4D Backup

Langage
Windows® / Mac™ OS

4D Backup - Langage
Versions Windows® et Mac™ OS

Sommaire

1

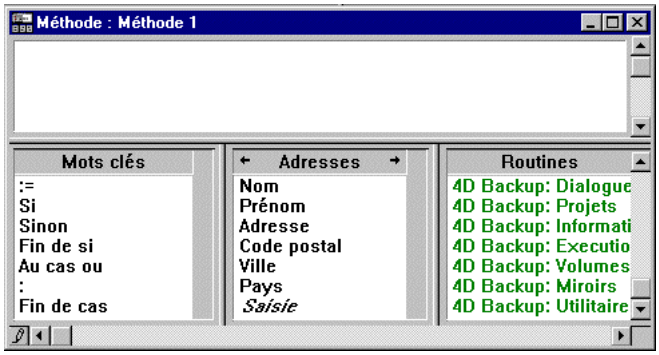
Introduction

Une fois installé dans votre environnement 4D, le plug-in de sauvegarde de 4D Backup vous fait bénéficier d'une quarantaine de commandes spécifiques qui viennent s'ajouter au langage de 4e Dimension. Ces commandes vous permettent de piloter le plug-in de sauvegarde de façon particulièrement fine, afin de mettre en œuvre des stratégies de sauvegarde automatisées, voire de créer une méthode de sauvegarde spécifique à votre application.

A propos de ce manuel

• Contenu

Ce manuel est le manuel *Langage* de 4D Backup. Destiné aux programmeurs, il décrit la syntaxe et le fonctionnement des commandes et des fonctions du langage de 4D Backup. L'interface, les menus et l'utilisation générale de 4D Backup font l'objet d'une documentation séparée, le manuel *Utilisation* de 4D Backup. En outre, deux chapitres annexes fournissent la liste des codes d'erreurs retournés par 4D Backup, ainsi que les procédures de redémarrage rapide en cas d'incident.
Les routines du langage de 4D Backup sont regroupées par thèmes, dans cette documentation tout comme dans la fenêtre d'édition des méthodes de 4D :



Afin de vous permettre d'exploiter au mieux le langage de 4D Backup, les descriptions des routines sont illustrées par des exemples décrivant l'implémentation de stratégies de sauvegarde simples.

• Conventions

Les conventions d'écriture et symboles utilisés sont ceux des manuels de 4D. En cas d'incertitude, reportez-vous au manuel *Langage* de 4D.
Les commandes 4D Backup (routines ne retournant pas de valeur) sont inscrites en majuscules et que les fonctions (routines retournant une valeur) n'ont que leur initiale en majuscule. Dans tous les cas, les commandes et fonctions du langage de 4D Backup sont écrites en caractères spéciaux et précédées du préfixe bk_, afin de les différencier de celles de 4D et des autres plug-ins.

• **MacOS/Windows**
Ce manuel s'adresse indifféremment aux utilisateurs des versions Windows et MacOS (Macintosh ou Power Macintosh) de 4D Backup. Les explications s'appliquent aux deux plates-formes. Toute différence de fonctionnement entre les versions MacOS et Windows de 4D Backup est toutefois signalée au cours du texte.

Nous attirons votre attention sur la structure particulière des commandes de 4D Backup, qui pilotent en fait une tâche de fond, un process, interne à 4D. Cette tâche de fond doit être explicitement activée pour exécuter les instructions de 4D Backup. En d'autres termes, la plupart des routines de 4D Backup ne fonctionnent que si le process de 4D Backup a été lancé puis refermé.

Ouverture du process de sauvegarde

Pour lancer le process de sauvegarde, vous disposez de deux routines :

- bk_Debut sauvegarde integrale pour la sauvegarde intégrale,
- bk_Debut mise a jour miroir pour la mise à jour d'un miroir logique.

Ces fonctions ne déclenchent pas la sauvegarde de la base ni la mise à jour du poste miroir, elles ouvrent le process de 4D Backup permettant de réaliser ces opérations, ainsi que les autres fonctionnalités offertes par le langage de 4D Backup.

Note : La sauvegarde étant donc elle-même un process interne à 4e Dimension, l'envoi des données s'effectuant également dans un process, nous vous conseillons de lire les chapitres de la documentation de 4e Dimension consacrés au multiprocess avant d'entreprendre la lecture de ce manuel.

Déclenchement de la sauvegarde

Une fois le process de sauvegarde ouvert par l'une des deux fonctions décrites ci-dessus, toute opération de sauvegarde (sauvegarde intégrale ou mise à jour d'un miroir par envoi du fichier d'historique) sera déclenchée par la fonction bk_Lancer copie. Cette fonction se trouve dans le chapitre "BK Exécution".

Après avoir lancé la copie, cette fonction vous rendra immédiatement la main, et il vous appartiendra d'attendre que la copie soit terminée dans son process, en l'interrogeant périodiquement à l'aide de la fonction bk_Lire etat. Cette fonction renvoie en effet la valeur 4 lorsque la copie est en cours. Typiquement, vos méthodes de sauvegarde et de mise à jour d'un miroir comporteront donc les instructions suivantes :

```
...
$vBackup:=bk_Lancer copie
Repeter
Jusque (bk_Lire etat#4)
...
```

Fermeture du process de sauvegarde

Le process de sauvegarde exerce un certain nombre de contrôles sur la base, comme par exemple le verrouillage en écriture lors d'une sauvegarde intégrale. Il doit donc être fermé lorsque les opérations sont terminées afin, en particulier, de libérer la base en écriture.

Pour cela, utilisez la commande bk_FIN SAUVEGARDE, qui valide puis ferme le process de sauvegarde.

Note aux utilisateurs de 4e Dimension monoposte : Si vous omettez la commande bk_FIN SAUVEGARDE à la fin d'une méthode de sauvegarde, la base restera bloquée. Dans ce cas, maintenez pendant quelques secondes la combinaison de touches **Alt**+**Majuscule**+**Clic bouton droit** (sous Windows) ou **Contrôle**+**Option**+**Majuscule**+**Clic** (sous MacOS) qui vous permet, en cas de nécessité, de "tuer" manuellement le process de sauvegarde.

Exceptions

Quatre routines de 4D Backup ne nécessitent ni ouverture ni fermeture explicites du process de sauvegarde (elles gèrent elles-mêmes l'ouverture et la fermeture de ce process) :

• bk_FENETRE MISE A JOUR MIROIR et bk_FENETRE SAUVEGARDE INTEGRALE (chapitre "BK Dialogues standard"). Ces commandes font apparaître les boîtes de dialogue de paramétrage de sauvegarde intégrale et de mise à jour du miroir, de la même manière que les commandes du menu **Plug-Ins**.

• bk_Mise a jour miroir (chapitre "BK Miroir logique") et bk_Sauvegarde integrale (chapitre "BK Exécution"). Ces fonctions déclenchent immédiatement les opérations correspondantes, à partir d'un projet par défaut se trouvant dans le répertoire de la base.

2

BK Dialogues standard

Ce thème regroupe les routines permettant de faire apparaître les boîtes de dialogue standard de 4D Backup, comme lorsqu'ils sont appelés par le menu **Plug-Ins** en mode Utilisation.
Leur objectif est de vous permettre de simuler ces commandes en mode Menus créés, de façon à laisser aux utilisateurs de votre base un accès standard au module de sauvegarde.

bk_FENETRE MISE A JOUR MIROIR

Paramètre	Type	Description

Cette commande ne requiert pas de paramètre

Description

La commande bk_FENETRE MISE A JOUR MIROIR fait simplement apparaître la fenêtre de mise à jour du miroir.

bk_FENETRE MISE A JOUR MIROIR permet de présenter aux utilisateurs d'une base en mode Menus créés l'accès à la fenêtre de paramétrage de la mise à jour du miroir, de la même façon que par la commande **Mise à jour du miroir...** du menu **Plug-Ins**.

Cette commande est principalement destinée à permettre à l'administrateur d'une base compilée — donc sans accès au mode Structure — d'accéder à la fenêtre de paramétrage de la mise à jour du miroir afin, par exemple, de modifier un projet de sauvegarde par défaut. Pour mettre à jour périodiquement le miroir, utilisez de préférence la fonction bk_Mise a jour miroir. En revanche, il peut être intéressant d'appeler la commande bk_FENETRE MISE A JOUR MIROIR lorsque bk_Mise a jour miroir retourne une erreur.

Contexte d'utilisation :
• dans une méthode appelée par une barre de menus,
• dans la méthode objet d'un bouton,
• depuis un process attendant qu'il soit une certaine heure pour proposer la mise à jour du miroir, lorsque vous quittez la base.

Note : Cette commande, comme la commande bk_FENETRE SAUVEGARDE INTEGRALE, n'a pas besoin d'être encadrée par les routines bk_Debut mise a jour miroir (ou bk_Debut sauvegarde integrale) et bk_FIN SAUVEGARDE : elle gère elle-même l'ouverture et la fermeture du process de sauvegarde.

Exemple

Cette méthode présente la boîte de dialogue de paramétrage de la mise à jour du miroir si la fonction bk_Mise a jour miroir a renvoyé une erreur :

```
Si (bk_Mise a jour miroir#0)
   ALERTE ("La mise à jour du miroir a échoué.")
⇒  bk_FENETRE MISE A JOUR MIROIR
Fin de si
```

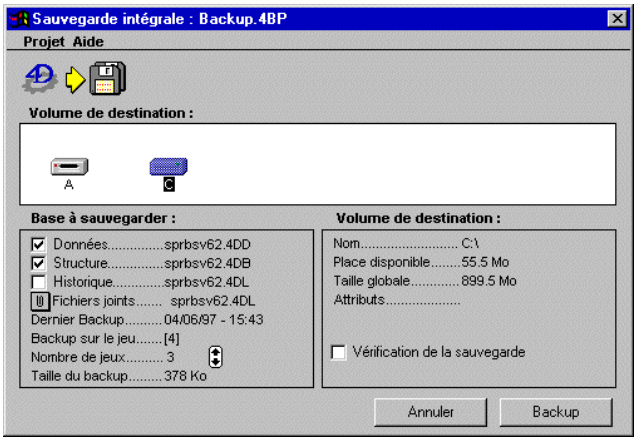
bk_FENETRE SAUVEGARDE INTEGRALE

Paramètre	Type	Description

Cette commande ne requiert pas de paramètre

Description

La commande bk_FENETRE SAUVEGARDE INTEGRALE fait apparaître la fenêtre de sauvegarde intégrale, de la même façon que si vous aviez choisi la commande **Sauvegarde intégrale...** du menu **Plug-Ins** de 4e Dimension ou de 4D Client.



Cette commande vous permet d'offrir aux utilisateurs de votre base en mode Menus créés les mêmes possibilités de sauvegarde qu'en mode Utilisation.

Si vous créez une méthode qui appelle la commande bk_FENETRE SAUVEGARDE INTEGRALE et placez l'appel à cette méthode dans la commande d'un des menus de votre base, la base pourra être sauvegardée depuis le mode Menus créés, même compilée et fusionnée avec 4D Engine.

Contextes d'utilisation :
• dans une méthode appelée par une barre de menus,
• dans la méthode objet d'un bouton,
• depuis un process attendant qu'il soit une certaine heure pour proposer la sauvegarde, lorsque vous quittez la base.

Note : Cette commande, comme la commande bk_FENETRE MISE A JOUR MIROIR, n'a pas besoin d'être encadrée par les routines bk_Debut sauvegarde integrale (ou bk_Debut mise a jour miroir), et bk_FIN SAUVEGARDE : elle gère elle-même l'ouverture et la fermeture du process de sauvegarde.

3

BK Exécution

BK Exécution, Introduction

Ce thème regroupe les routines relatives à l'exécution des sauvegardes et à leur suivi. Vous pourrez ainsi lancer la copie des données vers votre disque de sauvegarde et en suivre la progression. Vous pourrez également envoyer le fichier d'historique à la base miroir pour la mettre à jour.

Note : Ce thème traite des opérations de déclenchement des copies, dont celles du fichier d'historique. Les routines relatives au paramétrage et à la gestion d'un miroir logique sont traitées dans le chapitre "BK Miroir logique".

bk_Debut sauvegarde integrale → Entier

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Entier	←	0 si l'exécution est correcte, sinon Code d'erreur

Description

La commande bk_Debut sauvegarde integrale lance le process de sauvegarde de 4D Backup. Cette fonction réalise des opérations préparatoires nécessaires à la sauvegarde de la base : elle attend que toutes les transactions soient terminées et verrouille la base en écriture.

