/about/releasenotes.html?view=kc#accessibility).

Navigation par le clavier

Ce produit utilise les touches de navigation standard.

Informations sur l'interface

Les interfaces utilisateur ne disposent pas de contenu qui clignote 2 à 55 fois par seconde.

Les interfaces utilisateur Web s'appuient sur des feuilles de style en cascade pour afficher le contenu correctement et pour offrir une expérience exploitable. L'application offre aux utilisateurs dont la vision est altérée un moyen équivalent d'utiliser les paramètres d'affichage système, dont le mode de contraste élevé. Vous pouvez contrôler la taille de la police en utilisant les paramètres de l'unité ou du navigateur Web.

Les interfaces utilisateur Web incluent des repères de navigation WAI-ARIA que vous pouvez utiliser pour naviguer rapidement vers les zones fonctionnelles de l'application.

Logiciels fournisseur

La famille de produits IBM Spectrum Protect inclut certains logiciels fournisseur non protégés par le contrat de licence IBM. IBM décline toute responsabilité

concernant les fonctions d'accessibilité de ces produits. Contactez le fournisseur pour obtenir les informations d'accessibilité relatives à ses produits.

Informations relatives à l'accessibilité

Outre les sites Web du support et du centre d'assistance IBM, IBM propose un service de téléphone par téléscripteur à l'usage des clients sourds ou malentendants leur permettant d'accéder aux services des ventes et du support :

Service de
téléscripteur
800-IBM-3383 (800-426-3383)
(Amérique du Nord)

Pour plus d'informations sur l'engagement d'IBM en matière d'accessibilité, visitez le site IBM Accessibility(www.ibm.com/able).

Remarques

Le présent document peut contenir des informations ou des références concernant certains produits, logiciels ou services IBM non annoncés dans ce pays. Cette documentation peut être proposée par IBM dans d'autres langues. Vous pouvez toutefois devoir détenir une copie du produit ou une version du produit dans cette langue pour pouvoir y accéder.

Le présent document peut contenir des informations ou des références concernant certains produits, logiciels ou services IBM non annoncés dans ce pays. Pour plus de détails, référez-vous aux documents d'annonce disponibles dans votre pays, ou adressez-vous à votre partenaire commercial IBM. Toute référence à un produit, logiciel ou service IBM n'implique pas que seul ce produit, logiciel ou service IBM puisse être utilisé. Tout autre élément fonctionnellement équivalent peut être utilisé, s'il n'enfreint aucun droit d'IBM. Il est de la responsabilité de l'utilisateur d'évaluer et de vérifier lui-même les installations et applications réalisées avec des produits, logiciels ou services non expressément référencés par IBM.

IBM peut détenir des brevets ou des demandes de brevet couvrant les produits mentionnés dans le présent document. La remise de ce document ne vous donne aucun droit de licence sur ces brevets. Si vous désirez recevoir des informations concernant l'acquisition de licences, veuillez en faire la demande par écrit à l'adresse suivante :

IBM EMEA Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
U.S.A.

Pour le Canada, veuillez adresser votre courrier à :

IBM Director of Commercial Relations
IBM Canada Ltd.
3600 Steeles Avenue East
Markham, Ontario
L3R 9Z7 Canada

Les informations sur les licences concernant les produits utilisant un jeu de caractères double octet peuvent être obtenues par écrit à l'adresse IBM suivante :

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

LE PRESENT DOCUMENT EST LIVRE "EN L'ETAT" SANS AUCUNE GARANTIE EXPLICITE OU IMPLICITE. IBM DECLINE NOTAMMENT TOUTE RESPONSABILITE RELATIVE A CES INFORMATIONS EN CAS DE CONTREFACON AINSI QU'EN CAS DE DEFAUT D'APTITUDE A L'EXECUTION

D'UN TRAVAIL DONNE. Certaines juridictions n'autorisent pas l'exclusion des garanties explicites ou implicites dans certaines transactions, auquel cas l'exclusion ci-dessus ne vous sera pas applicable.

Le présent document peut contenir des inexactitudes ou des coquilles. Les informations fournies dans ce document sont régulièrement modifiées, ces modifications seront intégrées aux prochaines éditions de la publication. IBM peut, à tout moment et sans préavis, modifier les produits et logiciels décrits dans ce document.

Les références à des sites Web non IBM sont fournies à titre d'information uniquement et n'impliquent en aucun cas une adhésion aux données qu'ils contiennent. Les éléments figurant sur ces sites ne font pas partie des éléments du produit IBM et l'utilisation de ces sites relève de votre seule responsabilité.