Toutes les commandes et fonctions de 4D Backup doivent être encadrées par les routines bk_Debut sauvegarde integrale (ou bk_Debut mise a jour miroir) et bk_FIN SAUVEGARDE — quatre routines autonomes font exception à ce mode de fonctionnement (reportez-vous à la section Structure des instructions).

bk_Debut sauvegarde integrale retourne 0 lorsque son exécution est correcte. Dans le cas contraire, un numéro d'erreur est renvoyé (reportez-vous à la liste fournie à l'Annexe A, Codes d'erreurs de 4D Backup). Nous vous recommandons de tester systématiquement la valeur retournée. En effet, si bk_Debut sauvegarde integrale retourne une valeur autre que 0, les instructions suivantes seront toutes ignorées.

Accès à la base durant la sauvegarde

Lors de l'appel à bk_Debut sauvegarde integrale, les accès à la base sont restreints de la manière suivante :

• **Avec 4e Dimension monoposte** : tous les process autres que le process qui a appelé la sauvegarde sont endormis, ils ne seront libérés qu'après l'exécution de la commande bk_FIN SAUVEGARDE. Ainsi, la base ne risque pas d'être modifiée pendant la sauvegarde.

Note : La combinaison de touches **Alt**+**Majuscule**+**Clic bouton droit** (sous Windows) ou **Contrôle**+**Option**+**Majuscule**+**Clic** (sous MacOS) permet, en cas de nécessité, de "tuer" le process de sauvegarde.

• **Avec 4D Server** : Les clients connectés à 4D Server pourront consulter les enregistrements, mais sans pouvoir les modifier ni les effacer. Si un poste client envoie une requête d'ajout, de suppression ou de modification au serveur, il obtient une fenêtre l'invitant à attendre la fin de la sauvegarde. Une fois la base sauvegardée, la fenêtre disparaît d'elle-même et l'action est effectuée.
Le bouton **Annuler l'opération** permet à l'utilisateur d'annuler sa requête en cours pour ne pas avoir à attendre la fin de la sauvegarde.

Cependant, si l'action en attente provient d'une méthode lancée avant la sauvegarde, il est déconseillé de l'annuler car seules les opérations restant à effectuer seront annulées. Or, une méthode "à moitié" exécutée peut conduire à des incohérences logiques dans la base.

Note : Lorsque l'action en attente provient d'une méthode et que l'utilisateur clique sur le bouton **Annuler l'opération**, 4D renvoie l'erreur -9976.

Vous verrez également apparaître dans la fenêtre d'administration du serveur le process de sauvegarde.

Traitement des transactions

bk_Debut sauvegarde integrale attend pour démarrer que toutes les transactions soient terminées. Si dans le même process vous démarrez une transaction par la commande de 4e Dimension DEBUT TRANSACTION et que vous appelez ensuite la commande bk_Debut sauvegarde integrale, sans avoir exécuté ni VALIDER TRANSACTION ni ANNULER TRANSACTION, la fonction bk_Debut sauvegarde integrale retournera l'erreur 1404.
Dans le même ordre d'idée, évitez de présenter à l'utilisateur une boîte de dialogue de saisie au cours d'une transaction. Si un utilisateur part déjeuner en laissant sa machine allumée avec un dialogue ouvert pendant une transaction, vous ne pourrez pas sauvegarder la base. De plus, la base sera verrouillée en écriture pour tous les utilisateurs tant que cette transaction ne sera pas validée.
Il est donc plus astucieux d'effectuer vos transactions après les boîtes de dialogue. Ce n'est que lorsque l'utilisateur valide son opération que la transaction est lancée, puis validée ou annulée. De cette façon, non seulement vous optimisez le fonctionnement général de votre base, mais vous évitez de bloquer la sauvegarde à cause d'une transaction bloquée par un utilisateur.
A noter également que l'ensemble de la base étant verrouillé, si vous souhaitez mettre à jour un enregistrement, par exemple pour stocker des informations sur la sauvegarde, vous devrez le faire après la commande bk_FIN SAUVEGARDE. En effet, il n'est pas possible de modifier, ajouter ou supprimer un enregistrement entre les instructions bk_Debut sauvegarde integrale et bk_FIN SAUVEGARDE.

Exemples

(1) Voici la méthode la plus simple permettant de déclencher une sauvegarde intégrale :

```
C_ENTIER($vErreur)
$vErreur:=bk_Sauvegarde integrale
```

(2) Cette méthode, plus complète, déclenche également la sauvegarde :

```
⇒   Si(bk_Debut sauvegarde integrale=0)  `Ouverture du process de sauvegarde
      Si (bk_Lancer copie =0)  `Déclenchement de la sauvegarde
        Repeter
        Jusque (bk_Lire etat #4)  `Fin de la copie
      Fin de si
      bk_FIN SAUVEGARDE  `Fermeture du process de sauvegarde
    Fin de si
```

(3) Vous pouvez affiner la méthode précédente en gérant les éventuelles erreurs :

```
      C_ENTIER($vErreur;$vEtat)
⇒     $vErreur:=bk_Debut sauvegarde integrale
      Si ($vErreur#0)
         ALERTE("Impossible de démarrer la sauvegarde : erreur N° " +Chaine($vErreur))
      Sinon
         $vErreur:=bk_Lancer copie
         Si ($vErreur#0)
            ALERTE("Impossible de démarrer la copie : erreur N° " + Chaine($vErreur))
         Sinon
            Repeter
               $vEtat:=bk_Lire etat
            Jusque ($vEtat#4)
            Si ($vEtat#5)
               ALERTE("Problème durant la copie : erreur N° " + Chaine($vEtat))
            Fin de si
         Fin de si
         bk_FIN SAUVEGARDE
      Fin de si
```

Référence

bk_Debut mise a jour miroir, bk_FIN SAUVEGARDE, bk_Lire etat.

bk_FIN SAUVEGARDE

Paramètre	Type	Description
Cette commande ne requiert pas de paramètre		

Description

La commande bk_FIN SAUVEGARDE referme le process de 4D Backup. Dans le cas d'une sauvegarde intégrale, l'ensemble de la base est déverrouillé, autorisant à nouveau les ajouts, modifications et suppressions d'enregistrements.

Chaque appel aux fonctions bk_Debut sauvegarde integrale ou bk_Debut mise a jour miroir doit être suivi d'un appel à bk_FIN SAUVEGARDE une fois la sauvegarde terminée.

Note : Les enregistrements déjà verrouillés par des utilisateurs avant l'appel à bk_Debut sauvegarde integrale le restent après que la base ait été libérée. Le verrou installé par 4D Backup pendant la sauvegarde est global à l'ensemble de la base et il est distinct de celui spécifique à chaque enregistrement.

Lors de l'appel à cette commande, trois cas peuvent se présenter :

• La base ou le fichier d'historique a été correctement copié(e).
C'est le cas standard, la valeur retournée par la fonction bk_Lire etat vaut alors 5.

• La base ou le fichier d'historique est en cours de copie.
Dans ce cas, l'appel à bk_FIN SAUVEGARDE équivaut à un clic sur le bouton **Stop** lorsque l'on exécute la sauvegarde depuis la fenêtre de 4D Backup. L'opération en cours est annulée, le fichier de destination est effacé.
Pour la mise à jour du miroir, le fichier d'historique à envoyer et le nouveau fichier d'historique sont concaténés pour revenir à l'état précédent.

• La base ou le fichier d'historique n'a pas (encore) été copié(e).
Pour une sauvegarde intégrale, cette situation ne pose pas de problème particulier, dans la mesure où il suffira de recommencer la méthode de sauvegarde.
Pour une mise à jour du poste miroir, un certain nombre d'opérations ont été réalisées (le fichier d'historique courant est fermé, prêt à être envoyé, et un nouvel historique est créé). Dans ce cas, le fichier d'historique à sauvegarder et le nouveau fichier d'historique sont concaténés pour revenir à l'état précédent.

Note : Si vous utilisez 4e Dimension version monoposte et que vous omettez la commande bk_FIN SAUVEGARDE dans la méthode de sauvegarde, la base restera bloquée. Dans ce cas, utilisez la combinaison de touches **Alt**+**Majuscule**+**Clic bouton droit** (sous Windows) ou **Contrôle**+**Option**+**Majuscule**+**Clic** (sous MacOS) qui permet, en cas d'oubli, de "tuer" le process de sauvegarde.

Référence

bk_Debut mise a jour miroir, bk_Debut sauvegarde integrale, bk_Lire etat.

bk_Lancer copie → Entier

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Entier	←	0 si la copie a été correctement lancée, sinon Code d'erreur

Description

La commande bk_Lancer copie lance la copie proprement dite des données.

La valeur retournée est un code de contrôle précisant si la copie a pu démarrer ou non. bk_Lancer copie renvoie 0 lorsque la copie a été correctement lancée. Dans le cas contraire, un code d'erreur est retourné (reportez-vous à la liste fournie à l'Annexe A, Codes d'erreurs de 4D Backup).

Notez que la sauvegarde avec 4D Backup fonctionne dans un process indépendant, et qu'une fois la copie démarrée, la méthode continue de s'exécuter simultanément. Vous pouvez alors informer l'utilisateur de la progression de la copie, mais vous devez surtout attendre la fin de la copie avant d'appeler la commande bk_FIN SAUVEGARDE. En effet, si cette commande était appelée avant la fin de la copie, cette dernière serait annulée.

Si vous aviez ouvert le process de sauvegarde par la fonction bk_Debut mise a jour miroir, la fonction bk_Lancer copie démarrera l'envoi du fichier d'historique.

Référence

bk_Debut mise a jour miroir, bk_Debut sauvegarde integrale, bk_Lire etat.

bk_Lire etat → Entier

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Entier	←	Etat du process de sauvegarde

Description

La commande bk_Lire etat permet de suivre la sauvegarde intégrale ou la mise à jour du miroir, en vous informant sur l'état du process de 4D Backup.

bk_Lire etat peut être appelée dans les circonstances suivantes :

- avant la copie,
- pendant la copie,
- après la copie.

Il ne faut pas confondre la fonction bk_Lire etat et la fonction bk_Lire erreur. bk_Lire etat retourne une information sur l'état de la sauvegarde, ce qui permet par exemple de savoir que la copie est terminée et de clore la méthode par la commande bk_FIN SAUVEGARDE. La fonction bk_Lire erreur ne retournera un numéro d'erreur qu'en cas de problème.

Il est capital d'appeler périodiquement la fonction bk_Lire etat après avoir lancé la copie, car son rôle est double :

- elle renseigne sur l'état de la sauvegarde,
- elle alloue du temps machine au process de sauvegarde pour qu'il puisse avancer dans la copie. Si vous n'appelez pas cycliquement bk_Lire etat après avoir démarré la copie, cette dernière ne s'exécutera pas.

Les valeurs retournées par bk_Lire etat peuvent être les suivantes :

- 1 : Pas de disque sélectionné.

Cette valeur est retournée après le démarrage du process de sauvegarde, pour indiquer qu'il n'y a pas de disque sélectionné. Il vous faut en choisir un par la commande bk_FIXER VOLUME. Si vous lanciez malgré tout la sauvegarde par la fonction bk_Lancer copie, vous obtiendriez une erreur et la sauvegarde ne s'effectuerait pas.

- 2 : Impossible d'utiliser le disque sélectionné.

Le disque sélectionné par le projet ou par la commande bk_FIXER VOLUME ne convient pas pour la sauvegarde. Il ne dispose peut-être pas de la place nécessaire pour assurer la sauvegarde, ou alors il n'est plus présent. Il peut également ne pas y avoir de disquette insérée dans le lecteur.

• 3 : Prêt à copier.
Cette valeur signifie que tout est prêt pour que vous puissiez lancer la sauvegarde (le disque sélectionné est correct, le miroir est en attente, etc.).
Nous vous conseillons de procéder à ce test avant de démarrer une copie (avec la fonction bk_Lancer copie).

• 4 : Copie en cours.
La copie se déroule correctement. Vous pouvez montrer la progression de la copie en vous servant de la commande bk_LIRE PROGRESSION ou des Zones de progression %PROGRESSION SAUVEGARDE et %REMPLISSAGE DISQUE.

• 5 : Copie terminée.
La copie s'est correctement déroulée. Vous pouvez maintenant appeler la commande bk_FIN SAUVEGARDE pour conclure la sauvegarde.

• 6 : Process de sauvegarde non démarré.
Vous n'avez pas appelé la fonction bk_Debut sauvegarde integrale ou bk_Debut mise a jour miroir ou cette fonction a échoué. Le code retourné par la fonction vous informera sur la raison pour laquelle elle a échoué.

• 7 : Echec de la sauvegarde.
La fonction bk_Lancer copie a été lancée mais un problème a interrompu la sauvegarde.

Exemple

Voici un exemple d'implémentation standard de la sauvegarde, avec une gestion complète des états possibles :

```
      C_ENTIER($Etat;$Progress;$Rempliss)
      Si(bk_Debut sauvegarde integrale#0)
         ALERTE("Impossible de commencer la sauvegarde.")
      Sinon
         MESSAGE("Sauvegarde en cours…")
         bk_OUVRIR PROJET("Sauvegarde hebdomadaire")
⇒        $Etat:=bk_Lire etat
         Au cas ou
            :($Etat=1)
               ALERTE("Il n'y a pas de disque sélectionné.")
            :($Etat=2)
               ALERTE("Impossible d'utiliser le disque sélectionné.")
            :($Etat=3)
               Si(bk_Lancer copie#0)
                  ALERTE("Impossible de lancer la sauvegarde")
               Sinon
                  Repeter
                     bk_LIRE PROGRESSION($Progress;$Rempliss)
                     MESSAGE("Sauvegarde en cours : "+Chaine($Progress)+"%")
⇒                    Jusque(bk_Lire etat#4)
```

```
⇒                   Si(bk_Lire etat=5)
                        MESSAGE("La sauvegarde s'est correctement déroulée.")
                    Sinon
                        ALERTE("Problème durant la sauvegarde.")
                    Fin de si
                Fin de si
            Fin de cas
            bk_FIN SAUVEGARDE
        Fin de si
```

bk_LIRE PROGRESSION (progression; remplissage)

Paramètre	Type		Description
progression	Entier	←	Progression de la sauvegarde (en %)
remplissage	Entier	←	Remplissage du disque courant (en %)

Description

La commande bk_LIRE PROGRESSION est complémentaire aux Zones de progression proposées par 4D Backup. Elle retourne dans des variables les pourcentages respectifs de la progression de la sauvegarde et du remplissage du disque courant.

Les valeurs récupérées par cette commande sont des entiers de 0 à 100.

• Le paramètre progression vaut 0 au début de la sauvegarde et 100 à la fin.
• Le paramètre remplissage correspond au pourcentage de place utilisée sur le disque. Si la valeur vaut 0, le disque est vide, à 100 il est plein. Une valeur de 75 indique donc que 75 % de la place du disque est occupée.

Cette commande peut être utilisée pendant la sauvegarde pour assurer un suivi de la progression de la copie, par exemple à l'aide de la commande MESSAGE de 4D.

Exemple

Voici un exemple de méthode de suivi de sauvegarde :

```
C_ENTIER($Progress;$Rempliss)
Si(bk_Debut sauvegarde integrale=0)
   Si(bk_Lancer copie=0)
      Repeter
⇒        bk_LIRE PROGRESSION($Progress;$Rempliss)
         MESSAGE("Sauvegarde en cours : "+Chaine($Progress)+"%")
      Jusque(bk_Lire etat#4)
   Fin de si
   bk_FIN SAUVEGARDE
Fin de si
```

Référence

Zones de progression.

bk_Sauvegarde integrale → Entier

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Entier	←	0 si l'opération s'est bien passée, sinon Code d'erreur

Description

Si vous avez effectué une première sauvegarde intégrale et si les paramètres de cette sauvegarde ont été enregistrés dans un projet par défaut, cette fonction vous permet d'effectuer directement les sauvegardes ultérieures.

La commande bk_Sauvegarde integrale est autonome : elle n'a pas besoin d'être encadrée par les routines bk_Debut sauvegarde integrale et bk_FIN SAUVEGARDE. Son simple appel suffit à déclencher la sauvegarde de la base, à condition que tous les paramètres du projet de sauvegarde soient valides et que le volume sélectionné soit présent.

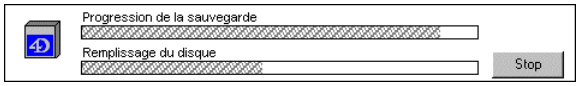
L'entier retourné par la fonction vous permet de savoir si un problème est survenu pendant la mise à jour :

• 0 : tout s'est bien passé.
• 1301 : *La sauvegarde a déjà été démarrée*. La base est déjà en cours de sauvegarde.
• 1302 : *Aucune sauvegarde n'a été lancée*. Cette erreur signifie que le process de sauvegarde n'a pu être ouvert.
• 1303 : *Destination incorrecte*. Le volume de destination est introuvable ou le chemin d'accès a été modifié.
• 1305 : *4D Backup n'est pas correctement installé dans 4D Server*.

Note : La liste complète des codes d'erreurs est fournie à l'Annexe A, Codes d'erreurs de 4D Backup.

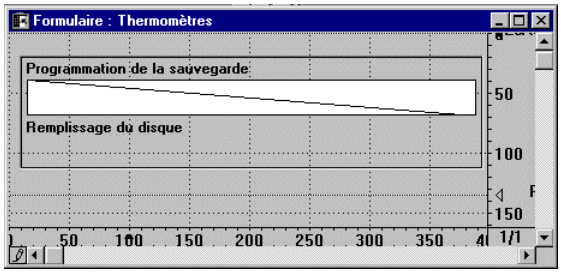
%PROGRESSION_SAUVEGARDE
%REMPLISSAGE_DISQUE

Si vous programmez dans 4D votre propre fenêtre de sauvegarde, ces zones externes vous permettent d'afficher des thermomètres personnalisés de progression de la copie et de remplissage du disque.
Vous pourrez, par exemple, simuler le fonctionnement des thermomètres de 4D Backup tels qu'ils apparaissent dans la fenêtre standard de sauvegarde :



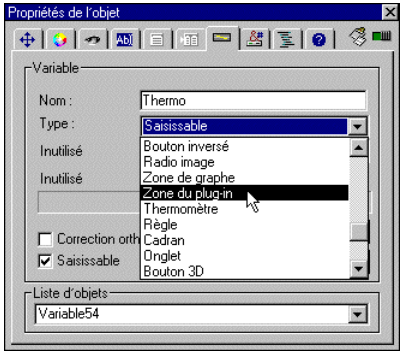
Pour installer ces zones externes :

1. Placez une variable dans un formulaire Entrée.
Pour plus d'informations sur ce point, reportez-vous au manuel *Mode Structure* de la documentation de 4D.

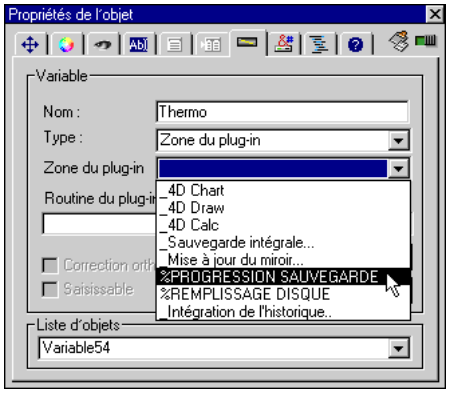


2. Double-cliquez sur la variable.
La palette de définition d'objet apparaît.

3. Donnez un nom à votre variable puis sélectionnez "Zone du plug-in" dans la liste déroulante Type :



4. Choisissez, dans le menu situé au-dessous du précédent et désormais intitulé "Zone du plug-in", %PROGRESSION SAUVEGARDE ou %REMPLISSAGE DISQUE, suivant la zone que vous désirez placer.



Note : Les intitulés _Sauvegarde intégrale et _Mise à jour du miroir (ainsi que _Intégration de l'historique lorsque module est installé) apparaissent par défaut dans ce menu, mais ne les sélectionnez pas car 4D Backup ne peut fonctionner en tant que zone externe placée dans un formulaire.

Une fois les zones externes installées dans le formulaire, elles réagiront comme les thermomètres de la fenêtre standard de sauvegarde.

• Le thermomètre de remplissage du disque indique le ratio de remplissage du disque sélectionné pour la sauvegarde.
• Le thermomètre de progression de la sauvegarde sera vide avant la sauvegarde et affichera la progression de la copie jusqu'à son terme.

Note : Vos thermomètres doivent être situés dans le même process que celui qui déclenche la sauvegarde afin de fonctionner (les autres process sont gelés).

Référence

bk_LIRE PROGRESSION.

4

BK Informations

Après chaque sauvegarde, le fichier de données conserve un certain nombre d'informations relatives à cette sauvegarde, vous permettant ultérieurement de savoir ce qui s'est passé.

Vous pouvez ainsi obtenir des informations sur la dernière sauvegarde effectuée sur la base : numéro, date et heure de cette sauvegarde.

Une routine vous informe sur la taille en octets des composants de la sauvegarde courante.

bk_Date derniere sauvegarde → Date

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Date	←	Date de la dernière sauvegarde effectuée

Description

La commande bk_Date derniere sauvegarde retourne la date de la dernière sauvegarde effectuée, que ce soit une sauvegarde intégrale ou la mise à jour d'un miroir. Si aucune sauvegarde n'a encore été réalisée, la date retournée vaudra !00/00/00!.

Exemple

La méthode suivante permet de vérifier que les sauvegardes sont bien réalisées régulièrement et indique la date de la dernière sauvegarde :

```
      C_ENTIER($Sauvegarde;$Période)
      C_DATE($DateBackup)
      $Sauvegarde:=bk_Debut sauvegarde integrale
      Si ($Sauvegarde=0)
⇒        $DateBackup:=bk_Date derniere sauvegarde
         Si ($DateBackup=!00/00/00!)
            ALERTE ("Cette base n'a pas encore été sauvegardée.")
         Sinon
            $Période:=Date du jour-$DateBackup
            Si ($Période>0)
               ALERTE ("La dernière sauvegarde date de "+Chaine($Période)+" jour(s).")
            Fin de si
         Fin de si
      Fin de si
      bk_FIN SAUVEGARDE
```

Référence

bk_Heure derniere sauvegarde.

bk_Heure derniere sauvegarde → Entier long

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Entier long	←	Heure de la dernière sauvegarde effectuée

Description

La commande bk_Heure derniere sauvegarde vous permet de connaître l'heure à laquelle a été effectuée la dernière sauvegarde de la base. La valeur retournée est le nombre de secondes écoulées entre minuit (?00:00:00?) et la fin de la dernière sauvegarde. Si la base n'a jamais été sauvegardée, bk_Heure derniere sauvegarde retourne 0.

Pour exploiter la valeur obtenue, utilisez la méthode suivante :

```
C_ENTIER LONG($vHeure;$vMinutes;$vSecondes)
Si(bk_Debut sauvegarde integrale=0)
   $vHeure:=bk_Heure derniere sauvegarde\3600
   $vMinutes:=(bk_Heure derniere sauvegarde%3600)\60
   $vSecondes:=bk_Heure derniere sauvegarde%60
Fin de si
bk_FIN SAUVEGARDE
```

Note : Bien entendu, vous pouvez aussi afficher directement l'heure dans un formulaire.

Exemple

Cette méthode affiche un message d'alerte lorsque votre dernière sauvegarde remonte à plus de huit heures.

```
C_ENTIER($vDurée)
Si (bk_Debut sauvegarde integrale=0)
   $vDurée:=Date du jour-bk_Date derniere sauvegarde
   Si ($vDurée=0)
⇒     Si ((Heure courante-bk_Heure derniere sauvegarde)>?08:00:00?)
         ALERTE ("Vous n'avez pas sauvegardé votre base depuis plus de 8 heures !")
      Fin de si
   Sinon
      ALERTE ("La base n'a pas été sauvegardée depuis "+Chaine($vDurée)+" jour(s).")
   Fin de si
Fin de si
bk_FIN SAUVEGARDE
```

Référence

bk_Date derniere sauvegarde.

bk_Lire jeu courant → Entier long

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Entier long	←	Numéro de jeu de la sauvegarde courante

Description

La commande bk_Lire jeu courant retourne le numéro de jeu de la sauvegarde courante. A chaque sauvegarde, ce compteur s'incrémente.