IBM pourra utiliser ou diffuser, de toute manière qu'elle jugera appropriée et sans aucune obligation de sa part, tout ou partie des informations qui lui seront fournies.

Les licenciés souhaitant obtenir des informations permettant : (i) l'échange des données entre des logiciels créés de façon indépendante et d'autres logiciels (dont celui-ci), et (ii) l'utilisation mutuelle des données ainsi échangées, doivent adresser leur demande à :

IBM EMEA Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
U.S.A.

Ces informations peuvent être soumises à des conditions particulières, prévoyant notamment le paiement d'une redevance.

Le logiciel sous licence décrit dans ce document et tous les éléments sous licence disponibles s'y rapportant sont fournis par IBM conformément aux dispositions de l'ICA (IBM Customer Agreement), des Conditions internationales d'utilisation des logiciels IBM ou de tout autre accord équivalent.

Les données de performance présentées ici ont été obtenues dans des conditions de fonctionnement spécifiques. Les résultats peuvent donc varier.

Les informations concernant des produits non IBM ont été obtenues auprès des fournisseurs de ces produits, par l'intermédiaire d'annonces publiques ou via d'autres sources disponibles. IBM n'a pas testé ces produits et ne peut confirmer l'exactitude de leurs performances ni leur compatibilité. Elle ne peut recevoir aucune réclamation concernant des produits non IBM. Toute question concernant les performances de produits non IBM doit être adressée aux fournisseurs de ces produits.

Le présent document peut contenir des exemples de données et de rapports utilisés couramment dans l'environnement professionnel. Ces exemples mentionnent des noms fictifs de personnes, de sociétés, de marques ou de produits à des fins illustratives ou explicatives uniquement. Toute ressemblance avec des noms de personnes, de sociétés ou des données réelles serait purement fortuite.

LICENCE DE COPYRIGHT :

Le présent logiciel contient des exemples de programmes d'application en langage source destinés à illustrer les techniques de programmation sur différentes plateformes d'exploitation. Vous avez le droit de copier, de modifier et de distribuer ces exemples de programmes sous quelque forme que ce soit et sans paiement d'aucune redevance à IBM, à des fins de développement, d'utilisation, de vente ou de distribution de programmes d'application conformes aux interfaces de programmation des plateformes pour lesquels ils ont été écrits ou aux interfaces de programmation IBM. Ces exemples de programmes n'ont pas été rigoureusement testés dans toutes les conditions. Par conséquent, IBM ne peut garantir expressément ou implicitement la fiabilité, la maintenabilité ou le fonctionnement de ces programmes. Les programmes exemples sont fournis "EN L'ETAT", sans garantie d'aucune sorte. IBM ne sera en aucun cas responsable des dommages liés à l'utilisation des programmes exemples.

Toute copie totale ou partielle de ces programmes exemples et des oeuvres qui en sont dérivées doit comprendre une notice de copyright, libellée comme suit: © (nom de votre société) (année). Des parties de ce code proviennent de IBM Corp. Sample Programs. © Copyright IBM Corp. _saisissez l'année ou les années_.

Marques

IBM, le logo IBM et ibm.com sont des marques d'International Business Machines Corporation aux Etats-Unis et/ou dans certains autres pays. Les autres noms de produits et de services peuvent être des marques d'IBM ou d'autres sociétés. La liste actualisée de toutes les marques d'IBM est disponible sur la page Web "Copyright and trademark information" à www.ibm.com/legal/copytrade.shtml.

Adobe est une marque d'Adobe Systems Incorporated aux Etats-Unis et/ou dans certains autres pays.

Linear Tape-Open, LTO et Ultrium sont des marques de HP, IBM Corp. et Quantum, aux Etats-Unis et/ou dans certains autres pays.

Intel et Itanium sont des marques d'Intel Corporation ou de ses filiales aux Etats-Unis et dans certains autres pays.

Linux est une marque de Linus Torvalds aux Etats-Unis et/ou dans certains autres pays.

Microsoft, Windows et Windows NT sont des marques de Microsoft Corporation aux Etats-Unis et/ou dans certains autres pays.

Java™ ainsi que tous les logos et toutes les marques incluant Java sont des marques d'Oracle et/ou de ses sociétés affiliées.

SoftLayer est une marque de SoftLayer, Inc., une société du groupe IBM.