Le numéro de la sauvegarde est inscrit à la fin du nom du fichier de sauvegarde :

• sous Windows, il est placé à la fin du nom du fichier, celui-ci étant réduit, si nécessaire, par suppression des voyelles puis des lettres superflues. Par exemple, le document "Mbs025.4BK" est la vingt-cinquième sauvegarde de la base "MaBase.4DB".

• sous MacOS, il est placé entre crochets. Par exemple, le document "MaBase[25]" est la vingt-cinquième sauvegarde de la base "MaBase".

Le numéro du jeu ne peut être que lu. Ce numéro n'est modifié (en étant incrémenté) qu'à la sauvegarde de la base. Cette information est stockée dans le fichier de données.

Le numéro de jeu représente celui de la sauvegarde à faire. Par exemple, le numéro de jeu d'un fichier de données qui n'a jamais été sauvegardé est 1. Après la première sauvegarde il vaut 2 et ainsi de suite.

bk_LIRE TAILLES (data; structure; historique; fichiersJoints; total)

Paramètre	Type		Description
data	Numérique	←	Taille en octets du fichier de données
structure	Numérique	←	Taille en octets du fichier de structure
historique	Numérique	←	Taille en octets du fichier d'historique
fichiersJoints	Numérique	←	Taille en octets des fichiers joints
total	Numérique	←	Taille totale en octets de la sauvegarde

Description

La commande bk_LIRE TAILLES fournit la taille en octets de chacun des fichiers à sauvegarder.

La variable fichiersJoints recevra une valeur correspondant à la somme des tailles en octets des fichiers joints.

La variable total recevra la taille totale du fichier archivé. A noter que cette taille sera légèrement supérieure à la somme des tailles des fichiers précédents, puisque le fichier sauvegardé inclut un en-tête décrivant son contenu. Les codes de vérification et de réparation automatiques des données accroissent également la taille de l'archive de quelques pourcents.

Exemple

Cette méthode vous propose uniquement les volumes disposant de suffisamment de place pour réaliser votre sauvegarde :

```
      C_ENTIER($Sauvegarde;$vErr;$i)
      C_REEL($Capacité;$Utilisé;$Disponible;$Data;$Structure;$Histo;$FicJoints;$Tot)
      $vErr:=bk_Debut sauvegarde integrale
      Si ($vErr=0)
         TABLEAU ALPHA(32;TabVolume;0)
         bk_LIRE LISTE VOLUME (TabVolume)
⇒        bk_LIRE TAILLES ($Data;$Structure;$Histo;$FicJoints;$Tot)
               `Commencer la boucle à 2 pour ne pas choisir le lecteur de disquettes
         Boucle ($i;2;Taille tableau(TabVolume))
            bk_LIRE PLACE VOLUME ($i;$Capacité;$Utilisé;$Disponible)
                  `Les 100 Ko supplémentaires constituent une marge de sécurité
            Si ($Disponible>($Tot+100000))
               CONFIRMER("Souhaitez-vous faire une sauvegarde sur le volume : "+
                                                                  TabVolume{$i})
```

```
            Si (OK=1)
               bk_FIXER VOLUME ($i)
               $vErr:=bk_Lancer copie
               Repeter
               Jusque (bk_Lire etat #4)
               $i:=Taille tableau(TabVolume)  `On force la sortie de la boucle
            Fin de si
         Fin de si
      Fin de boucle
   Fin de si
   bk_FIN SAUVEGARDE
```

5

BK Miroir logique

Les routines de ce thème permettent de gérer la sélection et la mise à jour d'un miroir logique.

bk_Debut mise a jour miroir → Entier

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Entier	←	0 si l'opération s'est déroulée correctement, sinon Code d'erreur

Description

Toutes les commandes et fonctions de 4D Backup doivent être encadrées par les routines bk_Debut mise a jour miroir (ou bk_Debut sauvegarde integrale) et bk_FIN SAUVEGARDE — seules quatre routines "autonomes" font exception à ce mode de fonctionnement (reportez-vous à la section Structure des instructions).

La commande bk_Debut mise a jour miroir déclenche le process de mise à jour du miroir. L'appel de cette fonction déclenche la fermeture du fichier d'historique courant, qui reçoit l'extension ".4L2" sous Windows et le suffixe ".2" sous MacOS. Un nouveau fichier d'historique est créé.
Si vous utilisez 4D Server, le nouvel historique commence à enregistrer les opérations apportées à la base. L'envoi du fichier d'historique au miroir peut alors avoir lieu sans interférer sur le fonctionnement de la base, puisque le fichier d'historique ".4L2" ou ".2" n'est plus en activité. Cet envoi sera réalisé à l'aide de la fonction bk_Lancer copie.

Une fois l'envoi exécuté et l'historique intégré, les données de la base miroir seront sauvegardées sur le poste miroir, en fonction du nombre de jeux défini dans le projet de sauvegarde.

Note : Pour plus d'informations sur le mécanisme de mise à jour du miroir, reportez-vous au chapitre "Exploitation d'un miroir logique" dans le *Guide d'utilisation* de 4D Backup.

bk_Debut mise a jour miroir retourne 0 lorsque son exécution est correcte. Dans le cas contraire, un numéro d'erreur est renvoyé (reportez-vous à la liste fournie à l'Annexe A, Codes d'erreurs de 4D Backup).

Nous vous recommandons de tester systématiquement la valeur retournée. En effet, si bk_Debut mise a jour miroir retourne une valeur autre que 0, les instructions suivantes seront toutes ignorées.

Accès à la base durant la mise à jour

Le process de mise à jour d'un miroir logique a pour principale caractéristique de ne pas verrouiller la base de données en mode multiposte : tous les postes clients connectés à 4D Server peuvent travailler en lecture et écriture.
Avec 4e Dimension monoposte, tous les process autres que celui qui a déclenché la mise à jour sont endormis, comme pour une sauvegarde intégrale.

Traitement des transactions

Comme pour la fonction bk_Debut sauvegarde integrale, bk_Debut mise a jour miroir ne s'exécute qu'une fois toutes les transactions en cours closes, et il faut, comme pour la sauvegarde intégrale, vérifier que l'on n'a pas débuté une transaction dans le même process avant d'appeler cette commande.
Si la commande bk_FIN SAUVEGARDE est appelée alors que la mise à jour du miroir n'a pas été réalisée ou qu'elle est en cours, les deux fichiers d'historique seront concaténés afin de revenir à l'état précédent.

Exemple

(1) Voici la méthode la plus simple permettant de déclencher une mise à jour du miroir :

```
C_ENTIER($vErreur)
$vErreur:=bk_Mise a jour miroir
```

(2) Voici une méthode gérant elle-même le process de mise à jour d'une base miroir :

```
⇒   Si (bk_Debut mise a jour miroir=0)
       Si (bk_Lancer copie =0)
          Repeter
          Jusque (bk_Lire etat #4)
       Fin de si
       bk_FIN SAUVEGARDE
    Fin de si
```

(3) Vous pouvez affiner la méthode précédente en gérant les éventuelles erreurs :

```
    C_ENTIER($vErreur;$vEtat)
⇒   $vErreur:=bk_Debut mise a jour miroir
    Si ($vErreur#0)
       ALERTE("Impossible de démarrer la mise à jour : erreur N° " +Chaine($vErreur))
    Sinon
       $vErreur:=bk_Lancer copie
       Si ($vErreur#0)
          ALERTE("Impossible d'envoyer l'historique : erreur n° "+Chaine($vErreur))
       Sinon
          Repeter
             $vEtat:=bk_Lire etat
          Jusque ($vEtat#4)
          Si ($vEtat#5)
             ALERTE("Problème durant la copie : erreur N° " + Chaine($vEtat))
          Fin de si
       Fin de si
       bk_FIN SAUVEGARDE
    Fin de si
```

Référence

bk_Debut sauvegarde integrale, bk_FIN SAUVEGARDE.

bk_Mise a jour miroir → Entier

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Entier	←	0 si l'opération s'est bien déroulée, sinon Code d'erreur

Description

Si vous avez effectué une première mise à jour du miroir et que les paramètres de cette mise à jour ont été enregistrés dans un projet par défaut, cette commande vous permet d'effectuer les mises à jour ultérieures.

Note : Vous pouvez créer un projet par défaut à l'aide de la fenêtre de mise à jour du miroir appelée soit par la commande **Mise à jour du miroir...** du menu **Plug-Ins**, soit par la routine bk_FENETRE MISE A JOUR MIROIR.

La commande bk_Mise a jour miroir est autonome : elle n'a pas besoin d'être encadrée par les routines bk_Debut mise a jour miroir et bk_FIN SAUVEGARDE.
Son simple appel suffit à déclencher la mise à jour du miroir, à condition que le projet de sauvegarde soit correctement paramétré et que la base miroir existe bien sur le réseau.

L'entier retourné par la fonction vous permet de savoir si un problème est survenu pendant la mise à jour :

- 0 : tout s'est bien passé.
- 1401 : ce miroir existe mais ne correspond pas à la base de données ouverte.
- 1402 : le miroir n'a pas été trouvé.
- 1403 : cette base travaille sans fichier d'historique.

Note : La liste complète des codes d'erreurs est fournie à l'Annexe A, Codes d'erreurs de 4D Backup.

Cette fonction peut être appelée à intervalle régulier depuis un process (par exemple une fois par jour), ou encore exécutée par l'utilisateur depuis une méthode objet ou une méthode appelée par une commande de menu.

bk_LIRE LISTE ZONE(tabZones)
bk_FIXER ZONE(nomDeLaZone)
bk_LIRE LISTE MIROIR(tabMiroirs)
bk_Fixer miroir(baseMiroir;possesseur;nomDuMac) → Entier

4D Backup, à partir de la version 1.5, fonctionne avec les composants réseaux standard de 4D (pour plus d'informations sur ce point, reportez-vous au manuel d'installation). Ces routines n'ont désormais plus d'effet et sont conservées par souci de compatibilité avec les versions précédentes de 4D Backup. Elles seront supprimées dans les prochaines versions du programme.

6

BK Projets

BK Projets, Introduction

version 6.0

Toute sauvegarde effectuée avec 4D Backup se définit à travers plusieurs paramètres :

- les fichiers à sauvegarder (structure, données, historique, fichiers joints),
- la destination de la sauvegarde (volume et chemin d'accès),
- les options de sauvegarde (vérification, effacement...).

4D Backup vous permet de sauvegarder ces paramètres au sein d'un document appelé "projet". Ce fichier peut être sauvegardé et relu, afin de conserver les paramètres définis d'une sauvegarde à l'autre.

Les commandes de ce thème vous permettent aussi bien d'ouvrir et d'enregistrer ces projets que de définir tous les paramètres qui les composent.

bk_AJOUTER FICHIER JOINT (nomFichier)

Paramètre	Type		Description
nomFichier	Alpha	→	Nom du nouveau fichier à joindre

Description

La commande bk_AJOUTER FICHIER JOINT permet d'ajouter un fichier à la liste des fichiers joints. Vous pouvez saisir le nom du fichier (s'il se trouve dans le dossier de la base) ou son nom précédé du chemin d'accès complet si le fichier est à un autre endroit.

Note 4D Server : Les fichiers à joindre doivent être accessibles depuis le poste serveur : vous devez les placer sur la machine où tourne 4D Server ou sur un volume partagé lui étant accessible.

Si le document entré n'est pas valide, la commande n'aura pas d'effet. Suivant les cas, la fonction bk_Lire erreur retournera les valeurs suivantes :

• 1201 : *Mauvais fichier joint*. Ce fichier ne peut être ajouté, il s'agit du fichier de données, du fichier de structure ou de l'historique.
• 1202 : *Fichier déjà joint*. Le fichier existe déjà dans la liste des fichiers joints.
• 1203 : *Trop de fichiers joints*. Ce fichier ne peut être ajouté, la limite du nombre de fichiers joints est atteinte.
• Numéro négatif : il s'agit d'une erreur du système d'exploitation.

La liste complète des codes d'erreurs est fournie à l'Annexe A, Codes d'erreurs de 4D Backup.

Exemple

Cet exemple utilise la boîte de dialogue standard d'ouverture de fichiers. Il n'est possible qu'avec 4e Dimension monoposte.

```
C_REEL($Ref)
C_TEXTE($NomFic)
Si(bk_Debut sauvegarde integrale=0)
   $Ref:=Ouvrir document("")   `Affichage du dialogue d'ouverture de fichiers
   Si(OK=1) `Si l'utilisateur a validé
      $NomFic:=document   `Lecture  du nom du fichier sélectionné
      FERMER DOCUMENT($Ref)   `Fermeture de ce document
⇒     bk_AJOUTER FICHIER JOINT($NomFic) `Et ajout à la liste
      bk_SAUVER PROJET   `On enregistre le projet
   Fin de si
   bk_FIN SAUVEGARDE
Fin de si
```

Référence

bk_ENLEVER FICHIER JOINT, bk_LIRE FICHIERS JOINTS.

bk_ENLEVER FICHIER JOINT (nomFichier)

Paramètre	Type		Description
nomFichier	Alpha	→	Nom du fichier à supprimer de la liste

Description

La commande bk_ENLEVER FICHIER JOINT permet de supprimer un fichier de la liste des fichiers joints. Ce fichier ne sera pas effacé du disque mais disparaîtra simplement de la liste.

Vous devez passer dans nomFichier le nom du fichier avec son extension sous Windows.

Note : Vous devez donner juste le nom du document, sans le chemin d'accès complet.

Si ce fichier n'existe pas dans la liste, la commande n'aura pas d'effet, et la fonction bk_Lire erreur retournera 1204 (impossible de retirer ce fichier, il n'appartient à la liste des fichiers joints).

Référence

bk_AJOUTER FICHIER JOINT, bk_LIRE FICHIERS JOINTS.

bk_FIXER OPTIONS (data; structure; historique; vérification; effacement; nbJeux; supprAvant; incrStructure)

Paramètre	Type		Description
data	Entier	→	Sauvegarde du fichier de données
structure	Entier	→	Sauvegarde du fichier de structure
historique	Entier	→	Sauvegarde du fichier d'historique
vérification	Entier	→	Vérification des données lors de l'écriture
effacement	Entier	→	Effacement du disque avant sauvegarde
nbJeux	Entier	→	Nombre de jeux de sauvegarde
supprAvant	Entier	→	Suppression de l'ancienne archive
incrStructure	Entier	→	Incrémentation du numéro de jeu

Description

La commande bk_FIXER OPTIONS fixe les options de la sauvegarde.

Les paramètres data, structure et historique indiquent si les fichiers correspondants sont sélectionnés pour la sauvegarde :
• si vous passez 1, le fichier fera partie de la sauvegarde,
• si vous passez 0, le fichier ne fera pas partie de la sauvegarde,
• si vous passez -1, le paramétrage précédent sera conservé.

Les paramètres vérification et effacement sont fixés de la même manière :
• si vous passez 1, l'option est sélectionnée,
• si vous passez 0, l'option n'est pas sélectionnée,
• si vous passez -1, le paramétrage précédent sera conservé.

Ces options correspondent aux cases à cocher équivalentes situées dans la fenêtre de paramétrage de la sauvegarde.

A noter que le paramètre effacement n'a d'effet que lors d'une sauvegarde sur disque amovible autre qu'une disquette. En effet, 4D Backup efface (et sous MacOS renomme) systématiquement les disquettes et, à l'inverse, l'effacement d'un disque dur non amovible est impossible. Pour savoir si un volume donné est éjectable, utilisez la commande bk_LIRE INFOS VOLUME.

Le paramètre nbJeux fixe le nombre de jeux de sauvegarde à conserver. La valeur doit être comprise entre 1 et 100. Si vous passez une valeur négative, la valeur précédente sera conservée.

A noter que le paramètre “nombre de jeux” est défini dans le projet mais est stocké dans le fichier de données de la base si la sauvegarde a effectivement lieu, pour éviter tout effacement involontaire d’archive lorsque vous utilisez plusieurs projets. La valeur passée dans la variable nbJeux s’appliquera donc quelque soit le projet.

Si supprAvant vaut 1, l’ancienne archive sera détruite avant le début de la sauvegarde courante, si supprAvant vaut 0, l’ancienne archive sera détruite après. Si supprAvant vaut -1, la valeur précédente sera conservée.

Si incrStructure vaut 1, le numéro de la sauvegarde sera incrémenté lors de la sauvegarde de la structure seule, sinon incrStructure vaudra 0. Si incrStructure vaut -1, la valeur précédente sera conservée.

Référence

bk_LIRE OPTIONS.

bk_LIRE FICHIERS JOINTS (tabFichiers)

Paramètre	Type		Description
tabFichiers	Tab alpha	←	Liste des fichiers joints

Description

La commande bk_LIRE FICHIERS JOINTS remplit un tableau de type alphanumérique avec la liste des fichiers joints. Seul le nom de chacun des fichiers est donné, sans son chemin d'accès.
Cette commande vous permet d'afficher dans un formulaire la liste des fichiers joints.

Pour que votre tableau Alpha soit rempli, vous devez déclarer un nombre de caractères correspondant à la taille maximale des noms de fichiers autorisée par le système d'exploitation :
• MacOS : 31 caractères,
• Windows 95, Windows NT : 255 caractères.

Si la base est susceptible d'être exploitée sur plusieurs plates-formes, il est donc préférable de dimensionner votre tableau à 255 éléments.

Note 4D Server : La liste des fichiers joints sera lue depuis le poste serveur (et non depuis un poste client). Les noms des fichiers devront donc correspondre à des fichiers placés sur la machine où tourne 4D Server ou sur un volume partagé lui étant accessible.

Exemple

L'exemple suivant affiche une alerte lorsqu'aucun fichier n'est joint :

```
    Si(bk_debut sauvegarde integrale=0)
       TABLEAU ALPHA(255;TabFicJoint;0)
⇒      bk_LIRE FICHIERS JOINTS(TabFicJoint)
       Si(Taille tableau(TabFicJoint)=0)
          ALERTE("Vous n'avez pas de fichiers joints !")
       Fin de si
       bk_FIN SAUVEGARDE
    Fin de si
```

Référence

bk_AJOUTER FICHIER JOINT, bk_ENLEVER FICHIER JOINT.

bk_LIRE NOMS (data; structure; historique)

Paramètre	Type		Description
data	Alpha	←	Nom du fichier de données
structure	Alpha	←	Nom du fichier de structure
historique	Alpha	←	Nom du fichier d'historique

Description

La commande bk_LIRE NOMS retourne dans les variables alphanumériques passées en paramètres le nom des différents fichiers composant votre base de données. Ces informations peuvent vous être utiles pour, par exemple, personnaliser une boîte de dialogue de gestion de sauvegarde.

Vous récupérez uniquement le nom de ces fichiers (avec leur extension sous Windows), sans leur chemin d'accès.

Si votre fichier de données est segmenté, c'est le nom du premier segment qui sera retourné.

Si la base travaille sans fichier d'historique, une chaîne vide est retournée dans historique.

Exemple

Cette méthode affiche une alerte si l'utilisateur travaille sans historique :

```
      Si(bk_Debut sauvegarde integrale=0)
         C_ALPHA(80;$Data;$Structure;$Historique)
⇒        bk_LIRE NOMS($Data;$Structure;$Historique)
         Si($Historique="")
            ALERTE("Attention, vous travaillez sans fichier d'historique !")
         Fin de si
         bk_FIN SAUVEGARDE
      Fin de si
```

bk_LIRE OPTIONS (data; structure; historique; vérification; effacement; nbJeux; supprAvant; incrStructure)

Paramètre	Type		Description
data	Entier	←	Sauvegarde du fichier de données
structure	Entier	←	Sauvegarde du fichier de structure
historique	Entier	←	Sauvegarde du fichier d'historique
vérification	Entier	←	Vérification des données lors de l'écriture
effacement	Entier	←	Effacement du disque avant sauvegarde
nbJeux	Entier	←	Nombre de jeux de sauvegarde
supprAvant	Entier	←	Suppression de l'ancienne archive
incrStructure	Entier	←	Incrémentation du numéro de jeu

Description

La commande bk_LIRE OPTIONS retourne dans les variables entières passées en arguments les options courantes de la sauvegarde.

Les variables data, structure et historique indiquent si les fichiers correspondants sont sélectionnés pour la sauvegarde :
• si la variable vaut 1, le fichier fait partie de la sauvegarde,
• si la variable vaut 0, le fichier ne fait pas partie de la sauvegarde.

Les variables vérification et effacement indiquent si les options correspondantes ont été sélectionnées :
• si la variable vaut 1, l'option a été sélectionnée,
• si la variable vaut 0, l'option n'a pas été sélectionnée.

Ces options correspondent aux cases à cocher équivalentes situées dans la fenêtre de paramétrage de la sauvegarde.

A noter que le paramètre effacement n'a d'effet que lors d'une sauvegarde sur disque amovible autre qu'une disquette. En effet, 4D Backup efface (et sous MacOS renomme) systématiquement les disquettes et, à l'inverse, l'effacement d'un disque dur non amovible est impossible.

La variable nbJeux reçoit le nombre de jeux spécifié. Le nombre de jeux par défaut est de 3. Il peut être modifié par programmation ou en cliquant sur les petites flèches situées en regard de l'information, dans la fenêtre de paramétrage de la sauvegarde.
A noter que le paramètre "nombre de jeux" est lu et stocké dans le fichier de données de la base, pour éviter tout effacement involontaire d'archive lorsque vous utilisez plusieurs projets. La valeur retournée par la variable nbJeux est donc identique pour tous les projets de la base.

Si supprAvant vaut 1, l'ancienne archive est détruite avant le début de la sauvegarde courante. Si supprAvant vaut 0, l'ancienne archive est détruite après.

Si incrStructure vaut 1, le numéro de la sauvegarde est incrémenté lors de la sauvegarde de la structure seule, sinon incrStructure vaut 0.

Référence

bk_FIXER OPTIONS.

bk_OUVRIR PROJET (nomProjet)

Paramètre	Type		Description
nomProjet	Alpha	→	Nom du projet à ouvrir

Description

La commande bk_OUVRIR PROJET ouvre le document dont le nom est spécifié par nomProjet. Ce document doit être un projet généré par 4D Backup, soit directement depuis l'interface de 4D Backup, soit par la commande bk_SAUVER PROJET.

Vous pouvez donner soit directement le nom du document (par exemple "Sauvquot.4BP" sous Windows ou "Sauvegarde quotidienne" sous MacOS) si le projet se trouve dans le dossier de la base, soit le nom précédé du chemin d'accès complet ("C:\Dossier\Sauvquot.4BP"sous Windows ou "Disque Dur:Dossier:Sauvegarde quotidienne" sous MacOS).

La fonction bk_Lire erreur vous permettra de gérer les éventuelles erreurs liées au projet à ouvrir.

Si vous appelez la fonction bk_Lire erreur après avoir appelé la commande bk_OUVRIR PROJET, vous récupérerez les valeurs suivantes :

• 0 (pas d'erreur) : l'ouverture du document s'est bien passée.
• 1101 : *Mauvais projet*. Le projet ne correspond pas à la base ouverte (il a été construit avec une autre base, et ne peut s'appliquer à celle actuellement ouverte).
• 1102 : *Pas un projet*. Le document ouvert n'est pas un projet (il s'agit d'un autre type de document, qui n'a donc pu être ouvert en tant que projet).
• Numéro d'erreur négatif : l'erreur provient du système d'exploitation.

Note : La liste complète des codes d'erreurs est fournie à l'Annexe A, Codes d'erreurs de 4D Backup.

Une autre fonction de 4D Backup, bk_Lire etat, vous permet quant à elle de gérer plus précisément les erreurs liées aux volumes décrits dans le fichier projet. Après l'appel à bk_OUVRIR PROJET, voici les différents "états" possibles retournés par la fonction bk_Lire etat :

• Si le volume sélectionné dans le projet n'est plus présent, l'état vaudra 1 ("Pas de disque sélectionné").
• Si le volume ne contient par exemple pas la place nécessaire à la copie, l'état vaudra 2 ("Impossible d'utiliser ce disque").
• Si la sauvegarde est prête à démarrer, bk_Lire etat retournera 3 ("Prêt à copier").

Dans les deux premiers cas, il vous faudra soit gérer la recherche d'un volume disponible d'une taille suffisante, soit avertir l'utilisateur de l'incident et abandonner la sauvegarde (par la commande bk_FIN SAUVEGARDE) puis lui proposer par exemple la fenêtre standard de sauvegarde intégrale.

Note 4D Server : Le projet sera lu sur le poste serveur (et non depuis un poste client). Le nom et le chemin d'accès s'appliqueront donc à un projet placé sur un disque du poste serveur ou sur un volume partagé accessible depuis le poste serveur.