UNIX est une marque enregistrée de The Open Group aux Etats-Unis et dans certains autres pays.

Dispositions relatives à la documentation du produit

Les droits d'utilisation relatifs à ces publications sont soumis aux dispositions suivantes.

Applicabilité
Ces dispositions s'ajoutent aux conditions d'utilisation relatives au site Web IBM.

Usage personnel
Vous pouvez reproduire ces publications pour votre usage personnel, non commercial, sous réserve que toutes les mentions de propriété soient conservées. Vous ne pouvez pas distribuer ni afficher tout ou partie de ces publications ou en faire des oeuvres dérivées sans le consentement exprès d'IBM.

Usage commercial
Vous pouvez reproduire, distribuer et publier ces publications uniquement au sein de votre entreprise, sous réserve que toutes les mentions de propriété soient conservées. Vous ne pouvez reproduire, distribuer, afficher ou publier tout ou partie de ces publications en dehors de votre entreprise, ou en faire des oeuvres dérivées, sans le consentement exprès d'IBM.

Droits Excepté les droits d'utilisation expressément accordés dans ce document, aucun autre droit, licence ou autorisation, implicite ou explicite, n'est accordé pour ces publications ou autres informations, données, logiciels ou droits de propriété intellectuelle contenus dans ces publications.

IBM se réserve le droit de retirer les autorisations accordées ici si, à sa discrétion, l'utilisation des informations s'avère préjudiciable à ses intérêts ou que, selon son appréciation, les instructions susmentionnées n'ont pas été respectées.

Vous ne pouvez télécharger, exporter ou réexporter ces informations qu'en total accord avec toutes les lois et règlements applicables dans votre pays, y compris les lois et règlements américains relatifs à l'exportation.

IBM N'OCTROIE AUCUNE GARANTIE SUR LE CONTENU DE CES PUBLICATIONS. LES PUBLICATIONS SONT LIVREES EN L'ETAT SANS AUCUNE GARANTIE EXPLICITE OU IMPLICITE. IBM DECLINE NOTAMMENT TOUTE RESPONSABILITE RELATIVE A CES PUBLICATIONS EN CAS DE CONTREFAÇON AINSI QU'EN CAS DE DEFAUT D'APTITUDE A L'EXECUTION D'UN TRAVAIL DONNE.

Politique de confidentialité

Les Logiciels IBM, y compris les Logiciels sous forme de services ("Offres Logiciels"), peuvent utiliser des cookies ou d'autres technologies pour collecter des informations sur l'utilisation des produits, améliorer l'acquis utilisateur, personnaliser les interactions avec celui-ci, ou dans d'autres buts. Bien souvent, aucune information personnelle identifiable n'est collectée par les Offres Logiciels. Certaines Offres Logiciels vous permettent cependant de le faire. Si la présente Offre Logiciels utilise des cookies pour collecter des informations personnelles identifiables, des informations spécifiques sur cette utilisation sont fournies ci-dessous.

La présente Offre Logiciels n'utilise pas de cookies ni aucune autre technologie pour collecter des informations personnelles identifiables.

Si les configurations déployées de cette Offre Logiciels vous permettent, en tant que client, de collecter des informations permettant d'identifier les utilisateurs par l'intermédiaire de cookies ou par d'autres techniques, vous devez solliciter un avis juridique sur la réglementation applicable à ce type de collecte, notamment en termes d'information et de consentement.

Pour plus d'informations sur l'utilisation à ces fins des différentes technologies, y compris celle des cookies, consultez les Points principaux de la déclaration IBM de confidentialité sur Internet à l'adresse http://http://www.ibm.com/privacy/fr/fr/, la section “Cookies, pixels espions et autres technologies” de la Déclaration IBM de confidentialité sur Internet à l'adresse http://http://www.ibm.com/privacy/details/fr/fr et la section “IBM Software Products and Software-as-a-Service Privacy Statement” à l'adresse http://www.ibm.com/software/info/product-privacy.

Glossaire

Un glossaire réunissant les termes et définitions qui se rapportent à la famille de produits IBM Spectrum Protect est disponible.

Voir le Glossaire IBM Spectrum Protect.

Pour consulter les glossaires d'autres produits IBM, voir IBM Terminology.

Index

Nombres

A

B

C

D

E

F

G

H

I

J

K

L

M

T

U

V

W

Numéro de programme : 5725-W98
5725-W99
5725-X15