Exemple

Imaginez que vous ayez 7 projets de sauvegarde différents correspondant aux 7 jours de la semaine, pour répartir les sauvegardes chaque jour sur des disques différents, et profiter du week-end pour sauvegarder un grand nombre de fichiers joints.
Les projets s'appellent "Jour1" sous MacOS ("Jour1.4BP" sous Windows), "Jour2"... "Jour7", et se trouvent dans le même répertoire que les données de la base. Ils ont été réalisés depuis la fenêtre standard de sauvegarde de 4D Backup (rappelons que pour 4e Dimension, dimanche=1, lundi=2... samedi=7).

```
C_ENTIER($Erreur)
Si(bk_Debut sauvegarde integrale#0)
   ALERTE("Impossible de démarrer la sauvegarde")
Sinon
⇒     bk_OUVRIR PROJET("Jour"+Chaine(Numero du jour(Date du jour)))
      $Erreur:=bk_Lire erreur
      Si($Erreur#0)    `Si une erreur survient
         ALERTE("Erreur lors de l'ouverture du projet : "+bk_Erreur texte($Erreur))
      Sinon
         Si(bk_Lire etat#3)  `Si l'état est différent de "Prêt à copier"
            ALERTE("Le projet n'est plus valide. La base n'a pu être sauvegardée.")
         Sinon
            Si(bk_Lancer copie=0)  `Si le démarrage de la copie s'est bien passé
               Repeter
               Jusque (bk_Lire etat#4) `Attente de la fin de la copie
            Fin de si
         Fin de si
      Fin de si
      bk_FIN SAUVEGARDE
Fin de si
```

Référence

bk_Erreur texte, bk_Lire erreur, bk_Lire etat, bk_SAUVER PROJET.

bk_SAUVER PROJET {(nomProjet)}

Paramètre	Type		Description
nomProjet	Alpha	→	Nom du projet à sauvegarder

Description

La commande bk_SAUVER PROJET sauvegarde le projet dont le nom est spécifié par nomProjet.
Si aucun projet de ce nom n'existe déjà, il est créé. S'il existait déjà, son contenu est remplacé par les nouveaux paramètres. Les paramètres stockés seront ceux actuellement en mémoire, que ce soient les paramètres par défaut si vous n'aviez pas de projet, les paramètres du projet ouvert ou ceux modifiés par les commandes du langage.

Vous pouvez passer soit directement le nom du document (par exemple "Sauvegarde quotidienne" sous MacOS ou "SauvQuot.4BP" sous Windows), soit le nom précédé du chemin d'accès complet (par exemple "DisqueDur:Dossier:Sauvegarde quotidienne" sous MacOS ou "C:\Dossier\SauvQuot.4BP" sous Windows). Dans le premier cas, le projet sera sauvegardé dans le dossier de la base.

Note 4D Server : Le projet sera sauvegardé depuis la machine où tourne 4D Server et non celle où tourne 4D Client. Si vous spécifiez un chemin d'accès complet, il devra donc correspondre au disque de cette machine ou à un volume partagé lui étant accessible.

Si vous omettez le paramètre nomProjet, le projet courant sera sauvegardé, que ce soit le projet par défaut ("Projet de sauvegarde" sous MacOS, "Backup.4BP" sous Windows) ou un projet ouvert par la commande bk_OUVRIR PROJET.
Si aucun projet n'avait été préalablement ouvert, le projet par défaut sera sauvegardé.

Si vous appelez la fonction bk_Lire erreur après avoir utilisé la commande bk_SAUVER PROJET, elle retournera les valeurs suivantes :

- 0 : la sauvegarde du document s'est bien déroulée.
- Numéro d'erreur négatif : l'erreur provient du système d'exploitation.

Note : La liste complète des codes d'erreurs est fournie à l'Annexe A, Codes d'erreurs de 4D Backup.

Référence

bk_Lire erreur, bk_OUVRIR PROJET.

7

BK Utilitaires

Ce thème regroupe les routines de gestion des erreurs proposées par 4D Backup.

bk_Erreur texte (codeErreur) → Texte

Paramètre	Type		Description
codeErreur	Entier	→	Code de l'erreur
Résultat	Texte	←	Texte de l'erreur

Description

La commande bk_Erreur texte retourne le texte complet décrivant l'erreur obtenue. Ce texte peut être utile pour informer l'utilisateur de la nature du problème. Cette commande ne fonctionne que pour les erreurs propres à 4D Backup, donc ayant un code positif.

Le texte complet des erreurs est fourni à l'Annexe A, Codes d'erreurs de 4D Backup, avec la liste des codes d'erreurs.

Exemple

L'exemple suivant propose une gestion poussée des erreurs provenant de l'ouverture du fichier de projet :

```
      C_ENTIER($erreur)
      Si(bk_Debut sauvegarde integrale#0)
         ALERTE("Impossible de lancer la sauvegarde.")
      Sinon
         bk_OUVRIR PROJET("SauvQuot.4BP")
⇒        $erreur:=bk_Lire erreur
         Au cas ou
            : ($erreur>0)
⇒              ALERTE(bk_Erreur texte($erreur))
            : ($erreur=-43)
               ALERTE("Le projet n'a pu être trouvé.")
            : ($erreur=-49)
               ALERTE("Le projet est déjà ouvert par une autre application.")
            : ($erreur<0)
               ALERTE("L'erreur n°"+Chaine($erreur)+" s'est produite. Le projet n'a pu être
                                                                                  ouvert.")
            : ($erreur=0)
               ...          `On effectue la sauvegarde
         Fin de cas
         bk_FIN SAUVEGARDE
      Fin de si
```

Référence

bk_Lire erreur.

bk_Lire erreur → Entier

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Entier	←	Numéro de la dernière erreur rencontrée par 4D Backup

Description

La commande bk_Lire erreur retourne le numéro de la dernière erreur rencontrée par 4D Backup.

Si la dernière commande n'a pas généré d'erreur, bk_Lire erreur retourne 0. Veillez donc à appeler cette fonction après chaque commande risquant de provoquer une erreur, par exemple les commandes travaillant sur des documents, afin de détecter les éventuelles erreurs disques.

Si le code d'erreur retourné est positif, il s'agit d'une erreur propre à 4D Backup. La liste complète des codes d'erreurs de 4D Backup se trouve à l'Annexe A, Codes d'erreurs de 4D Backup.
Si le code est négatif, il s'agit d'une erreur provenant du système d'exploitation. Les erreurs du système les plus fréquentes sont fournies à l'Annexe A, Codes d'erreurs de 4D Backup.

Note : Afin d'assurer l'indépendance de plate-forme de vos bases, les codes d'erreurs liés au système d'exploitation retournés par 4D Backup sont identiques sous MacOS et sous Windows, le programme se chargeant de les convertir si nécessaire. La table des codes d'erreurs utilisée est celle de MacOS. Reportez-vous à l'annexe A pour plus d'informations.

Exemple

Reportez-vous à l'exemple de la commande bk_Erreur texte.

Référence

bk_Erreur texte.

8

BK Volumes

BK Volumes, Introduction

version 6.0

Les commandes de ce thème vous donnent des informations sur les volumes de sauvegarde. Vous pouvez connaître le numéro du volume sélectionné, obtenir la liste des volumes connectés, changer de volume ou encore récupérer des informations précises sur un volume donné.

Remarque : Certaines routines attendent un numéro de volume : ce numéro correspond à l'indice de ce volume dans le tableau retourné par la commande bk_LIRE LISTE VOLUME.

bk_EJECTER DISQUE (numVol)

Paramètre	Type		Description
numVol	Entier	→	Numéro du volume dans la liste retournée par bk_LIRE LISTE VOLUME

Description

La commande bk_EJECTER DISQUE permet d'éjecter le volume dont le numéro est indiqué par numVol.
Bien entendu, bk_EJECTER DISQUE ne s'applique que sous MacOS.
Pour savoir si un volume est éjectable, il vous faut utiliser la commande bk_LIRE INFOS VOLUME.

Note : Lors d'une sauvegarde segmentée sur plusieurs volumes éjectables, 4D Backup prend l'initiative d'éjecter les volumes pleins et de réclamer les suivants. Il n'est donc pas nécessaire de surveiller par programmation le taux de remplissage des disques pendant la copie, ni de programmer leur éjection.

Référence

bk_LIRE INFOS VOLUME, bk_LIRE LISTE VOLUME.

bk_FIXER NOM FICHIER (nomFichier)

Paramètre	Type		Description
nomFichier	Alpha	→	Nom du fichier de sauvegarde

Description

La commande bk_FIXER NOM FICHIER permet de fixer le nom du fichier de sauvegarde. Vous pouvez soit donner le nom précédé du chemin d'accès complet, soit simplement le nom du fichier.

4D Backup ajoutera automatiquement à ce nom le numéro de la sauvegarde et, sous Windows, l'extension ".4BK" il est donc inutile de les gérer. Vous passez uniquement à la commande le chemin d'accès suivi du nom du fichier.

• Sous Windows, "C:\Dossier\Sauvegarde" créera par exemple le fichier "Svgrd035.4BK" dans le dossier "Dossier" sur le disque "C:".

• Sous MacOS, "Hector:MonDossier:Sauvegarde" créera par exemple le fichier "Sauvegarde[35]" dans le dossier "MonDossier" sur le disque "Hector".

Si vous donnez simplement le nom du fichier de sauvegarde sans le chemin d'accès, le fichier se créera au niveau principal du volume précédemment choisi.

bk_FIXER VOLUME (numVol)

Paramètre	Type		Description
numVol	Entier	→	Numéro du volume dans la liste retournée par bk_LIRE LISTE VOLUME

Description

La commande bk_FIXER VOLUME permet de changer le volume de sauvegarde.

Le chemin d'accès à l'ancien fichier de sauvegarde qui pouvait exister est réinitialisé. Par conséquent, à moins que vous ne spécifiiez à nouveau un chemin par la commande bk_FIXER NOM FICHIER, la sauvegarde suivante se fera au niveau principal du disque choisi.

Référence

bk_FIXER NOM FICHIER, bk_LIRE LISTE VOLUME, bk_Lire volume.

bk_Lire icone volume (numVol) → Image

Paramètre	Type		Description
numVol	Entier	→	Numéro du volume dans la liste retournée par bk_LIRE LISTE VOLUME
Résultat	Image	←	Icône du volume

Description

La commande bk_Lire icone volume retourne une image 4D contenant l'icône du volume considéré. Cette fonction vous permet d'afficher pour l'utilisateur une boîte de dialogue de sauvegarde où il pourra visualiser ses volumes, de la même façon que dans la fenêtre standard de sauvegarde.

Exemple

Cet exemple crée un tableau image contenant la liste des volumes présents. Placé dans la méthode objet du bouton d'une boîte de dialogue personnalisée de sauvegarde, il permet d'afficher dans un tableau la liste des volumes de façon visuelle (pensez à choisir une police de 32 points afin que les lignes du tableau aient une taille suffisante pour vous permettre de voir les images).

```
C_ENTIER($i)
TABLEAU ALPHA(32;tabVolume;0)
Si(bk_Debut sauvegarde integrale#0)
   ALERTE("Sauvegarde impossible.")
Sinon
   bk_LIRE LISTE VOLUME(tabVolume)
   TABLEAU IMAGE(tabIcone;Taille tableau(tabVolume))
   Boucle($i;1;Taille tableau(tabVolume))
⇒     tabIcone{$i}:=bk_Lire icone volume($i)
   Fin de boucle
Fin de si
bk_FIN SAUVEGARDE
```

Référence

bk_LIRE LISTE VOLUME.

bk_LIRE INFOS VOLUME

version 6.0

bk_LIRE INFOS VOLUME (numVol; nomVol; verrouillé; éjectable; présent; typeVol)

Paramètre	Type		Description
numVol	Entier	→	Numéro du volume dans la liste retournée par bk_LIRE LISTE VOLUME
nomVol	Alpha	←	Nom du volume
verrouillé	Entier	←	Le volume est-il verrouillé ?
éjectable	Entier	←	Le volume est-il éjectable ?
présent	Entier	←	Le volume éjectable est-il présent ?
typeVol	Entier	←	Type du volume

Description

La commande bk_LIRE INFOS VOLUME permet d'obtenir des informations sur le volume dont le numéro est numVol.

Les variables passées en paramètres reçoivent les informations suivantes :

• nomVol reçoit le nom du volume. Sous MacOS, ce nom est celui qui apparaît au niveau du Finder. Sa longueur maximale est de 27 caractères. Sous Windows, ce nom est la lettre du volume suivie de deux points ("C:").
Les variables suivantes correspondent aux propriétés particulières de ce volume.

• Si verrouillé vaut 1, le volume est verrouillé en écriture et ne pourra pas être modifié (ce peut être par exemple un lecteur de CD-ROM). Vous ne pouvez pas, bien entendu, sélectionner ce volume pour la sauvegarde. La commande bk_Lire etat retournerait dans ce cas la valeur 2 (impossible d'utiliser le disque sélectionné). Si verrouillé vaut 0, le disque est libre en écriture, et peut être utilisé pour la sauvegarde.

• Si éjectable vaut 1, le volume concerné est éjectable. Vous pourrez dans ce cas demander à ce que le contenu du volume soit effacé avant la sauvegarde, par la commande bk_FIXER OPTIONS, ou encore l'éjecter par la commande bk_EJECTER DISQUE. Si éjectable vaut 0, le volume n'est pas éjectable.

• Si présent vaut 1, cela signifie que le disque est physiquement présent. Bien entendu, cette information n'a de valeur que s'il s'agit d'un disque éjectable. Dans ce cas, si le disque est éjecté, présent vaudra 0. Les disques non éjectables retournent toujours 1 dans présent.

• La valeur typeVol permet d'obtenir des informations sur le type de volume présent. typeVol vaut 1 pour un disque dur, 3 pour un lecteur de disquettes de 800 Ko et 4 pour un lecteur de disquettes de 1,4 Mo.

Référence

bk_LIRE LISTE VOLUME.

bk_LIRE LISTE VOLUME (tabNomsVol)

Paramètre	Type		Description
tabNomsVol	Tableau alpha	←	Tableau des noms des volumes

Description

La commande bk_LIRE LISTE VOLUME remplit le tableau alpha de 32 caractères passé en paramètre avec les noms des volumes présents. Si ce tableau contenait déjà des éléments, ils seront effacés et remplacés par les noms des volumes. L'ordre dans lequel les volumes sont stockés dans le tableau correspond à l'ordre dans lequel ils apparaissent dans la fenêtre standard de sauvegarde.

Note : Sous Windows, cet ordre correspond à celui des lettres des volumes tels qu'il sont été définis (A:, B:, C:...).
Sous MacOS, l'ordre est celui dans lequel les volumes ont été "montés", c'est-à-dire en premier lieu le ou les lecteurs de disquettes, puis le disque dur de démarrage, puis les autres volumes.
Attention, il n'est pas garanti que les numéros soient conservés d'une sauvegarde à l'autre. Par exemple, si un disque dur externe est éteint ou un volume partagé indisponible, l'ordre changera.

Exemple

Nous voulons que la sauvegarde se fasse systématiquement sur le disque nommé "Backup", quel qu'il soit et où qu'il soit. De cette façon, l'utilisateur pourra changer de volume de sauvegarde simplement en renommant ses volumes.

```
      C_ENTIER($VolBackup)
      TABLEAU ALPHA(32;TabNomsVol;0)
      Si(bk_Debut sauvegarde integrale#0)
         ALERTE("Sauvegarde impossible.")
      Sinon
⇒        bk_LIRE LISTE VOLUME(TabNomsVol)
         $VolBackup:=Chercher dans tableau(TabNomsVol;"Backup")
         Si($VolBackup=-1)
            ALERTE("Le volume 'Backup' n'est pas présent : Sauvegarde impossible.")
         Sinon
            Si(bk_Lire volume#$VolBackup)   `Si le numéro du volume a changé,
               bk_FIXER VOLUME($VolBackup)   `on fixe le nouveau numéro
               bk_SAUVER PROJET   `et on sauve le nouveau projet.
            Fin de si
```

```
            Si(bk_Lancer copie#0)
               ALERTE("Impossible de lancer la sauvegarde.")
            Sinon
               Repeter
               Jusque(bk_Lire etat#4)   `On attend la fin de la sauvegarde
               Si(bk_Lire etat#5)
                  ALERTE("Problème durant la sauvegarde.")
               Fin de si
            Fin de si
         Fin de si
         bk_FIN SAUVEGARDE
      Fin de si
```

Référence

bk_LIRE INFOS VOLUME, bk_LIRE PLACE VOLUME.

bk_Lire nom fichier

bk_Lire nom fichier → Alpha

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Alpha	←	Nom courant du fichier de sauvegarde

Description

La commande bk_Lire nom fichier retourne le nom courant du fichier de sauvegarde.

- Sous Windows, ce nom est de la forme “Base021.4BK”.
- Sous MacOS, il est suivi de son numéro de sauvegarde entre crochets (“Base.data[21]”).

Le chemin d’accès complet n’est pas retourné.

Référence

bk_FIXER NOM FICHIER.

bk_LIRE PLACE VOLUME (numVol; total; utilisé; disponible)

Paramètre	Type		Description
numVol	Entier	→	Numéro du volume dans la liste retournée par bk_LIRE LISTE VOLUME
total	Numérique	←	Capacité maximale du disque
utilisé	Numérique	←	Espace disque utilisé
disponible	Numérique	←	Place disponible sur le disque

Description

La commande bk_LIRE PLACE VOLUME est destinée à vous informer sur les capacités et l'encombrement du volume désigné par numVol. Vous passez dans le paramètre numVol le numéro du disque à inspecter. Les variables entières total, utilisé et disponible reçoivent respectivement la capacité maximale du disque, l'espace utilisé par les données se trouvant actuellement sur le disque et la place restante, le tout exprimé en octets.

Rappelons qu'un Kilo-octet (Ko) vaut 1024 octets, et qu'un Méga-octet (Mo) vaut 1024 Ko. La capacité des disques se calcule usuellement en Méga-octets (on parle de disques durs de 200 Mo, 600 Mo, etc...). On parle pour les disques durs les plus importants en Giga-octets (Go), valant 1024 Méga-octets (MacOS — version 7.5 — peut gérer des disques jusqu'à 4 Go ; sous Windows NT, la taille maximale des disques peut être beaucoup plus importante.)

bk_Lire volume → Entier

Paramètre	Type		Description
Cette commande ne requiert pas de paramètre			
Résultat	Entier	←	Numéro du disque sélectionné

Description

La commande bk_Lire volume retourne le numéro du disque sélectionné pour la sauvegarde courante.

Référence

bk_FIXER VOLUME, bk_LIRE LISTE VOLUME.

9

Annexes

Cette section récapitule les codes d'erreurs renvoyés par 4D Backup.

Ces codes d'erreurs peuvent être récupérées par la fonction bk_Lire erreur, qui renvoie le numéro de la dernière erreur rencontrée. L'intitulé de l'erreur est retourné par la fonction bk_Erreur texte.

Le premier tableau fournit la liste des erreurs retournées par l'application 4D Backup. La liste des erreurs système les plus communes pouvant être rencontrées par 4D Backup est fournie à la suite de ce tableau.

La plupart des fonctions de 4D Backup peuvent également renvoyer un code d'erreur. Lorsque le déroulement d'une instruction est correct, 4D Backup retourne la valeur 0.

- **Erreurs retournées par 4D Backup (code >0)**

Code	Texte de l'erreur 4D Backup *Commentaire*
1000	Une erreur s'est produite *Une erreur est survenue durant la sauvegarde.*
1101	Mauvais projet *Ce projet ne correspond pas à la base ouverte.*
1102	Pas un projet *Le document spécifié n'est pas un fichier projet.*
1103	Rien à sauvegarder *Aucun fichier n'a été spécifié pour la sauvegarde.*
1201	Mauvais fichier joint *Ce fichier ne peut être ajouté, il s'agit du fichier de données, de structure ou d'historique.*
1202	Fichier déjà joint *Ce fichier existe déjà dans la liste des fichiers joints.*
1203	Trop de fichiers joints *Ce fichier ne peut être ajouté, la limite du nombre de fichiers joints est atteinte.*
1204	Ce n'est pas un fichier joint *Impossible de retirer ce fichier de la liste des fichiers joints : il n'appartient pas à la liste.*
1301	La sauvegarde a déjà été démarrée *Le process de sauvegarde a déjà été créé.*
1302	Sauvegarde non démarrée *Le process de sauvegarde n'a pas été créé. Appelez bk_Debut sauvegarde integrale ou bk_Debut mise a jour miroir.*
1303	Destination incorrecte *Le volume de destination est introuvable ou le chemin d'accès a été modifié.*
1304	La sauvegarde ne peut pas être démarrée *Vérifiez que vous avez correctement installé 4D Backup.*

1305 4D Backup n'est pas correctement installé dans 4D Server.
Vérifiez que vous avez correctement installé 4D Backup.
1401 Mauvais miroir
Ce miroir existe mais ne correspond pas à la base de données ouverte.
1402 Miroir non trouvé
Ce miroir n'existe pas ou n'est pas disponible.
1403 Pas d'historique
Cette base travaille sans fichier d'historique.
1404 Une transaction est ouverte dans le même process
Validez ou annulez votre transaction avant d'appeler 4D Backup.
1405 Changement d'historique impossible
1406 Un ancien historique existe déjà
Un fichier "MaBase.log.2" ou "MaBase.4L2" existe déjà.
1407 Le nouvel historique ne peut pas être créé.
1408 Le nouvel historique ne peut pas être ouvert.

• Erreurs liées au système d'exploitation (code <0)

Lorsque la fonction bk_Lire erreur renvoie un numéro d'erreur négatif, cela signifie que l'erreur a été détectée par le système d'exploitation.

Note : Pour assurer l'indépendance de plate-forme de vos bases, les codes d'erreurs liés au système d'exploitation renvoyés par 4D Backup sont identiques, quelle que soit la plate-forme utilisée. Ces codes sont basés sur la liste des erreurs de MacOS. Ainsi, il n'est pas nécessaire de réécrire vos méthodes de gestion des erreurs.

Code	Signification de l'erreur
-34	Le disque est plein
-36	Disque probablement défectueux
-43	Fichier introuvable
-44	Disquette protégée en écriture
-48	Le nom de ce fichier existe déjà
-49	Fichier déjà ouvert
-108	Mémoire saturée

Référence

bk_Erreur texte, bk_Lire erreur.

Annexe B, Redémarrage rapide

Cette annexe résume les opérations à effectuer pour redémarrer l'exploitation de votre base après un incident, en fonction du type d'incident et du mode de sauvegarde choisi. Pour plus de détails, reportez-vous au *Guide d'utilisation* de 4D Backup.

En cas d'arrêt accidentel de l'exploitation

L'arrêt inopiné de l'exploitation d'une base de données peut avoir plusieurs causes : coupure de courant, panne d'un élément du système, etc.

Trois cas peuvent alors se présenter, en fonction de l'état du cache de données de 4D :

• L'incident se produit alors que le cache de données est vide (aucune modification n'a été apportée sur la base depuis la dernière écriture du cache sur le disque).
• L'incident se produit alors que le cache contient des modifications de la base n'ayant pas été reportées dans le fichier de données.
• L'incident se produit pendant l'écriture sur disque du cache de 4D.

Dans tous les cas, relancez votre base avec 4D et identifiez votre situation. Les symptômes correspondant à chaque cas et les opérations de récupération sont décrits dans les paragraphes suivants.

Note : Pour plus d'informations sur le cache de données, reportez-vous à la documentation de 4D.

• Cache vide
Symptômes : Aucun, la base s'ouvre normalement
Procédures de redémarrage : Toutes les modifications apportées à la base ont été enregistrées. Ce cas ne nécessite aucune opération particulière.

• Cache contenant des opérations
Symptômes : Les dernières opérations effectuées dans la base manquent. De plus, si votre base travaillait avec un fichier d'historique, 4D affiche à l'ouverture de la base les messages "Le fichier d'historique contient plus d'opérations qu'il n'en a été effectué sur la base." et "Utilisez 4D Backup pour intégrer ces opérations dans la base."
Procédures de redémarrage :
1. Intégrez le fichier d'historique courant dans votre base pour récupérer les opérations manquantes.
Si vous travailliez sans fichier d'historique, ces opérations sont perdues.
2. Relancez votre base avec votre application 4D.

• Cache en écriture
Symptômes : Il est impossible d'ouvrir la base avec 4D. Le programme affiche un message d'alerte vous signalant des erreurs physiques dans le fichier de données.
Procédures de redémarrage :
Repartez de la dernière sauvegarde de la base :
1. Restituez votre dernière sauvegarde intégrale ou, dans le cas de sauvegardes par miroir logique, recopiez la base miroir sur le poste en exploitation.
2. Si votre base exploitait un fichier d'historique, intégrez l'historique courant dans cette base pour retrouver l'intégralité de vos données.

En cas de perte d'un fichier de la base

La perte d'un ou plusieurs fichier(s) de votre base peut avoir de nombreuses causes : secteurs défectueux sur le disque contenant la base, virus, ou encore effacement par erreur. Bien entendu, si le problème est d'ordre technique, commencez par le rechercher et l'éliminer, par exemple en lançant un utilitaire de détection des blocs abîmés sur votre disque, un détecteur de virus, etc.

Les opérations de redémarrage de l'exploitation diffèrent suivant votre mode d'utilisation de 4D Backup (sauvegardes régulières — avec ou sans fichier d'historique — ou miroir logique).

• Utilisation de sauvegardes intégrales
1. Restituez, à partir de la dernière sauvegarde intégrale de votre base, le ou les fichier(s) perdu(s).
Lorsqu'un fichier de données segmenté est endommagé, vous devez toujours restituer l'ensemble des segments (même si seul un segment a été perdu).
2. Si le fichier de données a été perdu et restitué, il se peut que 4D affiche les messages suivants à l'ouverture de la base :
Message 1 : "Le fichier d'historique contient plus d'opérations qu'il n'en a été effectué dans la base."
Message 2 : "Utilisez 4D Backup pour incorporer ces opérations dans la base."
Dans ce cas, intégrez le fichier d'historique courant dans la base que vous venez de restituer.
3. Si, depuis la dernière sauvegarde intégrale, vous aviez sauvegardé uniquement l'historique (sans le fichier de données), il vous faudra intégrer dans l'ordre les différentes sauvegardes de l'historique (comportant les suffixes "-a", "-b", etc).

• Utilisation d'un miroir logique
Les opérations de redémarrage d'une base sauvegardée à l'aide d'un miroir logique sont simples :
1. Recopiez le fichier de la base miroir sur le poste en exploitation.
2. Intégrez l'historique courant dans la base recopiée.

En cas d'incident lors de la mise à jour du miroir

Si votre base est sauvegardée à l'aide d'un miroir logique, il peut arriver qu'un incident se produise pendant la mise à jour du miroir. L'incident peut avoir pour origine une coupure de courant, un mauvais fonctionnement du réseau, un secteur défectueux sur le disque du poste miroir, etc.

Cet incident, relativement rare, nécessite des procédures de remise en route particulières, suivant la phase de mise à jour dans laquelle se trouvait le miroir.

Au moment du redémarrage du poste miroir, 4D Backup vous signale par des messages d'alerte toute anomalie rencontrée et vous donne des indications sur les opérations à effectuer.

Analyse de la situation

Vous devez connaître deux informations :
• l'état de vos deux bases (la base en exploitation et la base miroir),
• le stade de l'intégration du fichier d'historique au moment de l'incident.

Etat des bases
Vous devez savoir si vos bases sont utilisables ou non. Si l'incident a eu lieu pendant l'écriture sur disque du cache de données de la base en exploitation, la base sera endommagée. De même, si l'incident a eu lieu pendant que la base miroir intégrait le fichier d'historique, la base miroir sera également inutilisable.
Pour vérifier l'état de ces deux bases, ouvrez-les avec votre application 4D, qui vous signalera si la base est ou non en état.

Note : Si, sous MacOS, un fichier "Historique en intégration" se trouve dans le dossier "MaBase•", ou si, sous Windows, un fichier "Restore.4DL" se trouve dans le répertoire "MaBase.MIR", condidérez que votre base miroir est endommagée, même si elle semble s'ouvrir correctement avec 4D.

Quatre hypothèses peuvent donc se présenter :
• Les deux bases sont intactes (mais des fichiers d'historique intermédiaires non détruits empêchent le redémarrage de l'exploitation).
• Seule la base en exploitation est endommagée.
• Seule la base miroir est endommagée.
• Les deux bases sont endommagées.

Stade d'intégration du fichier d'historique
Lors du processus interne de mise à jour du miroir, le fichier d'historique est renommé à chaque étape (pour plus d'informations sur ce processus, reportez-vous au chapitre "Exploitation d'un miroir logique" du *Guide d'utilisation* de 4D Backup).
En conséquence, suivant l'instant de l'incident, un fichier d'historique initalement intitulé "MaBase.log" sous MacOS et "MABASE.4DL" sous Windows pourra avoir les noms suivants :

• Sur le poste en exploitation (dans le répertoire contenant l'historique) :

MacOS	Windows
MaBase.log	MABASE.4DL
MaBase.log.2	MABASE.4L2
Historique à envoyer	SENDING.4DL

• Sur le poste miroir (dans le dossier "MaBase•" sous MacOS, dans le répertoire "Mabase.MIR" sous Windows) :

MacOS	Windows
Historique en réception	RECEIVE.4DL
Historique en analyse	ANALYZE.4DL
Historique en intégration	RESTORE.4DL
Historique miroir	MIRROR.4DL

• Hypothèse n°1 : les deux bases sont intactes
L'incident a eu lieu en-dehors de toute phase d'écriture du cache des données. Cependant, la mise à jour du miroir peut avoir échoué et la présence de fichiers d'historique intermédiaires peut entraîner l'impossibilité de redémarrer. Plusieurs cas sont à distinguer, en fonction du stade d'intégration de l'historique et des fichiers présents sur les postes.

Premier cas

Fichiers sur le poste en exploitation		**Fichiers sur le poste miroir**	
MacOS	**Windows**	**MacOS**	**Windows**
MaBase.log	MABASE.4DL	Aucun	Aucun

Malgré l'incident, la mise à jour s'est déroulée correctement.
Redémarrez l'exploitation.

Deuxième cas

Fichiers sur le poste en exploitation		**Fichiers sur le poste miroir**	
MacOS	**Windows**	**MacOS**	**Windows**
MaBase.log + Historique à envoyer ou MaBase.log.2	MABASE.4DL + SENDING.4DL ou MABASE.4L2	Aucun ou Historique en réception	Aucun ou RECEIVE.4DL

Le fichier d'historique n'a pas été envoyé au miroir (ou pas en totalité).
1. Sur le poste miroir, effacez, s'il s'y trouve, le fichier Historique en réception ou RECEIVE.4DL.
2. Relancez le poste miroir.
3. Relancez la base en exploitation.
4. Effectuez une mise à jour du miroir.

Note : Si le fichier MaBase.log.2 (MacOS) ou MABASE.4L2 (Windows) se trouve sur le poste en exploitation, renommez-le Historique à envoyer (MacOS) ou SENDING.4DL (Windows).

Troisième cas

Fichiers sur le poste en exploitation		Fichiers sur le poste miroir	
MacOS	**Windows**	**MacOS**	**Windows**
MaBase.log	MABASE.4DL	Historique en analyse	ANALYZE.4DL

Le fichier d'historique a été envoyé au miroir mais n'a pas été intégré dans la base miroir.
1. Sur le poste miroir, intégrez Historique en analyse ou ANALYZE.4DL dans la base miroir.
2. Effacez le fichier que vous venez d'intégrer.
3. Relancez le poste miroir.

Quatrième cas

Fichiers sur le poste en exploitation		Fichiers sur le poste miroir	
MacOS	**Windows**	**MacOS**	**Windows**
MaBase.log	MABASE.4DL	Historique miroir	MIRROR.4DL

La base miroir a été mise à jour mais sa sauvegarde intégrale a été interrompue.
1. Sur le poste miroir, effectuez une sauvegarde intégrale de la base miroir.
Placez l'archive dans le dossier "MaBase•" (MacOS) ou "MABASE.MIR" (Windows).
2. Effacez le fichier Historique miroir ou MIRROR.4DL.
3. Relancez le poste miroir.

• Hypothèse n° 2 : seule la base en exploitation est endommagée

Lorsque la base en exploitation est inutilisable, le principe de remise en route consiste à recopier la base miroir sur le poste en exploitation. Plusieurs cas sont à distinguer, en fonction du stade d'intégration de l'historique et des fichiers présents sur les postes.

Premier cas

Fichiers sur le poste en exploitation		Fichiers sur le poste miroir	
MacOS	**Windows**	**MacOS**	**Windows**
MaBase.log + Historique à envoyer ou MaBase.log.2	MABASE.4DL + SENDING.4DL ou MABASE.4L2	Aucun ou Historique en réception	Aucun ou RECEIVE.4DL

Le fichier d'historique n'a pas été envoyé au miroir (ou pas en totalité).
1. Sur le poste miroir, effacez, s'il s'y trouve, le fichier Historique en réception ou RECEIVE.4DL
2. Recopiez le fichier de données de la base miroir sur le poste en exploitation.
3. Intégrez successivement les fichiers Historique à envoyer puis MaBase.log (MacOS) ou SENDING.4DL puis MABASE.4DL (Windows) dans la base que vous venez de recopier.
4. Ouvrez cette base avec votre application 4D et sélectionnez MaBase.log ou MABASE.4DL comme fichier d'historique courant.
5. Relancez le poste miroir.

Note : Si le fichier MaBase.log.2 (MacOS) ou MABASE.4L2 (Windows) se trouve sur le poste en exploitation, renommez-le Historique à envoyer (MacOS) ou SENDING.4DL (Windows).

Deuxième cas

Fichiers sur le poste en exploitation		Fichiers sur le poste miroir	
MacOS	**Windows**	**MacOS**	**Windows**
MaBase.log	MABASE.4DL	Historique en analyse	ANALYZE.4DL

Le fichier d'historique a été envoyé au miroir mais la base miroir n'a pas été mise à jour.
1. Intégrez le fichier Historique en analyse ou ANALYZE.4DL dans la base miroir.
2. Effacez le fichier Historique en analyse ou ANALYZE.4DL.
3. Recopiez le fichier de données de la base miroir sur le poste en exploitation.
4. Intégrez le fichier MaBase.log ou MABASE.4DL dans la base que vous venez de recopier sur le poste en exploitation.
5. Ouvrez cette base avec votre application 4D et sélectionnez MaBase.log ou MABASE.4DL comme fichier d'historique courant.
6. Relancez le poste miroir

• Hypothèse n° 3 : seule la base miroir est endommagée

Vous devez réinstaller le miroir.
1. Détruisez la base miroir inutilisable.
2. Sur le poste en exploitation, détruisez (le cas échéant) les fichiers MaBase.log.2 (MacOS) ou MABASE.4LD (Windows), ainsi que, éventuellement Historique à envoyer (MacOS) ou SENDING.4DL (Windows) désormais inutiles.
3. Recopiez le fichier de données de la base en exploitation sur le poste miroir.
4. Relancez le poste miroir en utilisant le fichier de données que vous venez de recopier.

• Hypothèse n°4 : la base en exploitation et le miroir sont endommagés

Cette situation extrême peut se présenter par exemple en cas de panne de courant alors que la base en exploitation écrit sur disque son cache de données et qu'au même moment la base miroir procède à l'intégration de l'historique
Les fichiers présents sont alors :
• MaBase.log (MacOS) ou MABASE.4DL (Windows) sur le poste en exploitation,
• Historique en intégration (MacOS) ou RESTORE.4DL (Windows) sur le poste miroir.

Le principe de remise en route de l'exploitation consiste à restituer la base à partir de la sauvegarde du miroir, puis d'y intégrer le fichier d'historique.
1. Effacez du miroir le fichier de données endommagé.
2. Restituez la base à partir de la dernière sauvegarde des données de la base miroir et placez-la dans le dossier de la base miroir.
La sauvegarde du miroir se trouve dans le dossier "MaBase•" sous MacOS ou "MABASE.MIR" sous Windows, au premier niveau du disque de sauvegarde du poste miroir.
3. Intégrez dans la base restituée le fichier Historique en intégration (MacOS) ou Restore.4DL (Windows) puis effacez ce fichier.
4. Copiez le fichier de données de la base sur le poste en exploitation.
5. Intégrez le fichier MaBase.log (MacOS) ou MABASE.4DL (Windows) sur le poste en exploitation.
6. Relancez le poste miroir et le poste en exploitation. Sur ce dernier, choisissez le fichier MaBase.log (MacOS) ou MABASE.4DL (Windows) comme historique courant.
L'exploitation peut redémarrer.

Index des commandes

A

D

E

F

H

I

L

M

O

S

U

V